

EVALUATING DATA MINING  
APPROACHES FOR THE  
INTERPRETATION OF LIQUID  
CHROMATOGRAPHY-MASS  
SPECTROMETRY

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER  
FOR THE DEGREE OF MASTER OF SCIENCE  
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2013

By  
Daniel Addison  
School of Computer Science

# Contents

<b>Abstract</b>	<b>8</b>
<b>Declaration</b>	<b>10</b>
<b>Copyright</b>	<b>11</b>
<b>Acknowledgements</b>	<b>12</b>
<b>1 Introduction</b>	<b>13</b>
1.1 Motivation . . . . .	13
1.2 Aims . . . . .	14
1.3 Objectives . . . . .	15
1.4 Contributions of this Project . . . . .	15
1.5 Structure of the Dissertation . . . . .	16
<b>2 Application Background</b>	<b>18</b>
2.1 The Drug Discovery Process . . . . .	18
2.2 Liquid Chromatography-Mass Spectrometry (LC-MS) . . . . .	19
2.2.1 Applications of LC-MS in Drug Discovery . . . . .	20
2.3 Quality Assurance by Compound Management (CM) . . . . .	22
2.4 LC-MS by Compound Management UK, AstraZeneca . . . . .	23
2.4.1 Structure of LC-MS Data . . . . .	23
2.4.2 Workflow for LC-MS Data Generation . . . . .	24
2.4.3 Interpretation and Storage of LC-MS Data . . . . .	26
2.4.4 LC-MS Interactive Review Interface . . . . .	28
2.5 Applications of Data Mining in Drug Discovery . . . . .	31
2.5.1 Decision Tree Induction to Support Drug Discovery Processes . . . . .	31

<b>3</b>	<b>Research Methods</b>	<b>36</b>
3.1	Overview of Knowledge Discovery from Data (KDD)	36
3.1.1	Supervised and Unsupervised Approaches to Data Mining	37
3.2	The KDD Process	39
3.2.1	Class Labels	41
3.2.2	Creation of Data Sets	42
3.2.3	Data Preprocessing and Transformation	43
3.2.4	Attribute Selection	47
3.2.5	Descritisation of Numerical Attributes	47
3.2.6	Cross-validation	48
3.3	Learning Algorithms	49
3.3.1	Probabilistic Classification	49
3.3.2	Decision Trees	50
3.3.3	Decision Tree Pruning	55
3.3.4	Ensemble Methods of Classification	59
3.3.5	Support Vector Machines (SVM)	65
3.3.6	Neural Networks (NN) and Backpropagation	65
3.4	Evaluation	68
3.5	Cost-sensitive Classification	70
3.5.1	Cost-sensitivity by Stratification	70
3.5.2	Cost-sensitivity via a Cost-matrix	71
3.5.3	MetaCost	72
3.6	Summary of Classification Models	72
<b>4</b>	<b>Experiments</b>	<b>75</b>
4.1	Pruning and Model Stability	75
4.1.1	J48 Pruning	77
4.1.2	BFTree Pruning	77
4.1.3	SimpleCART Pruning	78
4.1.4	Functional Tree and Logistic Model Tree pruning	79
4.1.5	Summary of Results for Pruning Methods	80
4.2	Evaluation of Classification Models	81
4.3	Cost Sensitive Classification using MetaCost	83
4.3.1	Analysis of the effect of MetaCost on classification performance	85
4.4	Variable Importance	87
4.4.1	Variable Importance using chi-squared	89

4.4.2	Variable Importance using Elimination by Pre-pruning . . . . .	89
4.4.3	Random Forest Implementation using Pipeline Pilot . . . . .	92
<b>5</b>	<b>Discussion</b>	<b>95</b>
5.1	Discussion of Results . . . . .	95
5.1.1	A Review of the Current Auto-pass Function . . . . .	96
5.1.2	Application of Classification Models into the LC-MS Workflow	99
5.1.3	Data Mining as a Complimentary Approach to Quality Assurance	102
5.1.4	Further Work . . . . .	103
	<b>Bibliography</b>	<b>105</b>
<b>A</b>	<b>Techniques and Components of LC-MS Analysis</b>	<b>114</b>
A.1	Liquid Chromatography (LC) . . . . .	114
A.2	Diode Array Detection (DAD) . . . . .	114
A.3	Charged Aerosol Detector (CAD) . . . . .	115
A.4	Mass Spectrometry (MS) . . . . .	115
A.5	Mass Analysers . . . . .	116
<b>B</b>	<b>LC-MS Data management system</b>	<b>118</b>
<b>C</b>	<b>Peak of Interest function</b>	<b>122</b>
<b>D</b>	<b>Confidence measures for Random Forest Models</b>	<b>123</b>
<b>E</b>	<b>Full Decision Trees Discussed in Chapter 5</b>	<b>125</b>

Word Count: 24,580

# List of Tables

2.1	Attributes recorded for each integrated peak detected by LC-MS analysis	25
3.1	Stratified Training and Test Data Partitions . . . . .	43
3.2	Comparison of Classification models. . . . .	74
4.1	Tree complexity and performance using different pruning methods . .	76
4.2	Performance of Classification Models . . . . .	82
4.3	Performance of Classification Models using MetaCost cost-sensitivity	85
4.4	Variable Importance measured using Chi-squared . . . . .	90
4.5	Performance of Classification Models . . . . .	91
4.6	Performance of Random Forest Models using Pipeline Pilot . . . . .	93

# List of Figures

2.1	LC-MS Instrument Configuration . . . . .	24
2.2	Total number of peaks integrated in LC-MS analyses . . . . .	26
2.3	An overview of the current Compound Management LC-MS workflow adapted from Charles et al 2013 [CSA] . . . . .	27
2.4	LC-MS manual review workflow adapted from Charles et al 2013 [CSA]	29
3.1	Overview of KDD process . . . . .	39
3.2	Class labels applied to LC-MS integrated peaks . . . . .	42
3.3	An example J48 Decision Tree showing splitting nodes (ellipses) and leaves (boxes) . . . . .	51
3.4	Path through DT represented as a set of rules . . . . .	51
3.5	Path through DT simplified as a set of rules . . . . .	51
4.1	Pruning methods for the J48 DT algorithm . . . . .	78
4.2	Pruning methods for BFTree algorithm . . . . .	79
4.3	Pruning methods for SimpleCart DT algorithm . . . . .	80
4.4	FT and LMT . . . . .	81
4.5	Decision Stump output . . . . .	83
4.6	Examples of ROC charts for classification models from Table 4.2 . . .	84
4.7	Examples of ROC charts for MetaCost models from Table 4.3 . . . .	86
4.8	A comparison of false negative rates for cost-insensitive and MetaCost classification models . . . . .	87
4.9	A comparison of false positive rates for cost-insensitive and MetaCost classification models . . . . .	88
4.10	Changes in false positive and false negative rates for cost-insensitive and MetaCost classification models . . . . .	88
4.11	Classification and regression tree generated by JMP . . . . .	90

4.12	Effect of Random Forest tree number on the percentage of high confidence (>0.9) predictions . . . . .	94
5.1	Extract from J48 . . . . .	97
5.2	Extract from J48 with Bagging . . . . .	97
5.3	Extract from REPTree . . . . .	98
5.4	Detection of peak integration errors using the <i>get_closest_peak</i> attribute . . . . .	99
5.5	Comparison of automatic and manual reviews for current LC-MS system versus Random Forest approach . . . . .	100
B.1	Workflow Management System . . . . .	119
B.2	Analysis review system . . . . .	120
B.3	Search engine . . . . .	121
D.1	Distribution of confidences for predictions by Random Forest (10 trees) . . . . .	124
E.1	J48 Decision Tree output . . . . .	126
E.2	J48 Bagging Decision Tree output (example from 10 trees in ensemble) . . . . .	127
E.3	REPTree output . . . . .	128

# Abstract

## EVALUATING DATA MINING APPROACHES FOR THE INTERPRETATION OF LIQUID CHROMATOGRAPHY-MASS SPECTROMETRY

Daniel Addison

A dissertation submitted to the University of Manchester  
for the degree of Master of Science, 2013

Over the past three years quality assurance activities have been performed by Compound Management UK on AstraZeneca's compound collection using an in-house visualisation and reviewing system. These activities involve the structural identification and purity evaluation of small molecule research compounds using a combined liquid chromatography-mass spectrometry (LC-MS) method. The results of LC-MS analysis provide a pass/fail characterisation for each sample analysed, with low quality samples being removed from the primary screening subset or from the compound collection altogether.

Whilst some post-analysis interpretation of LC-MS data has been automated, the majority (66%) required manual review by an experienced technician. The data generated from the current process offers an opportunity to apply data mining techniques to accurately identify LC-MS analytical results as pass or fail, and therefore minimise the number of analyses that require review by a domain expert.

Following the Knowledge Discovery from Data (KDD) process, a number of supervised learning approaches are investigated, including *Naïve Bayes*, *Decision Tree induction* (DT), *Random Forests* (RF), *Support Vector Machines* (SVM) and *Neural Nets* (NN). Model complexity, tree pruning and cost-sensitive classification are also investigated.



Experiments using the WEKA software platform found all classification models (with the exception of ZeroOne) performed with high accuracy, with F-measure values ranging from 0.949 (Decision Stump) to 1 (Random Forest). Random Forests made the most accurate predictions, with F-measures of 0.999 for a 5 tree ensemble and 1 for 10, 20 and 50 tree ensembles.

The implementation of Random Forests in Pipeline Pilot did not perform as accurately as the WEKA models, but was considered due to the ability to generate confidence statistics on each prediction. The 10 tree ensemble using this software was found to increase the number of analyses that would not require review from 33.8% using the existing system to 63.6% using a threshold of 90% confidence in predictions. Recommendations on how this method could be incorporated into the existing LC-MS workflow management system as discussed.

# Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

# Copyright

- i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.
- ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.
- iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.
- iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see <http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487>), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see <http://www.manchester.ac.uk/library/aboutus/regulations>) and in The University’s policy on presentation of Theses

# Acknowledgements

I would like to thank Professor John Keane for his invaluable support throughout this project and for his patience in supervising a part-time student. I would also like to thank my colleagues at AstraZeneca, Dr. Isabel Charles for her encouragement, guidance and unmatched eye for detail, both Isabel and Dr. Clive Green for accommodating my pursuit of an MSc and to Ian Sinclair and Sunil Sarda whose expertise in interpreting analytical data was essential to this project. This dissertation is dedicated to Vanessa Addison, a continuing source of inspiration.

# Chapter 1

## Introduction

### 1.1 Motivation

High Performance Liquid Chromatography (HPLC) coupled with Mass Spectrometry (MS) has long been established as a technique to provide structure elucidation and purity determination of large libraries of research compounds. It is essential that accurate information about the structure and purity of research compounds is established. This allows structure-activity relationships observed during screening to be correctly attributed to the target compound and not to impurities, degradants or the presence of an unknown substance. Increased pressure within the pharmaceutical industry to minimise the time taken to discover and develop new drugs has focused efforts on quality assurance by increasing instrument utilisation and throughput and therefore reducing the cycle time of sample analyses. Success in this area of research has meant that the rate-limiting steps are now data processing, storage and interpretation of analytical data.

Liquid chromatography-mass spectrometry (LC-MS) activities undertaken by the UK Compound Management (CM) group within AstraZeneca are characterised by a high throughput of several thousand analyses per week. A semi-automated system has been developed within CM that greatly increases the efficiency with which these data can be interpreted and is capable of automatically passing compounds that are confirmed as the correct mass and are found to be above a purity threshold. The function used to automatically pass analyses is a conservative one testing five features of the data before a pass is confirmed. As a result, many of the analyses still require a manual review stage, which is time-consuming, manual and experience-dependent, hence costly. The total number of analyses included in this project is 350,153 of which

231,160 were reviewed by a domain expert. Given an approximate time for each review of 1 minute, a total of 3,853 hours were spent reviewing these results.

Any data analysis capable of increasing the ability to confirm the results as a *pass* (optimising the current method of confirming a pass) or *fail* (introducing a new classification to the existing system) would be prima facie of value here. By automating both ends of the validation spectrum the actual number of analyses that require a manual review stage should be reduced, therefore improving the effectiveness and efficiency and reducing costs, whilst retaining the quality of the data generated on collection quality.

The classification models in this project were trained using data that was annotated by expert LC-MS technicians. As such, it is not expected that the number of correct classifications will be improved by using a data mining approach but rather the aim is to automate reviews that would otherwise have been performed manually.

## 1.2 Aims

The aim of this project is to assess data mining approaches for the interpretation of LC-MS analysis of screening compounds. *Supervised learning* approaches will be investigated with the aim of learning value-adding knowledge about the quality of AstraZeneca's compound collection. Supervised learning involves the classification of data according to a label, in this case a *pass* or *fail* result for LC-MS analysis. This class is said to supervise the learning which, if successful, will generate a classification model capable of accurately predicting the class outcome of future LC-MS analyses using the values of the other attributes. This would reduce the number of analyses that require manual reviewing by a specialist technician.

In-house software is currently in place that automatically flags the results of an analysis as passed if a number of criteria are satisfied, such as purity as detected by ultraviolet (UV) absorbance and spectral purity detected by MS. Due to the conservative approach currently in place, only approximately 33% of analyses have been automatically flagged as passed. The remaining analyses are provisionally passed or failed by the system and are subsequently reviewed manually to confirm the result. Over 80% of manual reviews confirm the pass/fail status suggested by the system. However, occasionally a manual inspection of the data results in the reviewer overwriting the suggested classification. This can be due to anomalies in peak integration by the LC-MS instrument software, chemical instability in the solvent used, errors in molecular weight determination due to pre-charged compounds or instrument errors during

processing. It is hypothesised that patterns may exist in the data that could allow such results to be identified and that this knowledge may be derivable via supervised learning methods. As such, an attempt will be made, using the Knowledge Discovery in Databases (KDD) process and in particular data mining techniques, to construct classification models which allow such results to be automatically identified.

## 1.3 Objectives

The objectives of this project are:

1. Copy data from the operational LC-MS database developed within CM to a development environment to allow preparation of training/test data sets to be used in data mining assessments.
2. Assess supervised learning methods, including simple *Classification Rules*, *Decision Trees* (DT), *Support Vector Machines* (SVM), *Neural Networks* (NN) and *ensemble* techniques, to produce a classification model that will accurately predict LC-MS analyses as passed or failed.
3. Use these classification models to optimise the current system of automatically passing high quality LC-MS results and introduce an additional system of automatically failing LC-MS results, thereby reducing the overall number of analyses that require a manual review.

## 1.4 Contributions of this Project

This project was undertaken to improve the efficiency of quality assurance activities performed by Compound Management UK, AstraZeneca. The large set of analytical data gathered from three years of LC-MS activities, much of which has been annotated and verified by domain experts, provided an opportunity to apply data mining techniques to ‘learn’ what features of the data contribute to the pass or fail outcome of an analysis. By encoding this expert knowledge into a learner model, the number of future analyses that require review could be significantly reduced. The data from the operational LC-MS database was extracted, cleaned and used to create training/test data sets to be used to construct and evaluate various classification models. A diverse set of models was selected, representing the range of algorithm types that have been

shown to be effective in this domain. The training/test data sets were cleaned to remove erroneous data and stratified to ensure that the various outcomes of analysis were in the same proportions as the entire data set, to avoid overfitting a particular outcome. Models were constructed using the WEKA software suite and assessed using a range of metrics including precision, recall and ROC area and by a comparison of ROC curves. Each model was evaluated first as a cost-insensitive classifier and again using MetaCost to apply cost-sensitivity. Various methods of pruning were also investigated. Variable importance analysis was done using the chi-squared measure.

The experiments performed using WEKA showed that the Random Forest performed with the highest accuracy, with the 50 tree ensemble making no misclassifications on the test data used. A Random Forest model was also built using Pipeline Pilot, which is the desired platform as it is already used extensively in the current LC-MS system. Confidence statistics on predictions made by this model showed that using a confidence threshold of 90%, the number of reviews that would not require manual review was increased from 59,497 to 111,806 or from 33.8% to 63.6%, a reduction of 52,309 analyses or 29.8%.

## 1.5 Structure of the Dissertation

The structure of the dissertation is organised into the following chapters:

- *Chapter 2* begins with an introduction to drug discovery activities within the pharmaceutical industry. The emergence of various analytical techniques are discussed, focusing on Liquid Chromatography-Mass Spectrometry (LC-MS). A brief overview is provided on the application of these techniques to the drug discovery process followed by detailed information on the LC-MS activities currently performed by CM UK, AstraZeneca. Further background information is given on how data mining processes have been applied in the field of LC-MS analysis. This chapter is intended to provide an overview of the domain in which this dissertation was undertaken.
- *Chapter 3* looks in detail at the Knowledge Discovery from Data (KDD) process utilised in this dissertation. An examination of the various methods of data mining are provided, along with the techniques used for evaluation of the various classification models produced.



- *Chapter 4* contains the results of the experimental work on pruning methods, evaluation of various data mining algorithms, the effect of cost-sensitive classification and the results of experimental work on variable importance.
- *Chapter 5* contains a discussion of the results and conclusions drawn. Information on the opportunities for increased automation of LC-MS interpretation is provided, with an assessment of the potential impact in terms of cost-savings. Opportunities for further work are also considered.

# Chapter 2

## Application Background

This chapter provides background information on the drug discovery process and the use therein of various analytical techniques. Particular focus is given to Liquid Chromatography-Mass Spectrometry (LC-MS) and the application of this method to the domain studied in this project, quality assurance activities by Compound Management UK, AstraZeneca (CM). The current workflow for LC-MS activities and the system for data storage and interpretation developed by CM is described in detail. The chapter concludes with a discussion of how data mining methods have been applied to the interpretation of analytical data within the pharmaceutical industry.

### 2.1 The Drug Discovery Process

The pharmaceutical industry has for decades sought to discover new drugs for the treatment of human diseases. The drug development ‘pipelines’ of pharmaceutical companies are populated with potential medicines to treat all major therapeutic areas such as oncology, neurology, gastro-intestinal and cardio-vascular. Huge investments in pharmaceutical R&D have fuelled a number of paradigm shifts in the nature of drug discovery, all of which have aimed to increase the efficiency with which new drugs are discovered whilst maintaining or even reducing costs [GA03]. The most transformative of the recent trends in drug discovery have been the development of high-throughput technologies, multiple parallel synthesis, informatics, computational chemistry and miniaturisation [KYO03]. In the environment that these technologies have created, speed, efficiency and quality are key. Quality of materials, of data and in decision-making are all of critical importance. If a drug project is ultimately headed for failure vast amounts of money and resources can be saved by recognising this as

early as possible.

The modern drug discovery process follows a well-established sequence of identifying, optimising and developing research compounds that target molecular pathways with a causal role in disease [KYO03]. The process often begins with a high throughput screening (HTS) campaign where thousands, sometimes millions of compounds are screened, and results in a single candidate drug entering human clinical trials [GA03], [Kor05]. A pharmaceutical company's compound collection is therefore a major asset, to the extent that it is sometimes referred to as the "crown jewels". It is by exploiting this reservoir of chemical entities that pharmacologically active structures are discovered, which may ultimately lead to the introduction of new healthcare products and treatments.

## 2.2 Liquid Chromatography-Mass Spectrometry (LC-MS)

As the size of compound collections has increased, so too has the need for rapid analytical methods of quality assurance. A number of techniques have been developed to achieve this, including (but not limited to) Liquid Chromatography (LC) and Mass Spectrometry (MS) [Süß99] [KCW01] [KYO03], nuclear magnetic resonance (NMR) [DVMWK99] and charged aerosol detection (CAD) [SG09] [CS09]. A common requirement for many of these methods is that they can be integrated into workflows characterised by highly automated systems capable of high throughputs. As instrumentation has become more sophisticated with the introduction of automated compound storage [Yat03] and nano-litre dispensing, so too have the demands on data handling and interpretation. The use of reference databases and data mining techniques to cope with the deluge of information generated by these highly automated systems is of growing importance, with manual forms of probing data becoming increasingly impractical [FPSS96].

Ideally, an analytical detection method would be available which would show a proportionate response and high sensitivity to compounds irrespective of structure so that compound identity and purity measurements could be gathered with ease [CS09]. In reality, none of the aforementioned techniques used independently provide such a universal method of analysis. For example, although many drug-like compounds contain structural characteristics that show strong ultraviolet (UV) absorbance, some contaminants may be invisible in the UV wavelength typically examined (220-300 nm).

By contrast, other contaminants could have much higher absorbance and thus provide misleading results [CS09]. Other analytical methods also show similar structure-dependent strengths and weaknesses. For decades, analytical techniques used for quality assurance within the pharmaceutical industry have therefore often been combined [Swe02], typically MS combined with a chromatographic method. Early examples from the 1960s, 70s and 80s involve the combination of Gas Chromatography and Mass Spectrometry (GC-MS) [GV03] but since then many other disparate techniques have been used in combination. The diversity of these techniques reflect numerous and varied applications within the drug discovery process, with each hyphenated technique combining different strengths into powerful, high speed analytical processes. Applications include drug metabolism and pharmacokinetics (DMPK) both *in vitro* and *in vivo*, analysis of degradations, metabolite identification and pharmacokinetic profiling. More recently, one of the most prominent of these hyphenation techniques to emerge has been LC-MS.

High Performance Liquid Chromatography-Mass Spectrometry (HPLC-MS) is now established as the method of choice for structural characterisation of small molecule libraries due to the combination of the high resolving power of HPLC and the superior mass detection capability of MS [CPL07], [KCW01]. Unlike gas chromatography, HPLC is compatible with almost all drug-like compounds without the need for derivatisation prior to analysis and avoids some of the problems with other analytical techniques; infrared spectroscopy is not sufficiently specific and NMR suffers from lower throughputs and sensitivity limitations [Shi01]. Comparisons made between MS and UV detection have shown that some library compounds are only detectable in one or the other detection mode so exclusive use of only one of these methods may be insufficient for reliable library characterisation [Süß99]. Furthermore, UV detection can generate information on compound purity and so a UV detector to generate LC/UV chromatograms is commonly incorporated into HPLC-MS workflows [CPL07], [Kor05].

### 2.2.1 Applications of LC-MS in Drug Discovery

Mallis et al 2002 [MSKW02] have described how an open-access HPLC-MS system based on a single quadrupole mass spectrometer has been used for aiding medicinal chemists as part of new drug discovery. Other systems have been developed that incorporate an automated purification step, where software-controlled fraction collectors are triggered when the expected MS signals of the compound of interest are detected

[IXC02]. This allowed the development of a highly automated system which included sample handling, analysis, purification and compound reformatting, through to final submission of the reformatted plates. The system also generated electronic files required for automated compound registration. With this system one scientist was able to completely process a 15-plate library in 11 days [IXC02]. Other studies have also highlighted how analytical processes have been accelerated by computer-controlled automation, both in characterisation/purification of compounds from combinatorial libraries and outputs from drug metabolism/pharmacokinetic (DMPK) studies [PS01].

### 2.2.1.1 Drug Metabolism and Pharmacokinetics (DMPK)

DMPK is another area of drug development that has seen a strong focus on LC-MS analytical techniques. Once a lead compound has been identified from an HTS campaign, a second stage of screening begins, the aim of which is to explore the chemical properties of the compound, for example solubility, permeability and chemical stability. DMPK studies reveal the absorption, distribution, metabolism and excretion (ADME) properties of lead compounds as well as various other pharmacokinetic properties [Kor05]. Particular focus has been given to these activities as inadequate ADME properties are responsible for a large proportion of drug candidates failing to reach the market. DMPK studies have therefore been shifted as early as possible in the drug discovery process in order to 'front-load' drug portfolios with compounds that are more likely to successfully negotiate the later, more expensive stages of development [EB00]. This shift has produced an intensified demand on DMPK resources, which require more high-speed, high-throughput analytical techniques [PS01]. HPLC-MS has been applied to meet these requirements and has been able to do so due to the rapid and robust nature of these methods and the opportunities for integration and automation that they provide.

Many examples have been reported of highly automated LC-MS systems for DMPK that are capable of running with high levels of automation. Korfmacher et al 1999 [KV99] described a parallel HPLC coupled with tandem MS (parallel LC-MS/MS) where two HPLC systems were linked, with the combined eluent flowing into one MS system capable of de-convoluting data to calculate results. Another study by Fung et al 2003 [FCL<sup>+</sup>03] showed how an HPLC-MS/MS system was used in a DMPK study of the human colon adenocarcinoma cell line (Caco-2), a common assay used to determine the permeability of a compound. The use of this system allowed 100 compounds to be screened in a Caco-2 assay per week. Xu et al 2002 [XNJR02] have demonstrated

a highly automated process for a metabolic stability study. The assay was performed using an eight-probe auto-sampler and eight HPLC columns in a parallel mode and was able to assay 240 samples per hour. Despite each sample producing multiple results to construct a calibration curve, the system was able to evaluate 176 test compounds per day. In order to maximise throughputs a data management system was developed to automate post-analysis data handling. This tool was able to import result files, de-convolute the eight channel result, generate stability plots for each compound and create a summary report for the whole plate. The tool also performed validation on each compound result and generated a flag for any compound with an incorrect stability trend, low MS signal, or broad chromatographic peak. This data processing tool reduced the post-analysis data processing time from approximately one day per plate to only a few minutes [XNJR02].

### 2.3 Quality Assurance by Compound Management (CM)

Recent years have seen major investments in CM activities ([WR11], [HBB<sup>+</sup>08]) which has accompanied a growth in size and diversity of compound collections [Shi01]. The synthesis of new chemical entities (NCE's) has gone from a rate-limiting step to a rate-setting one and associated downstream processes have had to undergo stepwise improvements to keep pace. One such focus of a CM group is to carry out quality assurance activities with regard to the compound collection. It is essential to the success of the drug discovery pipeline that the compounds entering screening have had the identity of the structure confirmed and that the sample is above a purity threshold. Without these two checks, if a compound appears to be active in an assay it would not be possible to rule out a situation where the activity detected is due to an unknown impurity and not the compound being tested, or that the compound is active but is in fact a different structure altogether. Both cases would lead to wasted effort, inaccurate data and difficulties in establishing quality structure-activity relationships as the project developed [Ken04].

Quality assurance activities by CM groups go beyond compound identification. The stability of compounds stored as dimethyl sulfoxide (DMSO) solutions must be monitored periodically as compounds may be stored for long periods of time, potentially years. It may also be advantageous to provide 'just-in-time' analyses to support secondary screening activities [CS09]. In all cases the requirements for this type of analysis are characterised by speed and high-throughputs. In the absence of sufficient

capacity to achieve this, organisations may be forced to analyse a statistically relevant subset of samples prior to HTS screening and infer assumptions on the quality of the wider compound collection from these results [DVMWK99].

## 2.4 LC-MS by Compound Management UK, AstraZeneca

The following section provides an overview of the LC-MS workflow used by CM UK at AstraZeneca to perform compound identification and purity analysis. The basic components of an MS system are an ionisation source, a mass analyser and a detector, as shown in Figure 2.1. A description of the structure of the analytical data generated by LC-MS is provided. A more detailed description of each technique is included in Appendix A.

### 2.4.1 Structure of LC-MS Data

LC-MS systems perform separation via LC followed by MS analysis. Each chromatographic peak is eluted (sequentially separated over time by chromatography) from the LC column and diverted to the MS system for analysis [Kor05]. The results of an analysis can be regarded as hierarchical in that there is analytical information provided at the sample level (LC/UV/CAD chromatographic spectra) and MS spectra for each peak separated by the chromatography. Pooling all these data allows structural identity (MS), purity (UV) and concentration (CAD) to be determined. When applied to the task of structure identification and purity determination of small molecule compound libraries, the desired results would show only one prominent peak in the UV chromatogram with a high UV percentage purity (this would mean that there are no contaminants detected which would show as additional peaks, and that a high purity has been confirmed by UV detection). In addition, the MS spectra for the integrated peak would show a high spectral purity in either the positive and/or negative ion modes, depending on the acidic/basic characteristics of the compound. A high spectral purity around the expected molecular weight (with either +1 mass unit (mu) increment for protonated ions or -1 for de-protonated ions) would confirm the identity of the compound. Other common adduct patterns may also be considered. A high number of peaks in the MS spectra would indicate breakdown, fragmentation, contamination or adduction of the compound. In other applications such as protein identification LC-MS, many peaks will be integrated by LC, indicating a complex mixture of proteins or

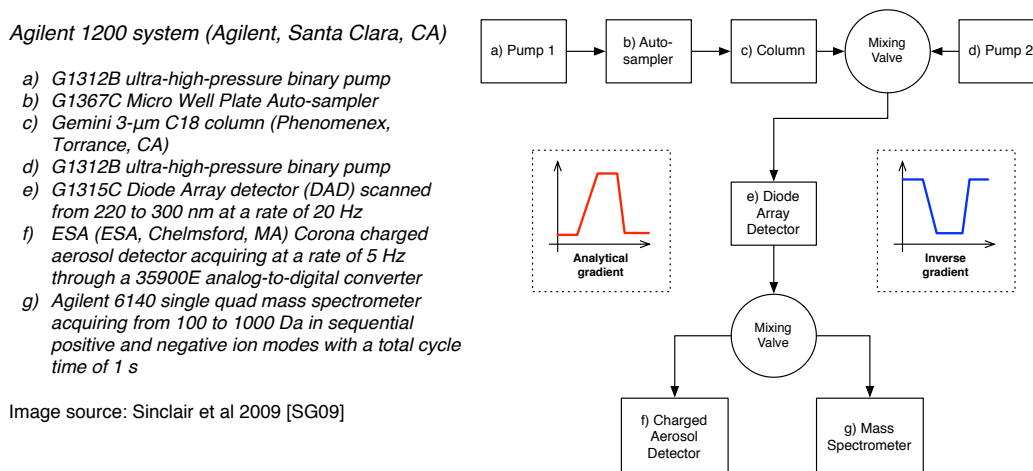


Figure 2.1: LC-MS Instrument Configuration

constituent peptides separated by the LC process.

## 2.4.2 Workflow for LC-MS Data Generation

LC-MS activities carried out by CM within AstraZeneca are characterised by a high throughput of over 25,000 analyses per month. With such throughputs, large amounts of analytical data are generated which create challenges regarding storage and interpretation. These issues were somewhat addressed with the introduction of a semi-automated system developed within CM that greatly increases the efficiency with which data can be interpreted [CSA]. The system is capable of automatically passing compounds that pass five measured criteria and are therefore confirmed as the correct mass and above a purity threshold (85%). To avoid falsely passing incorrect or impure compounds, this function currently has a very conservative configuration and as such, 60-65% of analyses still require a manual review stage, which is time-consuming, manual and experience-dependent.

The HPLC-MS strategy is designed primarily for mass determination and purity measurements of DMSO solutions, with the addition of a CAD detector for concentration measurements [SG09]. The hardware configuration is shown in Figure 2.1.

Each analysis generates a report either as a .rpt file (Waters™) or a .asr file (Agilent™). The data in these reports has been preprocessed by the LC-MS manufacturers' software. Data is filtered out from the output file according to a configurable signal to noise ratio. The system also performs peak integration on both the UV and CAD



chromatograms, which are encoded in the report as x/y coordinates of percentage base peak versus retention time. A full list of the attributes captured for each integrated peak are provided in Table 2.1.

Table 2.1: Attributes recorded for each integrated peak detected by LC-MS analysis

Attribute name	Description
<i>Percent ES pos</i>	Spectral purity as the percentage of the expected molecular weights (ESI positive ion mode)
<i>Percent ES neg</i>	Spectral purity as percentage (ESI negative ion mode)
<i>Percent AP pos</i>	Spectral purity as percentage (APCI positive ion mode)
<i>Percent AP neg</i>	Spectral purity as percentage (APCI negative ion mode)
<i>ESI</i>	Boolean flag to indicate that an ESI peak was detected
<i>Retention time</i>	The time at which the peak was eluted during chromatographic separation
<i>Percent purity by UV</i>	Percent purity as determined by UV absorbance
<i>UV peak height</i>	Height of the peak on the UV chromatogram
<i>UV peak width</i>	Width of the peak on the UV chromatogram
<i>CAD start/stop</i>	Start/stop times of peak integration in the CAD chromatogram
<i>DAD start/stop</i>	Start/stop times of peak integration in the DAD chromatogram
<i>CAD peak area</i>	Area of integrated peak as a percentage of total integrated area
<i>CAD peak width</i>	Width of integrated peak in CAD chromatogram

Each integrated peak produces a set of four MS spectra, two for the positive and two for the negative ion modes (one each for electrostatic spray ionisation and atmospheric pressure chemical ionisation), which are encoded as x/y coordinates of percentage base peak versus mass. The peaks are compared with pre-defined adduct patterns for the expected mass of the compound being tested. The adduct patterns are Mass (M) +1 (compound plus proton) and M+23 (compound plus sodium ion) for the positive ion, and M-1 (compound minus proton) for the negative ion. Sodium adducts are often encountered due to the solvent used in the HPLC being stored in glassware from which sodium ions leach, and are therefore included when analysing results for a confirmed identification. The sodium adduct can also provide a useful secondary species to confirm the protonated molecular ion.

The number of integrated peaks per sample for all analyses in the experimental database are summarised in Figure 2.2.

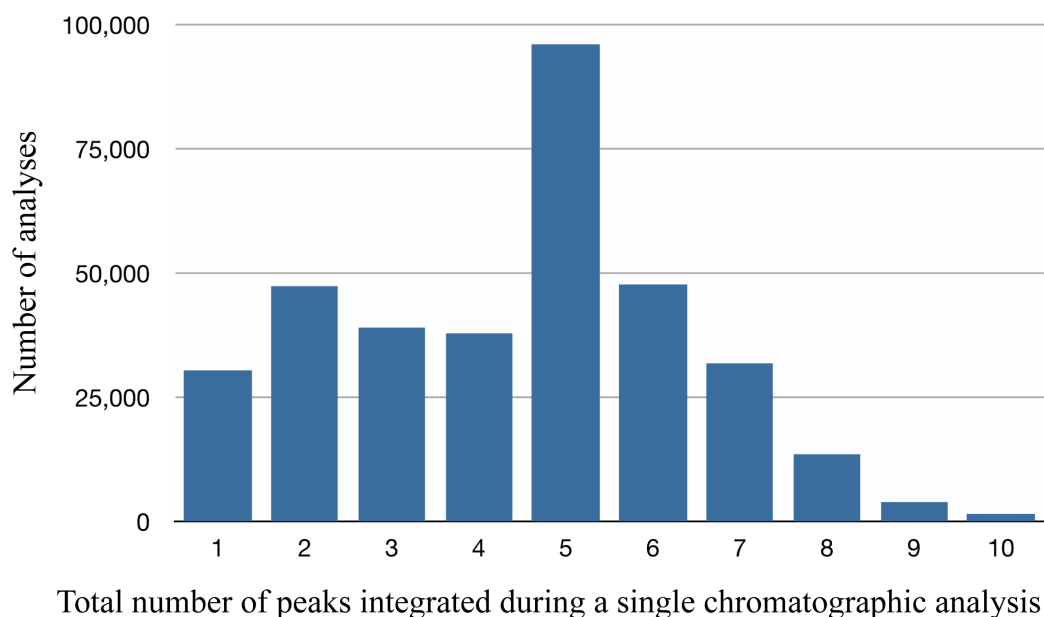
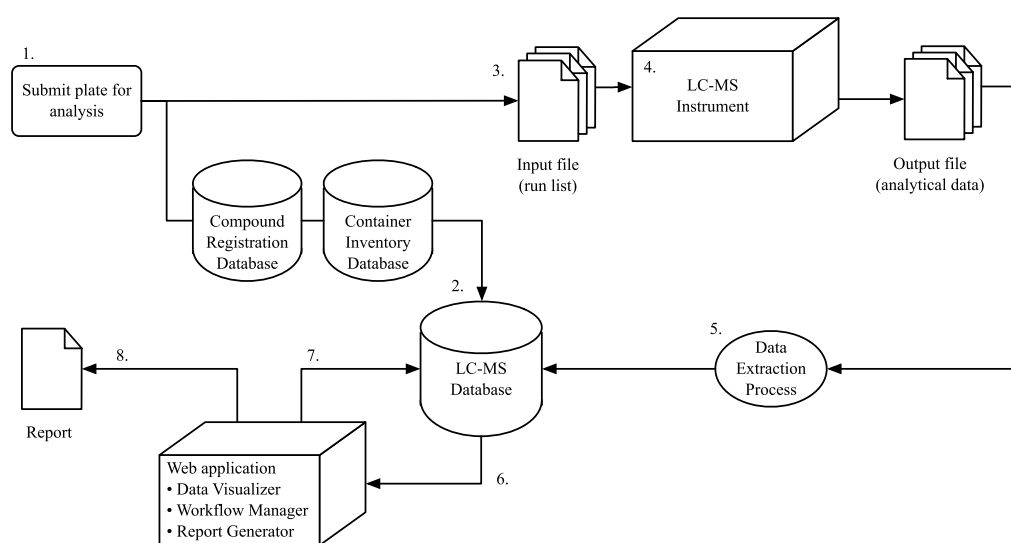


Figure 2.2: Total number of peaks integrated in LC-MS analyses

### 2.4.3 Interpretation and Storage of LC-MS Data

HPLC-MS analyses are performed as part of a highly automated, high-throughput workflow with an accompanying IS infrastructure summarised in Figure 2.3. The system consists of three main components; a relational database to store data generated by LC-MS activities; a set of web services implemented as Pipeline Pilot™ (Pipeline Pilot version 8.5.0.200, Accelrys, Inc., San Diego, CA) protocols to perform back-end processing (initial record creation in the database, generation of run files to drive the LC-MS instrumentation and extraction/uploading to the database of pertinent data from analysis results); a web application used for workflow management, visualising and reviewing analytical data via an interactive graphical user interface, a simple search engine and a report generation system (Appendix B).

Addressing problems of data storage and interpretation, this system allows pertinent information to be extracted from the report files generated by the LC-MS equipment from both vendors. Although the data generated is in two different formats, the extraction process allows the data to be converted into a standardised format before being inserted into the database. Analytical data are integrated with data from the global sample and compound database (for example sample identifiers, structural information, molecular weight) and from a global container inventory (plate barcode and



1. A 96 or 384 well plate is created for analysis.
2. Data for the plate is added to the database and new record ids are created for each well.
3. Run files to drive the LC-MS instruments are generated via a Pipeline Pilot protocol.
4. LCMS analysis is performed and output data generated by the instrumentation.
5. Pertinent data is extracted from the output data files via a Perl script executed within a Pipeline Pilot protocol. Extracted information is used to update the records in the database with the analysis results.
6. Analyses that are not automatically passed are reviewed manually via a web application.
7. Analyses are annotated and marked as passed or failed via an interactive graphical user interface.
8. Summary reports can be generated via the web application.

Figure 2.3: An overview of the current Compound Management LC-MS workflow adapted from Charles et al 2013 [CSA]

well position, creation date and source vessel from which the test well was created). By using a set of customisable parameters, functions on the database are used to automatically pass an analysis. Firstly, a “peak of interest” is selected from all the peaks detected and integrated in the chromatographic data. Both manufacturers provide software packages that attempt this peak selection using different strategies. The Agilent system selects the first peak eluted where the expected mass is recognised. The Waters system chooses the largest peak eluted. Neither of these strategies are satisfactory in all situations so a more complex system has been developed (Appendix C).

Once the peak of interest has been identified an automatic pass function is applied, which tests whether the expected mass has been detected and whether minimum quality thresholds have been reached. A total of five parameters must be satisfied before a sample is flagged as automatically passed (percent purity UV, UV peak height, UV peak width, CAD area and spectral purity). If at least one of these values does not reach a threshold then the analysis is flagged as requiring manual review. The percent purity UV provides a value for purity and spectral purity confirms the identification of the analyte, however these two measures alone are insufficient to pass the analysis automatically. The other three values are considered as supporting evidence of purity and identity and so only a pass on all five measures results in the auto-pass flag being applied.

#### 2.4.4 LC-MS Interactive Review Interface

A web application allows data from the database to be visualised, reviewed and annotated (Appendix B). The review system displays interactive chromatograms with colour-coded peak identification, MS spectra for each peak and allows additional annotation by the user via the following fields:

- *Compound confirmed* - was the expected molecular weight present in the MS spectra?
- *Interpreted purity* - allows the user to overwrite the purity if the reported purity is incorrect
- *Mass found* - the molecular weight if the substance detected was not the expected compound
- *Comments* - additional annotation such as “no peaks of interest” or “instrument error”

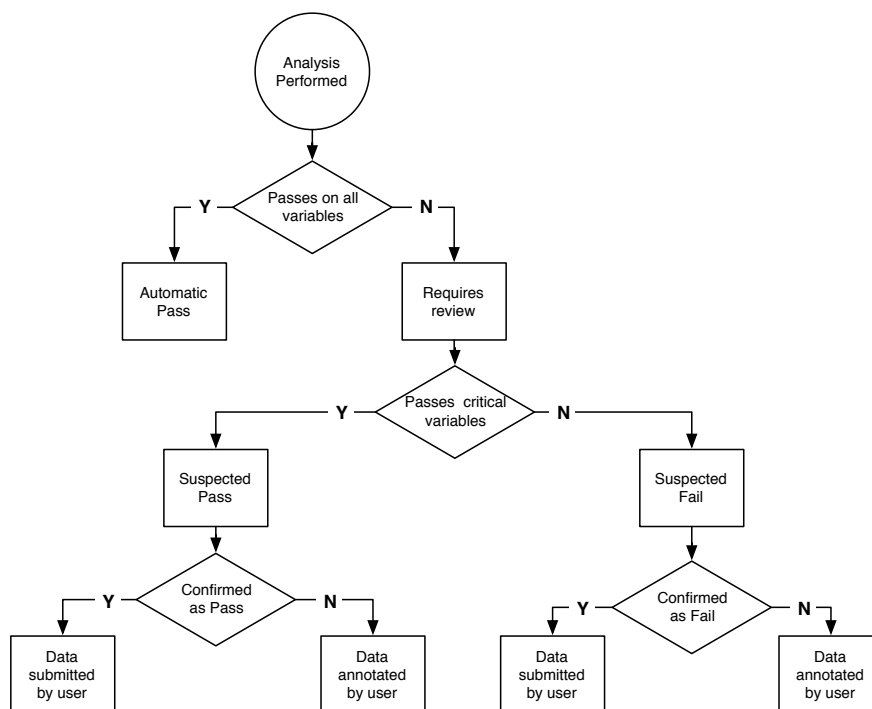


Figure 2.4: LC-MS manual review workflow adapted from Charles et al 2013 [CSA]

By entering values into these fields, the manual review process allows the user to either confirm that the analysis has passed or failed in agreement with the classification suggested by the system (confirmed pass/confirmed fail) or overwrite the suggested classification. It also allows the user to mark the analysis as “no peaks of interest” when a substance has been detected but does not match the expected compound, or mark the analysis as unsuccessful. The review workflow and possible outcomes are summarised in Figure 2.4. Examples of possible outcomes of the review process are provided below.

#### 2.4.4.1 Possible Outcome of the Review Process for LC-MS Data

A number of common outcomes from the review process are:

1. Analysis passes UV purity/spectral purity but fails another criteria. The system suggests a *pass* label for this analysis but must be confirmed by the user once other features of the results data have been examined. The *pass* status is confirmed via the review GUI.

2. Analysis fails either UV purity or spectral purity. The user can confirm the suggested *fail* status or overwrite to *pass* if the data reveals that for example, the structure appears to be undetectable by UV absorbance. In this case a purity value is entered in the *interpreted purity* field.
3. Sample structure incorrect. The resultant data may show a single substance detected via LC-MS but one which is not the expected compound. This may be due to errors by supply or process-related errors introduced during solubilisation activities. In this case, the user enters the molecular weight of the compound detected in the *mass found* field.
4. Chemical degradation or fragmentation. Some compounds may be unstable in DMSO and may degrade into a number of chemical fragments, shown in the LC-MS results as a number of integrated peaks which ‘share’ the UV absorption, with all peaks having low percent purity by UV.

This system offers a number of advantages. The extraction and standardisation of data means that the data storage and processing is independent of LC-MS instrumentation and so is future-proofed against the inevitable replacement or upgrade of hardware that is likely to occur over the long storage period of the compound collection. As the database is structured around individual analyses rather than plates (as is the case with most commercially available software) and stores a reference to the specific source vessels that created each analysis, it is possible to compare temporal trends in chemical stability of compounds stored as DMSO solutions. It also means that any historical data can be easily re-visualised, and ad-hoc clustering of sample results can be performed to highlight either structure- or process-related trends in quality.

The main advantage of this system, however, is the ‘peak of interest’ function and the ability to automatically pass an analysis, meaning that only those analyses which fail the automatic pass criteria need to be reviewed manually. The system has so far been used to process over 450,000 analyses. Of these, 147,195 (33%) have been flagged as automatically passed and have therefore not required a manual review. This is a considerable reduction in effort to process LC-MS data. However, more than 295,000 analyses still required reviewing manually by CM staff. This presents an opportunity to use a portion of the large dataset already generated as a training set, in order to build learning models for classification.

## 2.5 Applications of Data Mining in Drug Discovery

This section provides background information on the application of data mining approaches to various aspects of drug discovery. The use of reference databases is also discussed.

Analytical techniques have had to keep pace with the increasing speed and high-throughput environment of drug discovery research and therefore require high levels of automation and integration to be successful [Süß99], [KCW01]. As these advances in analytical instrumentation have developed, so too have the requirements for computerised systems for post-analysis data management and interpretation. Chen et al 2007 note that “with thousands of spectra produced everyday, an effective instrument data management system is essential to handle daily data storage and archiving, and organise the data for easy access by other researchers” [CPL07]. This view is supported by Sumuth et al 1999 who suggest that meeting the high-throughput demands is only conceivable by using “software-supported spectra interpretation” [Süß99]. One area of research that is of growing importance to the interpretation of analytical data is the use of data mining.

### 2.5.1 Decision Tree Induction to Support Drug Discovery Processes

A popular method for classification of LC-MS results is *decision tree induction*. The basic process involves recursive partitioning of data into more homogenous subsets by passing each record through a series of *if-then* rules arranged in a tree structure [MT03]. A detailed description of various data mining algorithms, including decision trees (DT) is included in Chapter 3. Applications of DT algorithms are increasingly prevalent in pharmaceutical research and healthcare more widely. Structure-activity relationships, hit identifications and disease diagnosis can all be facilitated by DT algorithms. In the absence of recognised biomarkers, DT classifiers have been used to help diagnose gastric cancer [SSQ<sup>+</sup>07] and premalignant pancreatic cancer [GW08]. Analytical techniques such as LC-MS produce rich datasets to which classification models can be applied, particularly when the outcome of analysis is a binary decision such as compound identity pass or fail, or a disease versus normal result. In many cases, however, such a binary decision is an unrealistic oversimplification of reality [AMBC04]. Instead, what is sometimes required is classification across a range of outcomes, with varying degrees of confidence. There may be situations where the outcome of structure identification is “beyond reasonable doubt”, “passed with 90%

confidence”, “suspected pass but requires confirmation” and so on. Fuzzy classification provides a formalisation of this concept and has been applied to MS data. For example, McJunkin et al 2006 [MS06] developed a fuzzy logic inference engine which allowed an operator’s attention to be drawn to questionable Fourier Transform MS data of mineral phases of basalt. This method allowed volumes of data to be processed of 3,600 spectra per hour that would otherwise have required an operator to manually review.

### 2.5.1.1 DT Induction in ‘Omics Cascade

Other examples exist in the literature of the application of DTs to the interpretation of LC-MS data. Areas where there have been significant effort are the so called “omics cascade”; genomics, transcriptomics, proteomics and metabolomics [DAa07]. Proteomics is the study of proteins within cells, and includes identification, modification, quantification and localisation [YR09]. Successful application of MS to proteomics is made possible by developments in soft ionisation techniques such as matrix-assisted laser desorption ionisation (MALDI) and electrospray ionisation (ESI) [YI00], [YR09]. Soft ionisation techniques allow proteins and peptides to be ionised and therefore analysed without extensive degradation. MS is now the most versatile and comprehensive tool in large-scale proteomics and its prevalence has driven a number of developments in the use of both machine learning and protein identification by database searching. Considerable effort has been dedicated to developing software to reduce the time taken for post-analysis data interpretation and reporting [DVMWK99].

### 2.5.1.2 Reference Databases to Support Data Mining in ‘Omics Cascade

The importance of database searching of MS data to identify proteins or peptides is another focus of recent research. Yates et al 1998 [YI98] found that when a protein is digested with a specific protease, the resulting collections of peptides, referred to as the mass fingerprint, can be determined using MS techniques and the results quickly identified using database searches. Several classification models have been developed to filter out low quality experimental spectra before attempting to match them in a theoretical spectra database such as SEQUEST [BGMY04], [Sun04]. Salmi et al 2006 [SMF<sup>+</sup>06] generated a C4.5 decision tree and Random Forest decision tree using WEKA [Wa05] to classify MS/MS spectra as good or bad in terms of the likelihood that a successful protein identification could be made. Sun et al 2004 [Sun04] developed functions based on two rules of mass spectra validation (fragment ion peaks



are clearly above the baseline and there is some continuity to the *b* or *y* ion series). Most noisy spectra, false interpretations and about half of poor fragmentation false positives were filtered out at low cost to true positives.

As the size of sequence databases has grown, so too has the speed and success of experiments to use MS analysis to understand protein functions. Gygi et al 1999 [GHGa99] searched for MS spectra in a composite database of known protein sequences (OWL) and a database of expressed sequence tags (EST), which contained approximately 250,000 proteins and 1.7 million small sequences respectively. By searching against both these databases, protein identification was often achieved with no human interpretation of MS data [GHGa99]. An algorithm for bacterial identification using MALDI-MS with automated data extraction and analysis has been developed [JCSa00]. This study reported a 90% correct identification rate, suggesting that their automated approach could potentially be used for accurate and reliable bacterial identification.

ProSight PTM is an integrated web-based software and database suite constructed for large scale proteomics [TKF<sup>+</sup>03]. The system has four main components; a database retrieval algorithm, a protein database, a file/data manager and a project tracker. Probability-based identifications using database searches were demonstrated. Other databases have been designed for storing MS results. The developers of the SpecDB system noted that MS data contains a large amount of information which makes it difficult to manage as a flat text file. Rather than storing a path to the physical location of the MS data, a loading interface was developed to extract data from text files, provide functionality for preprocessing/standardisation of spectral data and load it to the database. This process of data extraction and database storage is similar to the LC-MS system developed by CM UK, AstraZeneca and described in Section 2.4.3. As with the CM system, the approach used with SpecDB allowed a number of additional features such as historical data on spectra manipulation to add a temporal dimension to the database. It also featured the ability to prepare spectral data files for different kinds of machine learning tools such as WEKA [CV05].

### 2.5.1.3 Examples of Data Mining Approaches to Cancer Diagnosis

One area of research where machine learning algorithms have been successfully developed is cancer diagnosis from MS analysis. Early detection remains the most effective way to reduce mortality and pattern analysis of MS spectra of blood samples is one way this has been achieved. Biomarkers present at particular concentrations in blood

samples, such as the prostate specific antigen (PSA), show different patterns in cancerous and healthy samples, which can be perceived in MS data by machine learning algorithms [Sa06]. The study contributed to the ongoing effort to validate cancer detection through protein expression profiling using surface-enhanced laser desorption ionization (SELDI) and TOF-MS detection [Sa06]. Indeed, such data interpretation without the use of machine learning approaches would be far more time-consuming and labour intensive, and rely on human expertise [Sa06].

Markey et al 2003 [MT03] used a type of DT called *classification and regression tree model* (CART) on MS data to diagnose lung cancer. They showed that this classification model could be constructed by placing normalised peak widths within specific *mass-to-charge* ratio bins. The performance of the CART model was assessed using *Receiver operating characteristic* (ROC) analysis. This method allowed the trade-off between sensitivity and selectivity of the classifier to be evaluated [MT03].

Other studies have examined the effectiveness of *ensemble* learning models, which involve using a number of classification models in combination. These techniques are less prone to problems associated with overfitting as the resulting models are less biased towards a particular training dataset. Wagner et al 2003 [WAF<sup>+</sup>03] considered ensemble techniques with the aim of diagnosing ovarian cancer. A systematic comparison of various methods was made, the goal being to discover if it was possible to predict cancer on the basis of peptide/protein intensities revealed by MALDI ionisation coupled with Time of Flight (TOF) MS detection. The statistical methods examined were *linear discriminant analysis*, *quadratic discriminant analysis*, *k-nearest neighbour classifier*, *bagging* and *boosting* classification trees, *SVM*, and *Random Forests* (RF) [WAF<sup>+</sup>03]. RF methods outperformed all others in the study, which also raised the issues of data pre-processing, noise reduction and variable selection as additional challenges faced when making diagnoses using MS data.

*Support vector machine* (SVM) classification combines linear modelling and instance-based learning. SVM has several features that make it suitable to the classification of LC-MS data, such as its robustness to redundant features and therefore lower importance on feature selection [Sa06]. Burbidge et al 2001 [BTBa01] applied a SVM to structure-activity relationship data and concluded that SVM was an “automated and efficient deterministic learning algorithm” capable of highly accurate classification (error rate of 0.1269). The SVM algorithm outperformed other techniques such as the C5.0 decision tree algorithm, *radial based function networks* and *multi-layer perceptrons* (Neural Networks).

This chapter has provided an overview of the various analytical methods utilised by CM UK, AstraZeneca for quality assurance of the compound collection. It has also described how the instruments have been incorporated into an information management system, which offers some degree of automatic interpretation of analytical data and presents an opportunity to further automate data processing. Chapter 3 provides an overview of the Knowledge Discovery from Data (KDD) process used in this project and describes in detail the various data mining algorithms that have been evaluated. The data mining approaches discussed in this chapter will be investigated in terms of the ability to generate a classification model for accurately predicting the pass/fail outcome of LC-MS analyses. A number of measures will be used to evaluate the models, which will be primarily generated using the WEKA software suite (version 3.6.8), with further experiments conducted using Pipeline Pilot (version 8.5.0.200) and JMP version 10 (SAS, SAS Campus Drive, Building T, Cary, NC 27513, USA).

# Chapter 3

## Research Methods

This chapter describes the KDD process and various features of the data mining algorithms assessed in this project. Information on how the algorithms were evaluated is provided, along with details of techniques used to improve the performance of classification models, such as *cross-validation*, *decision tree pruning*, *ensemble* approaches and *cost-sensitivity*. The format of the equations in this chapter are taken from the book *Data Mining, Practical Machine Learning Tools and Techniques*, Witten et al (2005) [WF05].

### 3.1 Overview of Knowledge Discovery from Data (KDD)

Many industries are characterised by a proliferation of data, meaning that large databases are now ubiquitous. The highly automated, high throughput environments of modern drug discovery activities are certainly no exception, with huge amounts of analytical data generated on a daily basis. The low cost of data storage and the digitisation of data generated from every step of the Design-Make-Test-Analyse (DMTA) cycle has meant that the generation and storage of data is no longer a limitation. What remains a challenge is the conversion of all this data into knowledge, in the sense of extracting useful patterns and structure from data that will allow knowledge-based or expert systems to be developed. Historically, the extraction of such expertise has been achieved through interview style questioning of domain experts. This is problematic for a number of reasons; it is prone to bias, assumptions and preconceptions which are possibly inaccurate and can be difficult to extract successfully. Even with experience and expertise in a domain, the widespread use of databases and the sheer volume of digital information have rendered traditional methods of manual analysis and interpretation

of data increasingly difficult, even unrealistic.

The field of Knowledge Discovery from Data (KDD) addresses the problem of a data-rich, information-poor situation by applying a number of steps to elucidate meaningful patterns from examples within the data. A definition of KDD is provided by Fayyad et al 1996 [FPSS96]:

*The non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*

The steps involved in KDD can be summarised as data preprocessing, data mining, pattern evaluation and knowledge representation. An overview of the steps in the KDD process is provided in Figure 3.1 (adapted from Fayyad et al 1996 [FPSS96]). They are discussed in more detail later in this chapter.

The essence of data mining can be described as ‘mining’ through large amounts of data to discover ‘nuggets’ of knowledge, in the sense that these pieces of information can be incorporated into future analysis so that the characteristics of unseen data can be better understood [WFT<sup>+</sup>99]. Learning can be said to have occurred when useful patterns are discovered from the data that increase the predictive performance of understanding new unseen data.

### 3.1.1 Supervised and Unsupervised Approaches to Data Mining

Data mining techniques are described in terms of *concepts*, *instances* and *attributes*. The *concept description* is the result of the learning process and should be intelligible and operational so that it can be understood and applied to future data. Instances are the examples or tuples from the data that allow knowledge to be learned and are encoded as a vector of attribute values. Typically, either numeric attributes or nominal/categorical attributes are used by data mining schemes although other attribute types can also be included, such as relational attributes. It is important to realise that successfully extracting the most meaningful patterns from data is dependent on the attributes that are encoded in the data instances. Clearly, data mining can only operate on the attributes available therefore efforts should be made to capture all relevant information that contributes to data characteristics so that important features are not omitted.

Data mining schemes can be broadly categorised as one of the following types:

- *Classification learning*: This is a two stage process which involves the model being constructed using a training set from the subset of available data. This is

followed by a second stage where the performance of the model is tested using a test set, independent of the training data. The learning scheme is presented with the training instances, which are labelled with a class attribute. It is the value of this class that the model tries to predict, based on the values of the other attributes observed in the training data. The term *supervised* is sometimes used to denote that the model is under supervision by being provided the class outcome of the training examples. The class values of the test set are known but not made available to the classification model. The performance of the classification model can be evaluated by comparing the predicted class outcome from the test set with the actual outcomes. The success rate of classifying test data can be used to provide an objective measure of how well the model performed. However in practical terms there may be other considerations, for instance whether the learned description is intelligible to a human user or the degree to which learning is generalised to the wider population of real world data. More details on the evaluation techniques used to assess model performance are discussed in Section 3.4.

- *Association learning*: Any association among features is sought, not just ones that predict a particular class value. Association learning has the freedom to consider attributes in combination to allow predictions to be made. Unlike classification rules, association rules are not intended to be used as a set but rather can be considered individually to predict different features of the data. Generally, even small data sets will yield a large number of association rules, some of which may be due to randomly occurring patterns in the data that offer little useful knowledge or learning. To counter this, thresholds can be applied to the rules to try to focus on the more useful ones. Two common methods for achieving this are the *support* and *confidence* of a rule. The support measures the usefulness of a rule and is defined as follows:

For a dataset  $D$ ,  $supp(A \Rightarrow B)$  is defined as the proportion of  $D$  that contain  $A \cup B$

The confidence of a rule is a measure of the certainty of a rule and is defined as  $conf(A \Rightarrow B) = supp(A \cup B) / supp(A)$

By setting thresholds on support and confidence, weaker or poorly performing rules can be excluded from further consideration.

Other types of data mining include *clustering* where instances that show similarity

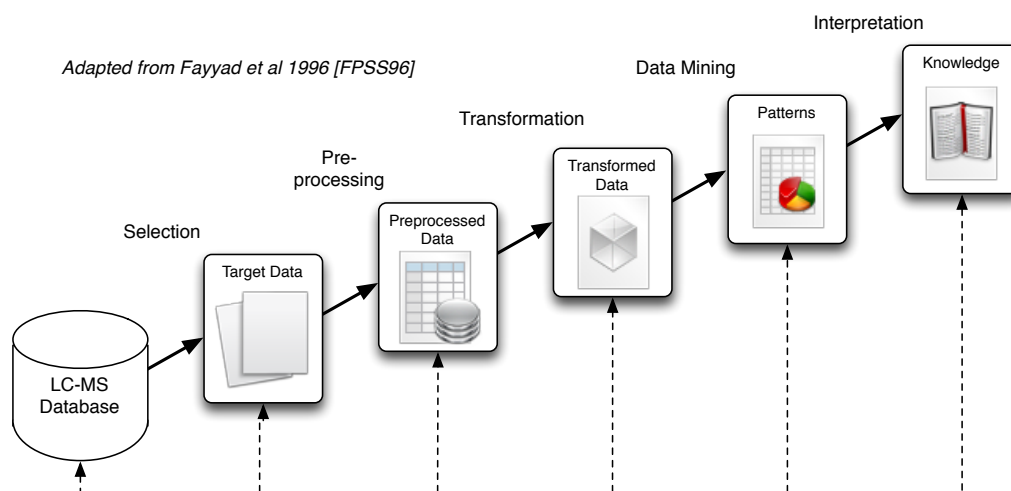


Figure 3.1: Overview of KDD process

with regard to some aspect of their structure are grouped together and *numeric prediction* where the outcome is not a discrete class but rather a numeric quantity.

In this dissertation, the concept under investigation is the outcome of LC-MS analysis of DMSO solutions to confirm the purity and chemical identity of the samples analysed. An instance is the analytical data generated by the instrument following an analysis. More specifically, an instance is a subset of the attributes generated for a single integrated peak from an analysis (See Section 3.2.3 on data preprocessing). The main focus of the experimental work undertaken was an empirical comparison of the performance of various supervised classification models. The aim was to identify a classification model capable of making highly accurate predictions of the outcome of future LC-MS analyses.

## 3.2 The KDD Process

The following section firstly provides a brief description of each of the KDD steps, followed by a more detailed description of how some of the steps apply specifically to the subject of this project, structural identification and purity confirmation of screening compounds by LC-MS.

1. *Learning the application domain.* Compound management staff are experts in the domain of quality assurance of compounds via LC-MS analysis and were

able to provide valuable input into the KDD process, for example by providing information on the expected importance of the available variables recorded by the instrumentation. They were also able to provide insight into what additional calculated attributes might make a useful contribution to the construction of classification models (see Section 3.2.3.6).

2. *Identify target dataset.* The analytical LC-MS database was used to provide a dataset of >450,000 analyses (a total of >1 million integrated peak records). The operational LC-MS database was copied to create an experimental database to provide training/test data for developing classification models (PostgreSQL 8.3.5).
3. *Data cleaning and pre-processing.* Data were extracted from the database in the ARFF format using a simple Java application and analysed using the WEKA software suite (version 3.6.8), Pipeline Pilot (version 8.5.0.200) and JMP (version 10). Compound/sample/supplier identifiers used within AstraZeneca were replaced with arbitrary identifiers to protect AstraZeneca's intellectual property.
4. *Data reduction/projection.* The LC-MS database contains many meta-attributes that relate to the compound being analysed but not the analysis itself. These include age of solution, supplier and the instrument on which the analysis was performed. Only data from the actual analysis was considered when evaluating the supervised classification models, as the aim was to perform a classification based on the outcome of the LC-MS analysis before any manual review by a domain expert.
5. *Choosing the function/purpose of data mining.* The functions required in this study were classification of LC-MS analysis results into *pass* and *fail*.
6. *Algorithm selection.* For the classification modelling, a number of algorithms were evaluated; *decision trees*, *support vector machines*, *neural networks* and *ensemble learning* methods. These were chosen as a diverse set covering most of the algorithms typically applied to this type of classification problem.
7. *Data mining.* This stage involves the actual application of the algorithms to search for patterns in the data. This involved the evaluation of various classification models via WEKA, with some additional experiments using JMP and Pipeline Pilot.



8. *Interpretation.* Interpretation of the discovered patterns was undertaken including discussions with domain experts, such as evaluation of false positives and false negative using metrics such as precision and recall calculations. Classification models were optimised using different methods of descritisation of numerical attributes, feature selection and pruning of decision trees to avoid problems with overfitting. This led to the re-evaluation of learning algorithms in an iterative manner which characterises the KDD process.
9. *Application of discovered knowledge.* High performing classification models were identified. A discussion on how these models could be incorporated into the current LC-MS workflow is included in Chapter 5.

### 3.2.1 Class Labels

The data generated from the LC-MS analysis of a sample is actually hierarchical in structure, as described in Section 2.4.3. Some attributes are at the top level (sample record level) such as target molecular mass and all manual annotations (interpreted purity, mass found etc). Other attributes are at the lower level (peak record level), such as spectral purity by MS. The process of manually reviewing the results of an analysis involves looking at the set of all peaks integrated by the instrument. Only by considering the full set can some problems be detected. For example, it is assumed that the peak of interest as determined by the database function described in Section 2.4.3 is correct. This however, is not always the case and on occasion a visual inspection of the other peaks, plus expert knowledge applied to a visualisation of the structure of the molecule can reveal that the system has in fact identified the wrong peak as the peak of interest. In such situations, the initial decision of fail can be overwritten by the user to pass. This presents a serious problem; the information that led to this reclassification is not encoded in the attributes of either the sample or the peaks, so for an analysis where the original outcome of fail was updated to a pass, it is not possible to establish which of the integrated peaks accounted for the result to be overridden. Instead the approach was to develop a classification model that could predict with high probability that each integrated peak was either a pass or a fail, leaving only those analyses with more questionable results to be reviewed manually. The decision was taken to recast the hierarchical data into a de-normalised set of independent peak instances and build a classification model at this lower level. It would then require a simple task to aggregate predictions on all peaks back up to the sample level by examining if one of the peak

Class label 1	Class label 2	Class label 3
Pass	Auto pass	auto pass
	Agreed pass	ms pass uv pass agreed pass
	Overwritten pass	ms pass uv fail overwritten pass
		ms fail uv pass overwritten pass
ms fail uv fail overwritten pass		
Fail	Agreed fail	ms pass uv fail agreed fail
		ms fail uv pass agreed fail
		ms fail uv fail agreed fail
		diff mass agreed fail
		no peaks agreed fail
		no spectra agreed fail
		no peaks-of-interest agreed fail
	Overwritten fail	ms pass uv pass overwritten fail
		diff mass overwritten fail

Figure 3.2: Class labels applied to LC-MS integrated peaks

instances from a sample is predicted as a pass or if all the peaks are predicted to be fails with a high confidence.

### 3.2.1.1 Binary and Multiple Outcome Class Labels

Three functions were written to the database to apply binary and multiple outcome class attributes to each peak. These class attributes provided increasing levels of details about the pass/fail outcome for the peak and how that outcome was reached. The first class label was a binary classifier with possible outcomes of *pass* or *fail*. The second split these two categories into a more detailed class label, with possible values of *auto pass*, *agreed pass*, *overwritten pass*, *agreed fail* and *overwritten fail*. The third class label divided the possible values of the second level into even more detailed outcomes, such as *ms fail uv fail overwritten pass* or *ms pass uv fail agreed fail*. The relationship between the three class labels is described in Figure 3.2.

## 3.2.2 Creation of Data Sets

Although the overall outcome for a peak was a binary decision, the arrival at that decision could be reached in different ways, as discussed above. A number of training and test datasets of increasing size were stratified so that the proportions of these various outcomes of analysis (class label 3) were consistent with the proportions in the database as a whole. This stratification was performed to avoid the models having

a bias towards a particular outcome. Furthermore, the datasets were partitioned into training/test sets (2/3 training and 1/3 test) to ensure that this stratification was maintained in both subsets. As the size of the datasets increased the computational cost and build time for the models also increased, but so too did the amount of available data on which to build the models. Initial experiments were performed on the smaller datasets and final assessments on the largest to maximise amount of data available and therefore the performance of the models. Table 3.2.2 lists the datasets generated and the number of peak records in the training/test partitions.

Table 3.1: Stratified Training and Test Data Partitions

Data set	Instances in training set	Instances in test set
Set_1k	1,000	501
Set_100k	99,973	49,989
Set_200k	199,945	99,973
Set_300k	299,919	149,962
Set_400k	399,891	199,946
Set_500k	499,865	249,936
Set_600k	599,836	299,920
Set_700k	699,809	349,908
Set_800k	799,783	399,893
Set_Max	1,546,262	773,176

### 3.2.3 Data Preprocessing and Transformation

Before being used to train and test classification models, a number of data preprocessing and transformation methods were used to remove noise, outliers, missing or invalid data. Such data present in training/test datasets could be translated into a loss in quality of even the most sophisticated data mining techniques.

#### 3.2.3.1 Data Cleaning

The data stored in the LC-MS database was extracted from output files generated by the analysis instrumentation. The extraction of data from these files and subsequent insertion into the LC-MS database was performed using a PERL script run via a Pipeline Pilot protocol. The script extracted pertinent data from the instrument output files and included validation steps that ensured the validity of data entering the database. Additional annotations were added by users for those records that required manually

reviewing. These annotations were entered via a web-based GUI that included client-side validation. This provided some guarantees about the validity of the user-entered data, however in early versions of the reviewing software it was possible for a user to enter the values for molecular weight and purity in the wrong fields. To check for such errors, the database was updated accordingly when the molecular weight was lower than 100 (only very few samples from a fragments library could actually have a molecular weight  $<100$ ) or when the percentage purity exceeded 100. More recent versions of the software GUI included validation to alert the user when such errors occurred.

A total of 2,389 analyses were marked by users as instrument errors during manual review. These records were filtered out from all experimental data as the data produced was spurious and may pollute data mining algorithms if included. Different LC-MS instrumentation used by Compound Management recorded UV peak height, UV peak width, CAD peak height and CAD peak area using different units. A multiplier was used to normalise these values across the different instruments.

### 3.2.3.2 Missing Values

Data in the LC-MS database was generated by analytical instrumentation with subtly different configurations. All mass spectrometry detectors used electrospray ionisation but some instruments also include an atmospheric pressure ionisation system. These missing values were assigned the value of 0. This had little effect on the data overall, as the spectral purity by MS values for individual ionisation techniques and ion modes were not considered directly. Instead, a function was used to find the maximum spectral purity value across each ionisation method and ion mode. This function is described below in Section 3.2.3.5. The likelihood of a sample being detected may have been slightly less if processed on an instrument with only ESI ionisation but this was not considered significant.

Null values for MS spectral purity or percent purity by UV were considered as valid results indicating that the target mass or any of the adducts included in the search pattern had not been detected. As such, null values were also assigned a zero value.

### 3.2.3.3 Instrument-generated Attributes

As the objective was to generate classification models that will avoid the need for manual review, the models had to exclude any user-entered attributes. Rather, the attributes used to construct the models were exclusively those either generated directly by the

instrumentation or derived from instrument data. Domain experts from the Quality Assurance group within CM were consulted to establish which attributes should be made available to the data mining algorithms. The available attributes are listed below (an explanation of each is provided in Table 2.4.2).

- *Percent purity by UV*
- *UV peak height*
- *UV peak width*
- *Retention time*
- *CAD peak area*
- *CAD peak width*
- *CAD start time*
- *CAD stop time*
- *DAD start time*
- *DAD stop time*

#### 3.2.3.4 User-generated Attributes

These attributes allowed data to be annotated during the manual review stage. Although these attributes were not included as features for the classification models, they were required in order to generate the class labels discussed in Section 3.2.1.

- *Compound confirmed* (yes/no) - was the expected molecular weight present in the MS spectra?
- *Interpreted UV purity* (0-100%) - allows the user to overwrite the UV purity.
- *Mass found* (molecular weight) - molecular weight of the structure detected if it is not the expected molecular weight.
- *Comments* (no peaks/no spectra/no peaks of interest/instrument error) - additional annotation to characterise the analytical results.

#### 3.2.3.5 Derived Attributes

The current LC-MS system uses a number of derived attributes to allow the results of analyses to be processed more efficiently.

- *peak\_of\_interest* - The peak of interest was discussed in more detail in Section 2.4.3 but is listed here again for completion. One peak from each sample analysed is assigned as the peak of interest. This is the peak that is most likely to allow an analysis to be passed and is presented first by default in the review system.
- *horz\_max\_ms\_purity* - This function finds the maximum spectral purity for each peak from the one or two (depending on which instrument was used) ionisation techniques employed by the MS detector. Each has two ion modes, giving a total of either two or four values. Whether a compound will be detected in the positive or negative ion modes of each ionisation method is a function of the chemical properties of the molecule and is not discussed in this project.

### 3.2.3.6 Additional Derived Attributes

Additional attributes were computed that were derived from the set of peaks integrated in a given sample. These were included following discussions with domain experts, to investigate whether the additional information could contribute to the classification models. The values were determined using functions on the database.

- *get\_closes\_peak* - This calculated the minimum difference in retention time between a peak and the previous or next peaks to be detected by DAD chromatography. This was calculated to examine the errors with peak integration, such as when two partially overlapping peaks are integrated as a single peak or vice versa.
- *num\_of\_peaks* - This value simply holds for each peak the total number of peaks eluted in the DAD chromatogram for the given sample. Ideally, a single peak would indicate maximum purity whereas a large number of peaks could indicate chemical degradation or contamination.
- *get\_cad\_percent* - The instruments output a value for the CAD peak area which represents the area accounted for by each integrated peak in the CAD chromatogram. These values were recorded in arbitrary units (standardised across different instruments). This function converted these values into a percentage of the total area under the CAD chromatogram.

### 3.2.4 Attribute Selection

Careful consideration was given to the selection of attributes to be included in the training/test data sets. Domain experts were consulted during the attribute selection process to ensure that the right balance was achieved between including attributes that may have made a useful contribution to classification whilst minimising the number of attributes to avoid building low-value complexity into the models (discussed further in Section 3.3.3).

The domain experts were experienced LC-MS technicians with a detailed knowledge of interpreting data from LC-MS analysis and in-depth knowledge of the chemical properties of small molecule research compounds. These discussions led to the following set of attributes being included in the training/test data sets:

1. percent\_purity\_uv
2. horz\_max\_ms
3. retention\_time
4. uv\_peak\_height
5. uv\_peak\_width
6. cad\_peak\_area
7. cad\_peak\_width
8. get\_cad\_percent
9. get\_closest\_peak
10. num\_of\_peaks
11. classifier

### 3.2.5 Descritisation of Numerical Attributes

Almost all the attributes extracted from LC-MS analytical data are numerical. Unlike categorical attributes which are only used in one splitting node as the information provided has then been used, numeric attributes can contribute to a number of nodes throughout the path from the root node to a leaf. This is advantageous but can lead to “messy” decision trees with a high number of nodes that require pruning.

This presents particular issues around descritisation of numerical data and there are a number of approaches to the process of descritisation; *unsupervised descritisation*, *entropy-based descritisation* and *error-based descritisation*.

*Unsupervised descritisation* quantises instances without considering their class allocation. Equal-width binning simply divides the range of numerical values into a predetermined number of bins. This method can often result in bins with varying numbers of instances and so can impair the resultant decision model. Equal-frequency binning addresses this issue but still creates intervals without considering the boundaries between the occurrences of different classes.

*Entropy-based descritisation* uses an entropy calculation in a similar way as selection of splitting attributes in the nodes of divide-and-conquer decision trees (Section 3.3.2.1). Information gain is calculated for each possible position and the split that results in the purest subdivisions is selected. The process is then repeated until a stopping criteria is satisfied, such as the *Minimum Description Length principle* (MDL) which states that “the best theory is the one that minimises the number of bits required to communicate the theory, along with the labels of the examples from which it was made” [WFT<sup>+</sup>99].

*Error-based descritisation* divides a number of instances,  $N$ , into a number of subdivisions,  $k$ , that minimises the number of classification errors in time linear in  $N$ . This means that descritisation can occur far more quickly than compared to entropy-based descritisation but is problematic due to the fact that the algorithm cannot produce adjacent intervals with the same label [WFT<sup>+</sup>99].

### 3.2.6 Cross-validation

Initial assessments were carried out using a subset of the available data. Cross-validation was used for the initial assessment of learning algorithms and for the experiments on pruning methods as smaller training sets are more likely to be biased by particular features of the data and therefore suffer from overfitting. The standard *10-fold cross validation* was used, where data is divided randomly into 10 folds or partitions. One of the folds is used as the holdout test set with the remaining nine-tenths used as training data. The process is repeated with each of the ten folds used as the holdout test set. Finally an average error estimate is calculated from the 10 error estimates.

When larger datasets were used to assess model performance, cross-validation became less essential and more computationally expensive. The amount of data used for the later assessment of classification models and for the experiments on cost-sensitivity contained approximately 500,000 instances. Due to memory constraints, cross-validation was not applied as maximising the size of data sets was preferred to emulating large dataset by applying cross-validation.



### 3.3 Learning Algorithms

The *IR* method is a simple classification model that acts on a single attribute. A set of rules is established for each attribute to measure how well it performs in terms of correctly predicting the majority class. The attribute with the best performance on training data is selected. Despite the simplicity of this approach, Holte et al 1993 found that *IR* performed only a few percentage points less than far more sophisticated decision tree induction schemes that were considered state-of-the-art at the time of publishing [Hol93].

Another simple classification algorithm is the *Naïve Bayes* technique which considers the full set of attributes as evidence for the probability of the class outcome. All attributes are considered to be equally important and independent of each other with regard to the class. Neither of these assumptions are generally true in real-world situations but despite this, *Naïve Bayes* has been used to yield accurate classification models, outperforming DTs, SVMs and NNs at predicting user profiling data for service provisioning applications [CLM08]. One vulnerability of *Naïve Bayes* is a situation where a value of a particular attribute does not appear in the training data for every class outcome. This can lead to a situation where the probability of one value of an attribute is zero and as the probabilities from other attributes are multiplied by this zero probability, the overall probability is also zero. This situation is mitigated by using the *Laplace estimator* where an amount for example 1, can be added to each nominator and the denominator adjusted according to the number of possible values to which 1 has been added [Dea82].

#### 3.3.1 Probabilistic Classification

The outcome of an LC-MS analysis event is a binary decision and although a number of subtle distinctions can be made between different results, these can be summarised to be either a pass or fail, with the result predicted by the classifier being either correct or incorrect. This can be described as the *0-1 loss function*;

$$\sum_i \begin{cases} 0 & \text{if prediction is correct} \\ 1 & \text{if prediction is incorrect} \end{cases} \quad (3.1)$$

As has been stated previously, the aim of this application is to minimise the number of analyses that require manual review by a technician, by automatically classifying as many analyses as possible. To achieve this, a probabilistic classifier was required as

there is a clear difference between an analysis that has a 95% probability of being a pass and one with a 60% probability. The first would be marked as passed automatically, the second flagged as requiring manual review. Two methods used to assess the probabilities of correct classifications are the *quadratic loss function* (3.2) and the *informational loss function* (3.3). The former considers the entire probability vector for all possible outcomes whereas the latter considers only the probability associated with the correct class.

$$\sum_j (p_j - a_j)^2 = 1 - 2p_i + \sum_j p_j^2 \quad (3.2)$$

$$-\log_2 p_i \quad (3.3)$$

### 3.3.2 Decision Trees

Decision Trees (DTs) are a popular method of data mining due to the ease with which they can be generated, the potential to generate powerful predictors and the fact that they produce explicit concept descriptions that are easy to interpret [Shi07a]. The basic process involves recursive partitioning of data into more homogenous subsets by passing each record through a series of *if-then* rules arranged in a tree structure [MT03]. Data passes through each node until finally reaching a leaf, where all the records are of the same class or alternatively where a particular purity threshold has been reached. Each attribute of the data is considered in turn and ranked according to its ability to partition the data such as in the C4.5 algorithm described by Quinlan et al 1986 [Qui86].

Figure 3.3 shows an example of a J48 DT (the WEKA implementation of C4.5). Each possible route from the root node down to a leaf can be easily converted into a set of rules. For this reason, DTs are considered to be easily interpretable. For example, the route highlighted as a blue dashed line would be converted into set of tests shown in Figure 3.3.2.

This includes every internal node from the root to the leaf. As numerical attributes can be used to split at different levels of the tree, this rule can be simplified further as shown in Figure 3.3.2.

By converting all routes to leaf nodes in this manner, DTs can easily be converted into a set of classification rules.

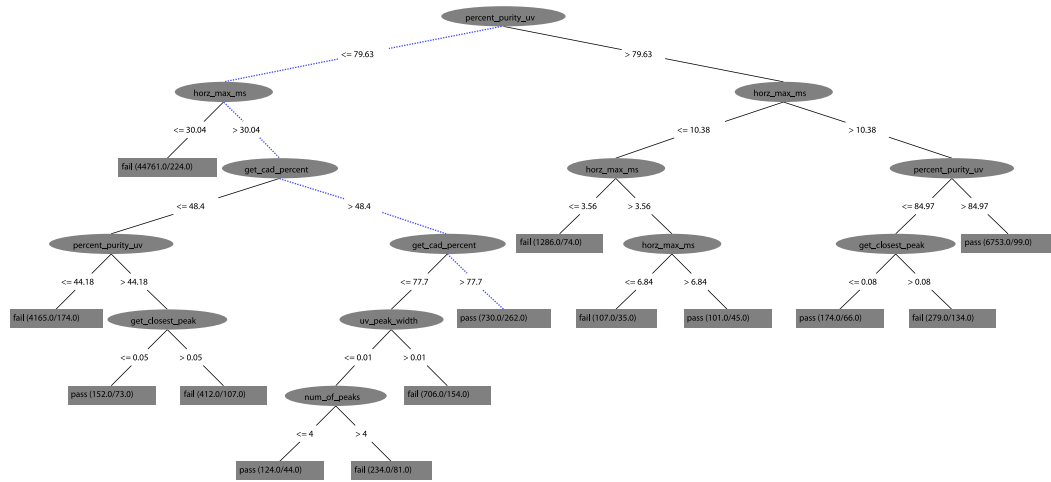


Figure 3.3: An example J48 Decision Tree showing splitting nodes (ellipses) and leaves (boxes)

If `percent_purity_uv <= 79.63` & `horz_max_ms > 30.04`  
 & `get_cad_percent > 48.4` & `get_cad_percent > 77.7`  
 then : pass

Figure 3.4: Path through DT represented as a set of rules

If `percent_purity_uv <= 79.63` & `horz_max_ms > 30.04`  
 & `get_cad_percent > 77.7`  
 then : pass

Figure 3.5: Path through DT simplified as a set of rules

Many extensions and developments have been made to early DT designs to improve performance such as modifications to the classic C4.5 algorithm by modifying the evaluation of continuous attributes, which leads to smaller and more accurate DTs [Qui96a]. Examples of common DT variations include the *ID3* (Iterative Dichotomiser) algorithm [JDIFx09], the *C4.5* algorithm [Qui86] and *Classification and Regression Trees* (CART) [BFOS84]. The features that are common to all these algorithms are the three parameters, a dataset  $D$ , an *attribute\_list* and an *attribute\_selection\_method*. DTs have a number of advantages in that they are easily interpretable and are not parametric but rather based on a series of tests, which can be translated into a set of production rules. DTs allow complex relationships between variables to be modelled and are capable of highlighting “important” variables [ZZ07]. They also require no a priori assumptions about the data, are capable of handling nominal and numerical data and are relatively quick to generate even on large datasets compared with SVMs and NNs. Small DTs tend to be more likely to generalise and avoid overfitting to training data. Methods to simplify DTs are discussed in Section 3.3.3.

### 3.3.2.1 Attribute Splitting Methods

Attribute splitting methods are heuristics that determine which attribute to split on at each internal non-leaf node in a DT. Examples of these methods are *information gain*, *gain ratio* and *Gini Index*.

### 3.3.2.2 Information Gain

Information Gain is implemented in the DT algorithm ID3 [Qui86]. The measure calculates the information gain for splitting on each candidate attribute and then selects the one that minimises the information required to classify the resulting partitions. In this way, the attribute that results in the purest or least random partitions is identified. The algorithm is not exhaustive and cannot guarantee that the optimal DT will be built but despite this, the use of information gain as an attribute selection method does generally yield concise DTs.

Firstly the *entropy* of a dataset  $D$  is calculated. This provides the probability that an arbitrary tuple in  $D$  belongs to the class  $C_i$  and is estimated by  $|C_{i,D}|/|D|$ . This estimate is based purely on the proportions of each class in  $D$  without considering any of the other attribute values. The entropy is given by:

$$Info(D) = - \sum_{i=1}^m p_i \log_2(p_i) \quad (3.4)$$

Each attribute is then considered for splitting. The amount of information still required to classify a tuple from  $D$  after splitting on attribute  $A$  is calculated as:

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j) \quad (3.5)$$

where  $\frac{|D_j|}{|D|} \times Info(D_j)$  is the weight of the  $j$ th partition. Finally, the information gain is calculated as the difference between the information required to classify a tuple from  $D$  before the split and the information required after splitting on attribute  $A$

$$Gain(A) = Info(D) - Info_A(D) \quad (3.6)$$

The attribute with the highest information gain is selected as the attribute to split on at the given node. The process is then repeated for subsequent internal nodes.

One weakness of this calculation is that it favours multi-valued attributes. The example often used to demonstrate this is the existence of a unique identifier attribute. In this case the information gain would choose this attribute to split on as the information required to classify tuples after this split would be zero as each tuple would occupy a pure, single instance leaf node.

### 3.3.2.3 Gain Ratio

An extension of the Information Gain calculation is Gain Ratio, as implemented by the DT algorithm C4.5 [Qui86]. This measure attempts to reduce the bias towards multi-value attributes from which information gain suffers by applying a normalisation to the information gain value. The *split information* value takes into consideration the number of instances in each partition with respect to the number of instances in  $D$ . It is calculated as follows:

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right) \quad (3.7)$$

The gain ratio is then calculated as:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)} \quad (3.8)$$

As the split information approaches 0 this equation becomes unstable so it is only applied when the information gain of a split is at least as much as the average gain over all attributes examined.

### 3.3.2.4 Gini Index

The Gini Index is a splitting criteria used in CART [OS84]. The calculation is similar to the information gain in that the reduction in impurity of a data partition  $D$  is calculated before and after splitting on attribute  $A$ . The initial Gini Index of data partition  $D$  is calculated as:

$$Gini(D) = 1 - \sum_{i=1}^m p_i^2 \quad (3.9)$$

In the case of categorical data, every possible binary split of the available values for attribute  $A$  is considered (excluding the power set and empty set as they do not provide a split). For each binary split of  $D$  into  $D_1$  and  $D_2$  the weighted sum of the impurity of each resulting partition is calculated as follows:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \quad (3.10)$$

For numeric attributes, binary splits are calculated for every possible split point, defined as the mid point between each pair of sorted adjacent values. In the case of both categorical and continuous attributes, the reduction in purity caused by splitting on attribute  $A$  is defined as:

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \quad (3.11)$$

The Gini Index for each binary split of all available attributes are then compared and the value that provides the maximum reduction in impurity of resultant data partitions is selected as the splitting attribute for the given node. The process continues in an iterative manner.

The process described has two problems as discussed by Loh et al (1997) [LS97]. The first stems from the fact that the search is exhaustive and is therefore potentially computationally expensive. For continuous attributes with  $n$  distinct values there are  $n-1$  potential splits and for categorical attributes the number of computations increases exponentially with each possible value, defined as  $(2^{M-1} - 1)$  for a variable with  $M$  values.

The second problem is one of bias. The Gini Index tends to favour attributes that have more splits. Quinlan et al (1995) [QCJ95] found that the chance of selecting ‘fluke’ theories that fit the data well but offer little in terms of predictive accuracy increases as does the number of hypotheses being explored.

### 3.3.2.5 Chi-squared Automatic Interaction Detector (CHAID)

CHAID is an extension of the *automatic interaction detector* method and was originally proposed by Kass 1980 [Kas80]. It operates on categorical predictors as a mechanism for partitioning data into mutually exclusive subsets that best describe the dependent variable [Kas80]. This is achieved by selecting a split that maximises the significance of a chi-squared test of independence. CHAID is not guaranteed to find the best split at each step but rather it searches heuristically to find a suitable split, avoiding the computational cost of searching all possible category subsets [Wil92].

### 3.3.3 Decision Tree Pruning

The aim when constructing classification models is to minimise the complexity of the model without significantly reducing performance. Simpler decision trees are more generalised and less prone to overfitting. They are also easier to interpret and transform into a set of rules. As all decision tree algorithms use the methods discussed for selecting the most suitable attribute to split on at each internal node, the inclusion of additional attributes might not seem problematic. However, whilst attributes that have little actual contribution to class outcomes will not be selected at the higher nodes in a decision tree, further down towards the bottom the amount of data on which attributes are selected for splitting is reduced. As a result, non-contributing attributes can by chance appear to be the preferred attribute on which to split. This can lower the performance of equivalent decision trees in which fewer attributes are included.

Even with a limited set of attributes, DTs often contain structure that accommodates anomalies or outliers in the training data. In order for DTs to perform more accurately on test data or unseen future data, a process of pruning is required to produce a more generalised form of the tree. Two main strategies are available to achieve this. *Prepruning* involves the decision not to continue dividing subsets of the training data during the construction of the tree. *Postpruning* involves removing subtrees from the overall decision tree after the full tree has been constructed.

### 3.3.3.1 Prepruning

Prepruning has an obvious advantage in terms of the computational cost of building the full tree. By stopping development of the tree during its construction, the cost of building subtrees which will ultimately be discarded can be saved. This is more advantageous if particular consideration is applied to the runtime taken to construct the tree. One problem with this strategy can occur when attributes have seemingly little value as splitting criteria in isolation but can be more powerful splitting criteria when considered in combinations.

The methods of assessing which attributes to split on, such as information gain, Gini Index and other tests for statistical significance, can be used to test for a threshold below which further splitting will not occur. This represents a trade off as too high a threshold will result in oversimplified trees that perform poorly and too low a threshold will provide insufficient simplification to avoid overfitting.

An example of this type of prepruning is presented by Kamber et al (1997) and is implemented in the MedGen system [KWGa97]. Two thresholds are introduced, an *exception threshold* and a *classification threshold*, both of which have the affect of ceasing further splitting. For any internal node, if the number of instances falls below the exception threshold, splitting is terminated and a leaf node is created that stores the subset and distribution of the subset. The classification threshold is applied by testing the percentage of each class at a given node. If the percentage of any class value exceeds the threshold splitting is terminated [KWGa97].

Another type of prepruning that was used in this project was to apply a minimum number of instances per leaf node. In a fully unpruned DT, splitting ceases when the data produced from a node is 100% pure with regards to the class label or when a split produces a single instance. By applying a minimum number of instances per leaf node, further splitting will not be applied even if the purity of the data partitioned is less than 100%. This method eliminates the lower branches of a DT and produces smaller, more generalised models.

### 3.3.3.2 Postpruning

Postpruning is achieved by two strategies; subtree replacement or subtree raising [WFT<sup>+</sup>99]. In both cases, the fully expanded tree is constructed. Postpruning then begins at the leaf nodes and works up towards the root node of the tree. Subtree replacement considers each subtree and tests the effect of replacing the subtree with a leaf node labelled



with the majority class. The process continues in an iterative manor towards the root node, until the tree has been pruned. Subtree replacements will certainly cause an increase in error rate on the training data that was used to build the tree, but may still be beneficial when classifying unseen test data. Simply using the training data to look for opportunities to post-prune would not result in any pruning, as the decision tree has been constructed according to specifics of the training data and is therefore heavily biased. One strategy is to hold back some of the training data to use specifically for pruning. This is called *reduced-error pruning*. A disadvantage of this method is that the amount of training data that is available to construct the tree is reduced.

*Pessimistic pruning* has an advantage in that it does not require a separate pruning set, but instead uses the training data to assess the predictability of subtrees that are being considered for pruning. A penalty is added to error rates obtains from the training data in an attempt to overcome bias. To decide whether subtree replacement should occur the estimated error rate of the subtree is compared with the estimated error rate of the proposed leaf node that will replace it. Subtree replacement occurs when the error rate of the replacement is less than that of the subtree to be replaced. The error rate estimations are calculated as follows. Given a confidence  $c$ , the confidence limits  $z$  as given by:

$$Pr \left[ \frac{f - q}{\sqrt{q(1-q)/N}} > z \right] = c \quad (3.12)$$

where  $N$  is the number of samples,  $f$  is the observed error rate and  $q$  is the true error rate. The (pessimistic) estimation of the error rate at the node is then calculated as:

$$e = \frac{f + \frac{z^2}{2N} + z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}} \quad (3.13)$$

Subtree raising differs in that subtrees are moved up to subsume the node found above them. Instances that would have reached other branches extending from the node above (which could be leaf nodes or other subtrees) are reclassified into the branches of the lower subtree. As this process is time-consuming and has a cost associated, it is generally only performed on the branch down which the majority of instances from the training data were directed.

### 3.3.3.3 Cost-complexity Pruning

The cost-complexity algorithm was first proposed by Brieman et al (1986) [BFOS84] and is used in *Classification and Regression Trees* (CART). This approach considers both the error rate (the number of tuples misclassified) and the complexity of the subtree in terms of the number of leaf nodes. The function is controlled by the parameter  $\alpha$ , which is defined as the average error increase per leaf of the subtree being assessed. In a similar way to other postpruning approaches, it begins at the leaf nodes and works up towards the root node. The first step is to generate a sequence of trees beginning with the original tree  $T_0$  and progressing through smaller pruned trees  $T_{i+1}$  until a final tree  $T_k$  is reached, which is a single leaf node. In each iteration, the subtree with the smallest value for  $\alpha$  is pruned. The second stage of the algorithm then selects one of the candidate trees from the sequence of trees with decreasing complexity generated in step 1. The performance in terms of error rate of each tree is assessed either using a hold out data set, or by cross-validation.

For a decision tree  $T$ , where  $N$  is the number of instances from the training set,  $E$  is the number of misclassified instances and  $L(T)$  is the number of leaves, cost-complexity of  $T$  is defined as

$$\frac{E}{N} + \alpha \times L(T) \quad (3.14)$$

If the subtree  $S$  of  $T$  is replaced by the leaf with the majority class,  $M$  more instances would be misclassified but the tree would contain  $L(S) - 1$  fewer leaves. The new tree would have the same cost-complexity as  $T$  if

$$\alpha = \frac{M}{N \times (L(S) - 1)} \quad (3.15)$$

To produce the next tree in the sequence  $T_{i+1}$  from  $T_i$ , the subtree or subtrees with the minimum value of  $\alpha$  are removed and replaced with the majority class leaves.

The second stage selects one of the trees from the sequence generated above. If a hold out set of unseen data  $N$  is used to assess the candidate trees the algorithm proceeds as described below:

Let  $E$  be the minimum number of errors observed with any  $T_i$  and the standard error of  $E$  calculated as

$$se(E') = \sqrt{\frac{E' \times (N' - E')}{N'}} \quad (3.16)$$

The tree selected is the smallest  $T_i$  where the observed number of errors is less than  $E + se(E)$ . Each subtree is assessed in terms of the cost complexity function and is replaced only when the cost complexity is reduced by replacing the subtree with a leaf node.

#### 3.3.3.4 1 Standard Error (1SE) Pruning

In one standard error pruning a comparison is made between the estimated risk of a tree before and after pruning. In other pruning methods, pruning only takes place when the predicted accuracy of the pruned version is greater than or equal to that of the tree being replaced. By contrast, one standard error pruning relaxes this constraint and allows replacement providing the difference in risk estimate does not exceed one standard error, allowing more liberal pruning to take place [Min89].

### 3.3.4 Ensemble Methods of Classification

Another approach is classification by *ensemble learning*. The principle in ensemble learning is to combine a number of classification models so that the overall classification decision is the amalgamation, by an average or weighted average, of the decision outcomes of each model, analogous to decision-making “by committee”. These methods can often greatly increase the accuracy of classification and at the same time are less prone to over-fitting due to the fact that the combined decision avoids any bias that may be caused by the idiosyncrasies of one particular set of training data [MO11] [Bre96a] [Qui96b].

There are a number of methods of combining models such as *bagging* [Bre96a], *boosting* [FS], *random forests* [Leo01] and *stacking* [Wol92]. In all cases the main advantage is one of variance reduction; the combined outcome of diverse classifiers compensates for the error rate of any individual classifier [MO11]. A disadvantage with some but not all ensemble methods is the loss of interpretability, as the base classifiers cannot be visualised in the same way as an individual classifier such as a decision tree. Some ensemble methods do produce a single standard DT (MetaCost) [Dom99a] and some produce new types of DTs such as *Alternating Decision Trees* which contain alternating layers of prediction and splitting nodes and are capable of not only classification but of providing a classification confidence or *margin* for each instance [FM99]. Boosting alternating decision trees using the ADTree algorithm produces a single structure and is therefore more interpretable than applying boosting to

other types of DT, such as C4.5.

The increase in accuracy achieved by combining multiple classifiers will clearly be most beneficial if there is some disagreement between individual classifiers built according to different subsets of training data. If all classifiers vote the same way then there would be no gain by combining their results. It has been demonstrated that the best ensembles combine individual classifiers with high accuracy but that disagree as much as possible. Opitz et al (1996) achieved this by training an ensemble of neural networks using different parameters such as initial weights [OS96].

#### 3.3.4.1 Bootstrap Aggregation (Bagging)

*Bagging* gives equal weight to the outcome of each model. When a new instance is processed, it is classified by each of the base classifiers. Each model ‘votes’ on which class the instance belongs to and the votes are combined. For numerical outcomes the average is calculated and for nominal classifiers the class value with the majority of votes is assigned to the instance [Bre96a]. Bagging uses bootstrap sampling with replacement to create  $k$  artificial data sets from the original data set  $D$ . For each iteration  $i$  ( $i = 1, 2, \dots, k$ ) a data set  $D_i$  is generated by taking a sample of the instances from  $D$ . Some of the instances will not occur in  $D_i$  and some may occur more than once. This reuse of the original dataset, adjusted by bootstrap sampling, means that multiple base classifiers can be constructed without the need for a new independent training data set for each model.

As the DT induction process is generally unstable, differences that exist between the sampled data sets will result in differences in the base classifiers. In other words, it is highly unlikely that an identical classification model will be generated from each sampled set of the training data. It is these differences that allow the overfitting introduced by a single classifier to be ‘smoothed’ out [Bre96a]. More formally, this technique lowers the variance component of the *bias variance decomposition* [KD95]. For a given classification model there are two potential sources of expected error. One arises from the fact that there will be a persistent error rate found for a model even if an infinite number of training data partitions were tested. This is the *bias* component of the expected error rate. The second expected error rate is due specifically to the training data used to construct the model. It is this component that ensemble techniques such as bagging are often able to lower. As mentioned, bagging is most effective in situations where there is instability in the base models. As such, the performance of bagging may be increased if pruning is not applied to the base classifiers, as this will

introduce more diversity among the base models.

An extension of basic bagging can be achieved by using base models that predict a probability estimate rather than just a classification label. In this situation, the ensemble model can calculate the average probability estimate from the underlying models, which is often more accurate than that of a single model. An example is provided by the cost-sensitive meta-classifier MetaCost [Dom99a], which reassigns instances in the training data with the class value that minimises the cost, derived from the probability estimate. These newly assigned class values are then used to construct a single decision tree, one that has the costs of class predictions built into it. This approach is discussed in more detail in the section on cost-sensitivity (Section 3.5).

In summary, bagging can often increase the overall accuracy of classification by reducing the variance between base models. The benefits of bagging are low in situations where there is little diversity amongst the base classifiers and differences present in the bootstrap samples have little effect on the structure of the classification models. Breiman et al (1996) found that on a variety of test data sets, bagging reduced misclassification rates between 6% and 77% but was also found to decrease accuracy of stable classification models [Bre96a].

### 3.3.4.2 Boosting

*Boosting* is an iterative process that uses the performance of one model to influence the construction of the next. Models are constructed that can be described as having “complimentary areas of expertise” in the sense that each model takes into account the incorrectly identified instances of previous models. By updating the weights assigned to each training tuple, subsequent base models can focus on the misclassifications from the previous model. The final ensemble model again combines the class assignments from each base model but also factors in the weight of each classifier’s vote as a function of its accuracy.

For the first iteration, all tuples in the data set  $D$  are assigned the same weight,  $\frac{1}{d}$  where  $d$  is the number of tuples in  $D$ . Sampling with replacement is used, as in bagging, to create a training set. The difference with boosting is that the likelihood of a tuple being included in the training set is a function of the weight assigned to each tuple. After each model, the weight is increased for instances that are misclassified and decreased if the classification was successful. The way in which the weights are adjusted is derived from the error rate of  $M_i$ , the current model. This error rate is calculated as:

$$error(M_i) = \sum_{j=1}^d w_j \times err(X_j) \quad (3.17)$$

All tuples that were classified correctly are multiplied by  $error(M_i) / (1 - error(M_i))$ . The weights of all tuples are then normalised. This has the affect of increasing the weights of misclassified tuples and reducing the weights of those correctly classified. Finally, the votes are counted for each classifier, taking into account the performance of each one, with classifiers having a lower error rate being given a higher weighted vote. The weight of the vote for a classifier  $M_i$  is calculated as

$$\log \frac{1 - error(M_i)}{error(M_i)} \quad (3.18)$$

Freund et al (1996) boosting outperforms bagging when the base algorithm generates fairly simple classifiers, such as simple classification rules and tests on conjunctions of attributes but less significantly when combined with C4.5. Similarly to bagging, boosting showed the largest increases in accuracy on unstable classifiers however, unlike bagging, boosting actively seeks to force weak learning classifiers to change their hypotheses by changing the distribution over the training examples as a function of the errors made by previously generated hypotheses [FS].

Opitz et al (1999) compared the performance of bagging and boosting ensemble techniques using Decision Trees and neural networks as the base classifier. The results showed that bagging generally produced an increase in accuracy for both DTs and NNs, whilst the results for boosting were more varied. Boosting increased accuracy even when compared to bagging on some datasets but on others performed worse than a single classifier. The authors suggest an explanation for this may be Boosting's sensitivity to noise, supporting the conjecture of Freund and Schapire (1996) [FS]. The study also investigated the effect of increasing the number of component classifiers used in an ensemble and found that the largest gains in accuracy occur with the first few additional classifiers, with little increase occurring beyond 25 classifiers. Breiman et al (1996) also investigated the effect of increasing the number of bootstrap replicates, measuring the misclassification rates for 10, 25 50 and 100 bootstrap replicates. The experimental data had an unbagged error rate of 29.1%, which reduced to 21.8% when 10 bootstrap replicates were used. Increasing this number further, however, only reduced the error rate to 19.4%, 19.3% and 19.3% for 25, 50 and 100 replicates respectively [Bre96a].

### 3.3.4.3 Random Forests

*Random forests* (RF) possess a number of advantages over other ensemble techniques and have been widely applied in the field of Life Sciences, particularly omics data (proteomics, genomics, metabolomics, transcriptomics) where the number of attributes can often be vast [FPF10]. RFs combine two methods to build classification models, namely bagging and random feature selection. As such they introduce randomness in two, possibly complimentary areas. Sampling with replacement is conducted as with standard bagging, producing bootstrap samples for training that contain  $n$  number of instances randomly drawn with replacement from the learning set of  $n$  instances [RŠ04]. A random set of attributes is then generated and the base model is constructed using only those attributes using the Gini index discussed in Section 3.3.2.4 as the splitting attribute selection mechanism. In a paper focusing on possible ways to improve RFs, Robnik-Sikonja (2004) suggests that the use of the Gini index does have a disadvantage in that it evaluates each attribute independently and does not take into account the context of other attributes, which may result in poor performance. The alternative examined by the author was to use a combination of impurity measures, such as Gain ratio, Relieff [RŠK03] and MDL. Combining attribute evaluation measures for split selection was found to result in a slight increase in performance especially evident on data sets with highly dependent attributes [RŠ04], however combinations of attribute selection measures were not investigated in this study.

Base DTs are constructed using the CART method [Sti84] [OS84]. Pruning mechanisms typically used to make DTs more general and less prone to overfitting are not necessary in RFs and are generally not performed. The rationale for this is that any individual tree suffering from overfitting training data only contributes a single unweighted vote in the ensemble and as a result will be ‘smoothed out once all the votes are counted. For this reason RFs are generally not susceptible to overfitting and noise. An alternative to selecting a random vector of attributes to split on is to create random linear combinations of attributes and use them to establish splitting criteria on which to build a decision tree [Leo01]. This approach may be more suitable than the previous method when the number of attributes available is small. As only a subset of attributes are considered for splitting each node, Random Forests can generally be constructed more quickly than other bagging or boosting methods.

Breiman (2001) also addressed the question of how many attributes to select randomly for splitting. The results were found to be insensitive to the number of attributes,

with one or two features giving near optimum results [Leo01]. The other main parameter in RF and other ensemble techniques is the number of classifiers to include. Breiman (2001) found that most reductions on test-set errors occurred with the addition of as few as ten classifiers, with some measurable improvements for up to 25 classifiers for Adaboosting and Arcing, at which point the graphs appeared to have asymptoted to a plateau [Leo01].

The main advantages of RF algorithms are summarised by Breiman et al (2001) [Leo01] as:

1. It's accuracy is as good as Adaboost and sometimes better.
2. It is relatively robust to outliers and noise.
3. It is faster than bagging or boosting.
4. It gives useful internal estimates of error, strength, correlation and variable importance.
5. It is simple and easily parallelised.

#### **3.3.4.4 Option Trees and Alternating Decision Trees**

Option trees attempt to combine the accuracy of ensemble learning models but still produce a single, interpretable decision tree as an output. The strategy involves the introduction of a new type of internal node, an option node. When a tuple being processed by the model reaches an option node, all branches are followed. This means that the tuple will arrive at more than one leaf node. The results of each of these leaf nodes are then combined to give an overall classification. Giving each node an equal vote is not particularly useful for binary splitting option nodes, as a majority will only be reached if both outcomes are the same. A more sophisticated approach is to use probability estimates for each leaf node at which the tuple arrives to combine into an overall result.

*Alternating Decision Trees* also include option nodes (referred to as prediction nodes). They represent binary splits where each branch is assigned either a positive or negative value. For a given tuple, all the numbers from every possible path through the tree are summed and the class value assigned depending on whether the sum is a positive or negative value. They are grown using a boosting algorithm with every node in the decision tree being considered for replacement by a prediction node [FM99].



### 3.3.4.5 Stacking

*Stacking* is similar to bagging and boosting in that the class predictions of multiple base models are combined to provide an overall classification. The difference is that whilst bagging and boosting are generally applied to base models of the same kind, i.e. decision trees, stacking is used to combine the predictions of different types of classification models. Rather than a simple voting system as before, stacking uses the concept of a *metalearner* to combine base level predictions. An instance to the ensemble model has as many attributes as there are base models, with the prediction of each model stored in one of the attributes. Rather than using the same training data from the base level to train the metamodel, a holdout set is used so as to avoid any bias from the training data used to build the initial models.

### 3.3.5 Support Vector Machines (SVM)

SVM is a classification algorithm capable of classifying linear and non-linear data. Instances that describe the boundaries or dichotomizing *hyperplanes* and the margins of these boundaries are referred to as the *support vectors* and are used to train the model. All other instances are irrelevant which makes the classification boundary in SVMs more stable, as only the support vectors affect the position and orientation of the hyperplane. SVMs are therefore insensitive to outliers and less prone to overfitting. A disadvantage is that they are computationally expensive compared to other learning algorithms such as Decision Trees. For non-linear data, kernel functions are used to transform the original data into higher dimensional data until a separating hyperplane is established.

### 3.3.6 Neural Networks (NN) and Backpropagation

Neural Networks provide another approach to building classification models. NN operate by passing data from interconnected nodes or ‘neurons’, from input nodes to output nodes either directly or via one or more hidden layers. Each connection has a weight associated with it. *Single layer perceptrons* can approximate linear functions of the inputs, whereas *multilayer perceptrons*, such as feed forward networks with hidden layers can implement non-linear relationships in the data. NN can be difficult to interpret and are computationally more expensive than DTs, but are very insensitive to noisy data and are capable of classifying patterns on which they were not trained.

### 3.3.6.1 Basic Structure of a Multi-layer Feed-forward Network

The NN used in this project was a *multi-layer feed-forward network*. This type of NN is described as fully connected as each unit has a connection to every unit in the next layer. The feed-forward part of the name comes from the fact that data moves through the network in one direction only, with no samples being fed back into previous layers. The three types of layer are:

- Input layer - inputs are fed simultaneously into the neurons of the input layer
- Hidden layer(s) - the weighted outputs of the input layer are fed into the neurons of the first hidden layer. The weighted outputs from these neurons are fed into the next layer and so on.
- Output layer - one output neuron per class outcome is present in the output layer. In a two class system it is possible to have just one output node, with the values of 0 and 1.

The algorithm used to process data through the NN is described as follows:

- *Initialise the weights* - The weights from the input nodes are initialised. Training data is then passed through the network. For each sample, the mean squared error between the predicted class and the actual class is minimised by adjusting the weights of each connection in a backward direction, from output node to hidden layers to input node. Weights throughout the network are initially set randomly. The values used are typically between -1 to 1. Each unit also has an associated bias, which again is initially set randomly.
- *Propagate the inputs forwards* - Compute the net input and output of each node in the hidden/output layers. The input to one of these nodes is calculated by a linear combination of its inputs - each input connected to the unit is multiplied by its weight and these are summed. A bias value is then added.

Backpropagation

$$I_j = \sum_i w_{ij} O_i + \theta_j \quad (3.19)$$

where  $w_{ij} O_i$  is the weight of the connection from unit  $i$  in the previous layer to unit  $j$ ;  $O_i$  is the output from unit  $j$  in the previous layer;  $\theta_j$  is the bias of the unit.

An activation function is then applied to the input of each hidden/output layer node in the following way:

$$O_j = \frac{1}{1 + e^{-I_j}} \quad (3.20)$$

This is sometimes referred to as a squashing function as it maps a large input domain to a smaller range, 0 to 1. This logistic function is capable of modelling linearly inseparable classification problems, as it is non-linear and differentiable.

Backpropagate the error:

Weights and biases are updated according to the networks error rates. The error rate for output layer units are calculated by

Error of unit  $j$  (output layer)

$$Err_j = O_j (1 - O_j) (T_j - O_j) \quad (3.21)$$

where  $O_j$  is the actual output of unit  $j$  and  $T_j$  is the true output of unit  $j$  according to the known class labels.

The output error rate for hidden layer units is calculated as

Error of unit  $j$  (Hidden layer)

$$Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk} \quad (3.22)$$

where  $w_{jk}$  is the weight of the connection from unit  $j$  to unit  $k$  in the next layer and  $Err_k$  is the error rate of unit  $k$

Weights and biases are updated to reflect the propagated errors. Weights are updated by

$$\Delta w_{ij} = (l) Err_j O_i \quad (3.23)$$

Variable  $l$  is a constant called the learning rate.

Biases are updated by

$$\Delta \theta_j = (l) Err_j \quad (3.24)$$

Variable  $l$  is a constant called the learning rate.

*Case updating* is when the weights and biases are updated after a single sample. *Epoch updating* is when the weights and biases are adjusted after the entire training set has been processed through the network.

- *Terminating Conditions* - processing terminates when one of the following conditions are met:
  - All  $\Delta w_{ij}$  in the previous epoch were below a threshold
  - The percentage of samples misclassified in the previous epoch fell below a threshold
  - The maximum number of iterations has been reached.

### 3.4 Evaluation

A range of supervised learning algorithms were assessed in terms of the ability to make accurate predictions on the pass or fail status of LC-MS analyses. Various measures were considered to quantify various aspects of performance. LC-MS data in this domain is a binary decision of *pass* or *fail* and as such there are four possible outcomes for each analysis;

- *True positive (TP)* - Actual pass that was correctly predicted as a pass.
- *True negative (TN)* - Actual fail that was correctly predicted as a fail.
- *False positive (FP)* - Actual fail that was predicted as a pass.
- *False negative (FN)* - Actual pass that was predicted as a fail.

As an approximate measure, the overall success rate or *accuracy* and *error rate* for predicted outcomes on test data can be used as a simple measure of the performance of learning models. *Accuracy* is the number of correct classifications divided by the total number of classifications and is calculated as shown in 3.25

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.25)$$

Conversely, *error rate* is calculated simply as in 3.26

$$error\ rate = 1 - accuracy \quad (3.26)$$

The *kappa statistic* is a measure of overall success rate that also takes into account how many correct classifications would be expected from a random classifier. The number of correct classifications expected by chance is deducted from the results for a learning model yielding a percentage, with 100% indicating a perfect performance by the model and 0% a performance no more accurate than would be expected by chance.

The error rate on training data, also called *re-substitution error*, is not a reliable indication of performance as it will always be overly optimistic. The training data is used to generate the model and so an optimal performance when the training data is then run against the model would be expected.

Error rates on test data alone are insufficient to make the most meaningful comparisons between different learning models as no distinction is made between the different types of misclassification. Two models could potentially have the same error rate but perform differently, with one model poor at distinguishing false positive and the other false negatives. Measures such as *recall*, *precision* and *F-measure* were also compared, all of which examine the overall accuracy taking into account the different classification errors.

*Recall* expresses the number of *true positives* as a percentage of the total number of positives (3.27):

$$recall = \frac{TP}{TP + FN} \quad (3.27)$$

*Precision* calculates how often a prediction of pass is correct by comparing the number of true positives with the total predicted to be passes (3.28):

$$precision = \frac{TP}{TP + FP} \quad (3.28)$$

*F-measure* combines recall and precision into a single value (3.29):

$$F\text{-measure} = \frac{2 \times precision \times recall}{precision + recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (3.29)$$

In the domain of interest here, a classifier should behave conservatively, with a low recall and high precision as the highest percentage of those analyses automatically passed was desired, with any ambiguous analyses being sent for manual review.

As the *true positive rate* (TPR) of a classification model increases a cost is incurred in terms of the *false positive rate* (FPR). The optimum situation is one where

an increase in TPR can be achieved with minimal increase in FPR. The relationship between these two rates can be visualised using *Receiver Operating Characteristic curves* (ROC) and the area under the ROC curve (AUC) describes the quality of the classifier, a larger AUC signalling a better performance.

## 3.5 Cost-sensitive Classification

The classification algorithms discussed so far are designed to minimise *zero-one loss* and assume equal costs for all types of misclassifications. As with many real world data sets this assumption is incorrect for the LC-MS data analysed in this project. A false positive result for an analysis could potentially lead to a low quality compound being included in the screening collection. The reasons for failure could vary; the purity was recorded as lower than 85%, an impurity more sensitive to UV absorption was present, the compound could not be detected by UV or the compound was pure but was a different structure to the one expected. For most high throughput screening campaigns, activity in an assay is detected by a cluster of hits rather than a single reactive compound.

Libraries of compounds are synthesised such that a number of compounds share a similar parent structure or in other words occupy a similar point in the chemical space represented by the screening set. As such, the inclusion of a compound with a false positive classification is considered low-cost. Furthermore, it is possible that the compound could show high activity in an assay despite the presence of an impurity or even because of it. If such a compound was to pass through to lead optimisation screening, it would be re-synthesised at which point any activity due to an impurity would be elucidated. By contrast, a false negative could result in a high quality compound being excluded from the screening collection or disposed of altogether. Given the costs associated with the design, synthesis, purification, transportation (often cross-continent), and the potential cost of a highly active compound being excluded from screening, the costs of false negative misclassifications are considered to be much higher. To address this inequality, cost-sensitive classification was investigated.

### 3.5.1 Cost-sensitivity by Stratification

There are two well established approaches to achieve cost-sensitivity. The first is *stratification*, where the proportion of members of a particular class are altered, either by

oversampling or undersampling, to bias the classification model towards a particular class outcome. For example, as false negative misclassifications are more costly than false positives the proportion of failed analyses could be artificially increased in the training data set to enhance the ‘focus’ of the classification algorithm on this class. Whilst this approach was not applied in this project it is worth noting that a significant proportion of the analytical data used has been generated from activities aiming to increase the overall quality of the compound collection by specifically targeting suspected low quality samples, so the numbers of failed analyses is somewhat higher than would be routinely expected.

There are a number of problems with stratification as a strategy for cost-sensitivity. If oversampling is used, computation time increases whereas if undersampling is used not all the available data is included when constructing the classifier. The distribution of instances is distorted with can impair the performance of the algorithm. Finally, stratification is best suited to binary classification problems [Dom99b].

### 3.5.2 Cost-sensitivity via a Cost-matrix

The second method of cost-sensitivity is to use a cost matrix. Different costs for misclassifications can be defined as a cost matrix  $C(i,j)$  where  $i$  is the actual class and  $j$  is the predicted class, as show below [EGJ09].

$$C(i,j) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (3.30) \quad C(i,j) = \begin{pmatrix} -1 & 20 \\ 0.5 & 0 \end{pmatrix} \quad (3.31)$$

Equation 3.30 represents a cost matrix that assumes the same cost for FP and FN misclassifications. Equation 3.30 defines a cost matrix that assigns a much higher cost to FN misclassifications than to FP and ‘rewards’ TP more than TN by assigning a negative score.

Such cost matrices can be applied in data mining in two ways. The first is *cost evaluation* where the cost matrix is applied to the predictions of a classifier, with the effect of altering the assessment of performance according to the differing costs of misclassifications. The second, more powerful approach is to make the classifier cost-sensitive by incorporating the cost matrix during the construction of the classification model. The first approach allows the evaluation of the predictions to be altered according to costs, whereas the second alters the predictions themselves.

### 3.5.3 MetaCost

Cost-sensitive classification models were investigated using the *MetaCost* meta-learner [Dom99b]. MetaCost is not an algorithm for building classification models but rather a ‘wrapper’ that can be used in conjunction with an arbitrary base classifier to make the underlying classification algorithm consider a cost matrix during its construction. MetaCost doesn’t make any changes to the base classifier and in fact has no knowledge of its type, instead treating it like a black box. For this reason, MetaCost can be applied to a range of different classification algorithms.

The basic idea of MetaCost is to relabel training examples with the estimated minimal-cost class and then apply the error-based learner to the newly labelled training set [Dom99b]. The effect of this relabelling is that the base classification model learns according to the cost-minimising frontiers, rather than the zero-one loss frontiers of the original dataset. It is this shift in the boundaries within the data that makes the underlying classification models cost-sensitive. Empirical assessment of MetaCost found that it almost always produces large cost reductions when compared to a cost-blind classifier (C5.0) and to two forms of stratification (oversampling and undersampling). It was also found to scale well on large data sets [Dom99b].

## 3.6 Summary of Classification Models

This section contains a list of the classification models investigated in this project and some brief details of their notable features. A further summary and comparison is provided in Table 3.2

- *ADTree* - alternating decision tree that uses boosting.
- *BFTree* - Builds decision trees using a best-first expansion of nodes. Pre- and post-pruning methods are used and will yield different pruned tree structures than depth-first strategies.
- *DecisionStump* - Builds one-level binary decision trees using either categorical or numerical attributes
- *FT* - builds trees with linear functions at the leaves and uses C4.5s splitting criterion to choose attributes to split on.
- *J48* - Modified C4.5 algorithm. Generates a decision tree by recursive partitioning. The split that gives the best information gain from all possible splits is



selected. For continuous attributes, data at each node is sorted by the continuous attribute and the entropy gain based on each distinct value are calculated in one scan of the sorted data.

- *LMT* - builds logistic model trees
- *NBTree* - Hybrid decision tree/*Naïve Bayes*.
- *RandomForest* - Constructs random forests by bagging ensembles of random trees
- *REPTree* - Fast decision tree which builds a decision/regression tree using information gain calculations to select the splitting criteria. Trees are pruned using reduced error pruning. Values for numeric attributes are only split once to maximise speed.
- *SimpleCart* - uses a minimal cost-complexity pruning strategy as used in *Classification and Regression Trees (CART)*. As subtrees are pruned the increase in error rates on training data, relative to their size, is monitored and measured as the value  $\alpha$ . The average error rate per leaf of the subtree is considered. Pruning continues with the subtrees exhibiting the smallest value of  $\alpha$  being pruned creating a sequence of successively smaller trees.

This chapter introduced the various steps involved in the KDD process and how they were applied to the data studied in this project. The classification algorithms used to build learner models were described, as were the methods of evaluation. The principles of cost-sensitive classification were also discussed along with the methods used to apply cost-sensitivity. Chapter 4 contains the results of the experimental work on classification models.

Table 3.2.: Comparison of Classification models.

Name	Description	Splitting Attribute Selection	Pruning	Ensemble
J48	Modified C4.5	Gain ratio	Subtree raising (option to use reduced-error pruning)	
BFTree	Best-first decision tree	Information/Gain ratio	Best-first based pre/post pruning	
FT	Builds trees with linear functions at the leaves	Functional splitting criteria	Pruning using backed-up-error and static error	
ADTree	Alternating decision tree	Weighted error of added rule	None	Boosting
DecisionStump	One-level binary decision tree	Gini index	None	
LMT	Logistic Model trees	LogitBoost procedure	Cost-complexity pruning	
NBTree	Hybrid decision tree/ <i>Naive Bayes</i>			
RandomForest	Constructs random forest from random trees	Gini index	None	Bagging
REPtree	Decision/regression tree	Information gain	Reduced-error pruning	
SimpleCart	Classification and Regression Tree (CART)	Gini index	Cost-complexity pruning	

# Chapter 4

## Experiments

This chapter contains a description of the experiments on data mining and the results obtained. The experiments are split into three categories; the effect of different methods of pruning on model complexity and performance, an evaluation of a range of classification models and an evaluation of the effect of cost-sensitivity on classification model performance. Variable importance measures are also described. Finally, the chapter concludes with the results of the implementation of Random Forests using Pipeline Pilot.

### 4.1 Pruning and Model Stability

Many studies have shown certain learning algorithms to be ‘unstable’, where small differences in training data can produce large changes in the models produced, and therefore in classification accuracy [Bre96b] [LS97] [MO11]. Breiman et al 1996 found that this was particularly true of decision trees and neural networks [Bre96b]. During preliminary experiments in this project it was noted that when 10-fold cross validation was applied to training data the resultant models often varied substantially in size and complexity. To investigate the effect of pruning on tree complexity, a number of classification models were examined in terms of number of nodes/leaves and variance in complexity, comparing unpruned models with several pre-pruning and post-pruning techniques. This also allowed the effect of these various pruning methods to be evaluated with regard to the ability to generate smaller, more generalised trees. Finally, the effect of this reduction in complexity was evaluated using several measures of predictive accuracy to investigate the effect of pruning on performance.

The data set Set\_100k (Table 3.2.2) was used to build DTs using various pruning

Table 4.1: Tree complexity and performance using different pruning methods

Model/pruning	Mean	S.D.	Mean	S.D.	Accuracy	Precision	Recall	F-measure	ROC Area	FP Rate	FN rate
	leaf nodes	leaf nodes	tree size	tree size							
J48 no pruning	636.2	49.49	1271.40	98.98	97.36	0.973	0.974	0.973	0.957	0.012	0.114
J48 subtree	369.8	34.65	738.60	69.30	97.49	0.975	0.975	0.975	0.966	0.011	0.114
J48 REP	116.9	27.49	232.80	54.99	97.46	0.974	0.975	0.974	0.983	0.009	0.125
J48 REP subtree	116.1	25.88	232.80	54.99	97.46	0.974	0.975	0.974	0.983	0.009	0.125
J48 minNum 100	22.2	2.658	43.40	5.32	97.42	0.974	0.974	0.974	0.979	0.01	0.124
BFTree unpruned	1973.7	26.27	3946.40	52.54	96.69	0.967	0.967	0.967	0.892	0.02	0.115
BFTree prepruning ISE	617	707	1233.00	1414.87	97.24	0.972	0.972	0.972	0.946	0.012	0.123
BFTree prepruning	204.7	631	408.40	1262.32	96.88	0.969	0.969	0.968	0.908	0.005	0.193
BFTree postpruning	526.6	526.5	1052.20	1052.96	97.38	0.973	0.974	0.973	0.947	0.01	0.126
SimpleCart no pruning	1978.5	26.92	3956.00	53.85	96.69	0.967	0.967	0.967	0.933	0.02	0.113
SimpleCart CCP	61.4	18.88	121.80	37.75	97.49	0.975	0.975	0.975	0.984	0.009	0.121
SimpleCart CCP ISE	31.1	8.144	61.20	16.29	97.47	0.974	0.975	0.974	0.979	0.009	0.127
SimpleCart minNum 100	42.7	3.466	84.40	6.93	97.32	0.973	0.973	0.973	0.992	0.01	0.129
LMT nopruning	40.5	10.00	80.00	20.01	97.58	0.975	0.976	0.975	0.992	0.009	0.118
LMT minNum 100	40.3	10.89	79.60	21.79	97.57	0.975	0.976	0.975	0.992	0.009	0.117
FT no pruning	242.2	25.35	483.40	50.71	97.25	0.972	0.973	0.972	0.95	0.013	0.119
FT minNum 100	23.9	3.281	46.80	6.56	97.35	0.973	0.974	0.973	0.952	0.01	0.128

methods and included the attributes listed in Section 3.2.4. 10-fold cross validation was applied in each case to produce sets of ten DTs. The average, standard deviation and variance of both the number of leaf nodes and the total number of nodes in the trees were recorded. Measurements of accuracy, precision, recall, ROC area, kappa statistics and FP/FN rates were considered to evaluate the performance of each algorithm and pruning method. These performance measures were calculated by combining the results of all ten models in each set. The results are shown in Table 4.1.

#### 4.1.1 J48 Pruning

Sets of ten DTs were generated using the J48 DT algorithm firstly with no pruning with the minimum number of instances per leaf node set to 1. This set was compared to J48 with subtree raising (Section 3.3.3.2), reduced error pruning (Section 3.3.3.2) and a minimum number of instances per leaf node of 100 (Section 3.3.3.1). Whilst subtree raising and reduced error pruning produce smaller DTs the largest reduction in both number of leaf nodes and overall tree complexity was achieved by the minimum instances threshold. The results are summarised in Figure 4.1. All pruning approaches caused a slight increase in overall accuracy, precision and ROC area compared to the unpruned models. All reduced the FP rate but the FN rate increased from 0.114 to 0.125 for reduced error pruning and reduced error pruning combined with subtree raising and to 0.124 for minimum number of instances set at 100. The largest reduction in tree complexity was achieved using the minimum instances threshold, reduced from a mean number of nodes of 1271.4 to 43.4. Whilst there was some instability in the sets of J48 models, the standard deviation and variance in all cases was low (see Table 4.1).

#### 4.1.2 BFTree Pruning

Sets of 10 BFTree models were constructed using 10-fold cross validation. Pre-pruning (with and without 1SE), post-pruning and minimum number of instances were compared to a set of unpruned DTs. The pre-pruning technique implemented in the BFTree algorithm is best-first based pre-pruning where further splitting is stopped when it appears to increase the error rate as described in Shi 2007 [Shi07b]. The post-pruning strategy for BFTree works by building all training folds in parallel. An average error estimate is calculated for the temporary trees in all fold, producing a sequence of error rates for each expansion. The number of expansions with the minimum error rate is selected as the number of splits and a final tree is then generated according to this

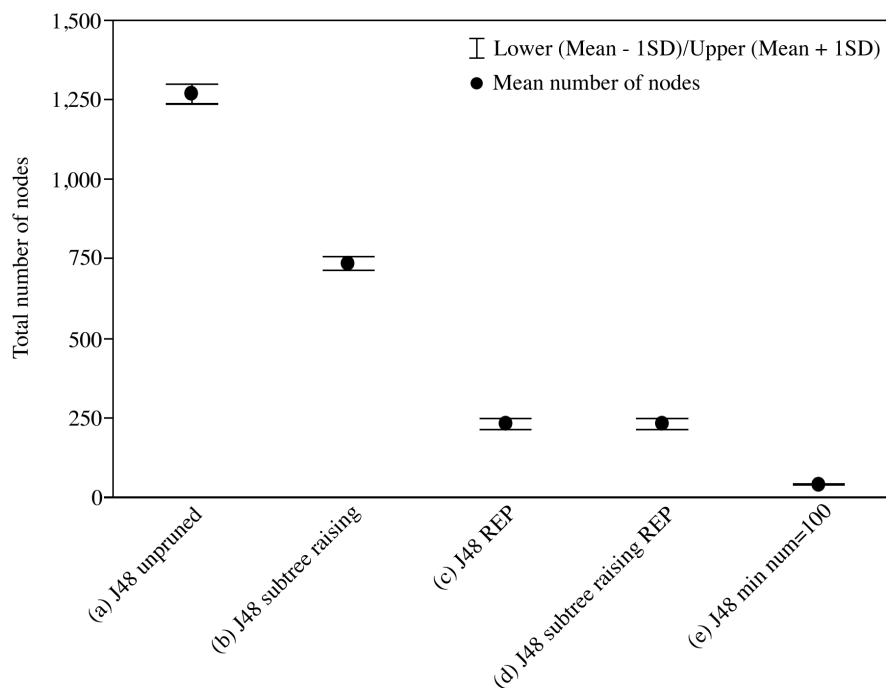


Figure 4.1: Pruning methods for the J48 DT algorithm

number [Shi07b].

Again, the most effective strategy at reducing tree complexity was the application of a minimum number of instances per leaf node as shown in Figure 4.2 . The three other pruning techniques reduced tree size significantly and also produced sets of models with greater variance than the unpruned or minimum threshold sets. All pruning methods produced more generalised trees with minor increases in accuracy, precision, recall and ROC area. Pruned DTs also showed an decrease in FP rates and an increase in FN rates.

### 4.1.3 SimpleCART Pruning

Unpruned DTs were generated using the SimpleCART algorithm and compared to cost complexity pruning (Section 3.3.3.3) with and without 1SE (Section 3.3.3.4) and a minimum number of instances per leaf node of 100 (Section 3.3.3.1). The results are summarised in Figure 4.3 All three pruning methods caused a large reduction in tree complexity from a mean number of nodes of 3,956 for the unpruned DTs compared to 121.8, 61.2 and 84.8 for cost complexity pruning, cost complexity pruning with 1SE

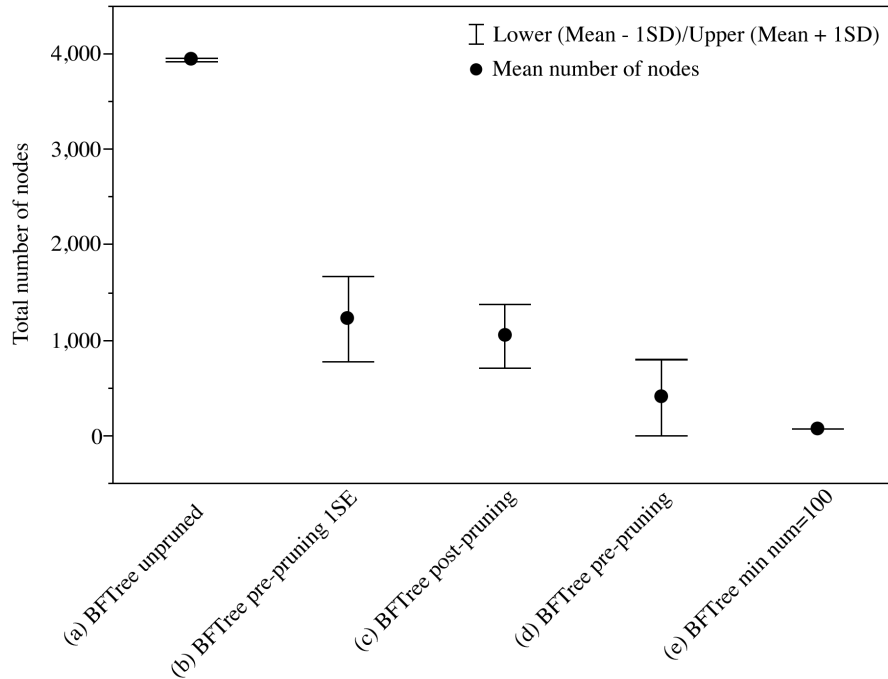


Figure 4.2: Pruning methods for BFTree algorithm

and minimum instances threshold respectively. Again, despite such large reductions in tree size, the accuracy, precision, recall and ROC area all showed minor improvements for the pruning methods. Variance among sets of DTs was lower for SimpleCART than with J48 showing that this algorithm was also stable when constructed using the vector of attributes described in Section 3.2.4.

#### 4.1.4 Functional Tree and Logistic Model Tree pruning

The FT algorithm has a built in post-pruning method that constructs functional leaves using a bottom-up, post-order strategy. Estimations of *static error* (estimated error if the node were a leaf) and *back-up error*, described as the ‘weighted sum of the estimation of the errors of all subtrees of the current node’ [Gam04] are calculated. Nodes are converted to leaves if the back-up error is greater than or equal to the static error. The algorithm also applies a boosting mechanism, as described in Section 3.3.4.2. A minimum number of instances threshold of 100 was also applied to a set of 10 FT models to investigate the affect of this type of additional pre-pruning. The application of this threshold reduced tree complexity from a mean of 483.4 to 46.8 with a minor increase in all the performance measures with the exception of the FN rate, which increased

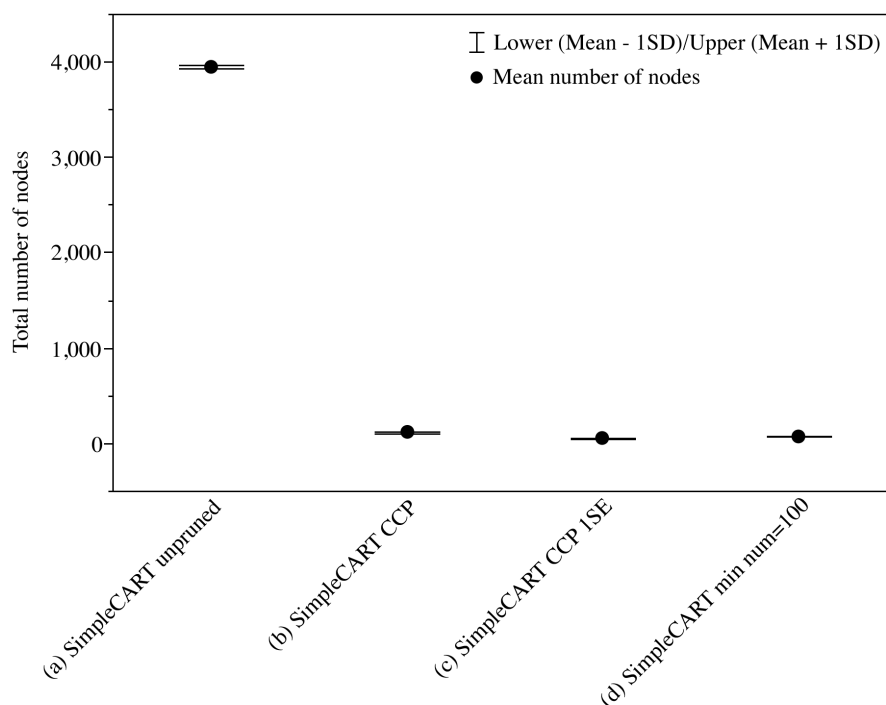


Figure 4.3: Pruning methods for SimpleCart DT algorithm

from 0.119 to 0.128.

The LMT algorithm also contains the cost complexity pruning method (Section 3.3.3.3) employed by SimpleCART. In the paper by Landwehr et al 2005, the authors note that correctly pruned logistic model trees will usually be much smaller than ordinary classification trees due to the complexity of regression functions at the leaves [LHF05]. This was found to be the case with the application of a minimum number of instances threshold of 100 resulting in little reduction in tree complexity, as shown in Figure 4.4.

#### 4.1.5 Summary of Results for Pruning Methods

With the exception of LMT, which generated compact models using the in-built pruning methods discussed earlier, the application of a minimum number of instances per leaf node of 100 was found to have the most significant effect on reducing tree size and complexity, with a minor increase in performance on test data sets. This suggests some degree of overfitting on training data with unpruned models, with large numbers of nodes created at the lower levels of DTs where the number of instances passing



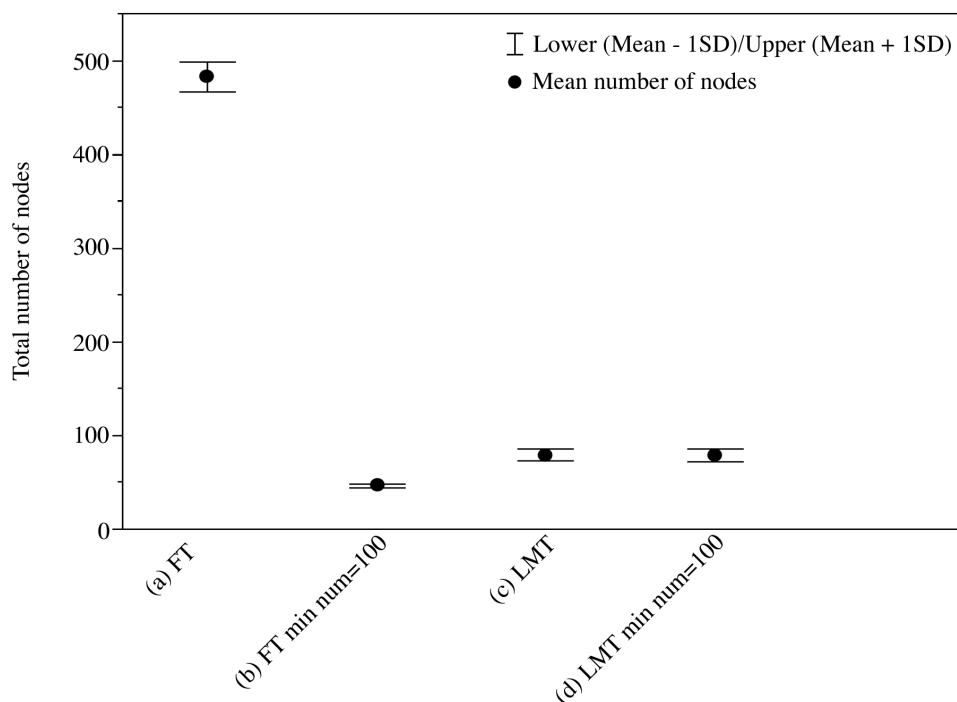


Figure 4.4: FT and LMT

these decision points was less than 100, or  $<0.1\%$ . At these decision points, the splitting criteria were calculated as the ‘best’ attribute to split on, but none of the available attributes were particularly useful in terms of further partitioning of the instances that reached those nodes. A minimum instances threshold of  $0.1\%$  was selected as the most effective pruning method and was applied to subsequent classification models, sometimes in combination with other post-pruning methods.

The BFTree algorithm with pre- and post-pruning was found to produce the most unstable models. The remaining DTs were found to show low variance with regard to model size and complexity, suggesting these algorithms produced stable trees when constructed with the attributes described in Section 3.2.4.

## 4.2 Evaluation of Classification Models

A number of classification models were evaluated with regards to the models’ ability to predict the pass/fail outcome for each integrated peak of LC-MS analyses. In accordance with a “simplicity-first” approach to data mining, learning models of increasing

complexity were evaluated, beginning with simple classification rules and progressing through more complex algorithms. The data set Set\_500k (Table 3.2.2) was used to construct each of the models listed in Table 4.2. Various aspects of the models' performance were evaluated using the methods described in Section 3.4. The precision, recall, f-measure and ROC area are combined values for pass and fail outcomes. The false positive rate is the rate of failed peaks misclassified as passes and the false negative rate represents passed peaks misclassified as fails.

Table 4.2: Performance of Classification Models

Model	Accuracy	Precision	Recall	F-measure	ROC Area	Kappa	FP Rate	FN rate
ZeroOne	85.95%	0.739	0.86	0.795	0.5	0	0	1
DecisionStump	94.84%	0.949	0.948	0.949	0.902	0.7899	0.033	0.164
SMO	96%	0.96	0.96	0.96	0.913	0.8331	0.022	0.152
FT	97.43%	0.974	0.974	0.974	0.954	0.891	0.01	0.121
NaiveBayes	93.28%	0.947	0.933	0.937	0.977	0.7572	0.068	0.064
REPTree	97.27%	0.972	0.973	0.972	0.978	0.8831	0.009	0.139
J48	97.38%	0.973	0.974	0.973	0.979	0.8883	0.009	0.128
J48 (Bagging)	97.44%	0.974	0.974	0.974	0.981	0.8908	0.009	0.128
SimpleCART	97.86%	0.979	0.979	0.979	0.986	0.9102	0.007	0.105
ADTree	97.10%	0.971	0.971	0.971	0.991	0.8762	0.011	0.14
Multilayer	97.31%	0.973	0.973	0.973	0.991	0.8859	0.011	0.124
NBTree	97.60%	0.976	0.976	0.976	0.994	0.8988	0.01	0.108
LMT	97.84%	0.978	0.978	0.978	0.994	0.9084	0.008	0.103
RF (5 trees)	99.88%	0.999	0.999	0.999	1	0.9952	0	0.005
RF (10 trees)	99.97%	1	1	1	1	0.999	0	0.001
RF (20 trees)	99.99%	1	1	1	1	0.9999	0	0
RF (50 trees)	100%	1	1	1	1	1	0	0

Due to the majority of integrated peaks being fails, the ZeroOne algorithm simply predicted the class of all peaks as fail and was accurate for 85.95% of instances. Clearly this rule is of no use in making predictions, reflected in a Kappa score of zero, a FP rate of zero and a FN rate of one. It is useful however in establishing a baseline against which the other classification models can be compared. The single test on *percent purity by UV* produced by the Decision Stump, the output of which is displayed in Figure 4.5, is successful in predicting a large number of instances, with measurements of 0.949 and 0.902 for f-measure and ROC area respectively. This highlights the fact that as expected, the attribute *percent purity by UV* is the single most important variable

```

Classifications
percent_purity_uv <= 78.725 : fail
percent_purity_uv > 78.725 : pass
percent_purity_uv is missing : fail

Class distributions
percent_purity_uv <= 78.725
pass: 0.026895335190230442 fail: 0.9731046648097695

percent_purity_uv > 78.725
pass: 0.8048857346413109 fail: 0.1951142653586891

percent_purity_uv is missing
pass: 0.14049393336200774 fail: 0.8595060666379922

```

Figure 4.5: Decision Stump output

in the decision of pass or fail, as confirm by the results of variable importance discussed in Section 4.4.

The more sophisticated algorithms all increased predictive performance of the remaining instances, with Random Forests (RF) showing the highest performance on all measures (see Table 4.2). All RF models showed high accuracy with very few misclassification even when using an ensemble of 5 trees. Increasing the number of trees combined to 10, 20 and 50 trees produced marginal increases in accuracy, with the 50 tree RF achieving 100% accuracy.

The ROC Area curves generated by a selection of the algorithms from Table 4.2 are shown in Figure 4.6

### 4.3 Cost Sensitive Classification using MetaCost

The MetaCost algorithm [Dom99b] was used as a wrapper to convert the base classifiers into cost-sensitive models, as described in Section 3.5, using the following cost matrix:

$$C(i, j) = \begin{pmatrix} 0 & 10 \\ 2 & 0 \end{pmatrix} \quad (4.1)$$

The values in the cost matrix were used to reflect the relative high cost of false

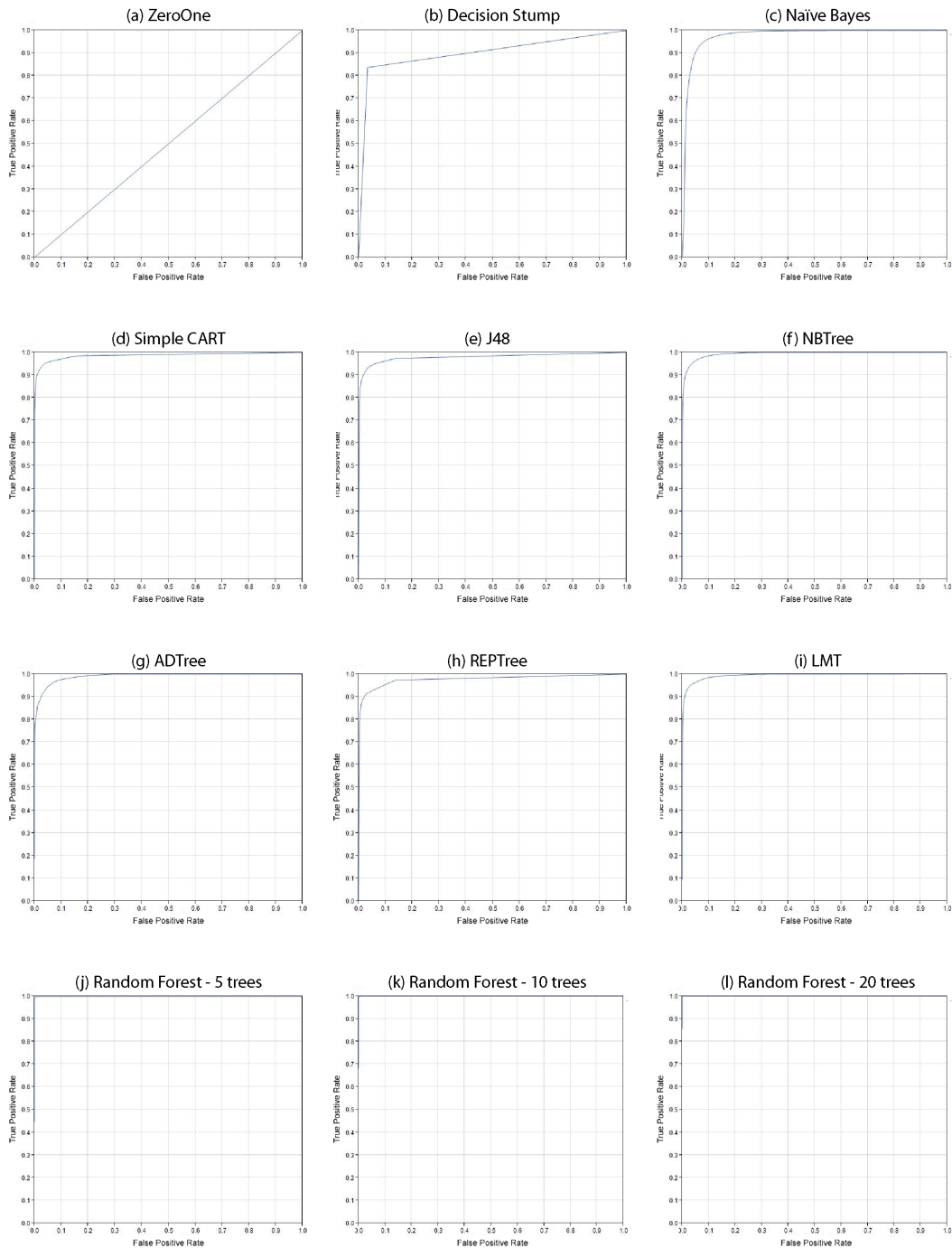


Figure 4.6: Examples of ROC charts for classification models from Table 4.2

negative classifications after preliminary experimentation with different misclassification costs. These values were found to be satisfactory at reducing the FN rates without too great a cost being incurred in the increase of FP misclassifications, as was the case with higher ratios of costs for FP and FN.

The results of performance of the various classification models using MetaCost are shown in Table 4.3. The ROC area curves for the MetaCost models are shown in Figure 4.7.

Table 4.3: Performance of Classification Models using MetaCost cost-sensitivity

Model	Accuracy	Precision	Recall	F-measure	ROC Area	Kappa	FP Rate	FN rate
J48	96.11%	0.965	0.961	0.962	0.957	0.8485	0.035	0.064
ADTree	91.42%	0.944	0.914	0.922	0.974	0.7127	0.096	0.022
NBTree	96.63%	0.97	0.966	0.967	0.99	0.8685	0.032	0.046
LMT	96.74%	0.971	0.967	0.968	0.991	0.8728	0.031	0.043
ZeroOne	85.95%	0.739	0.86	0.795	0.5	0	0	1
SMO	95.99%	0.96	0.96	0.96	0.913	0.8328	0.022	0.151
Multilayer	85.95%	0.739	0.86	0.795	0.5	0	0	1
REPTree	96.37%	0.965	0.964	0.964	0.941	0.855	0.028	0.085
DecisionStump	94.84%	0.949	0.948	0.949	0.902	0.7898	0.033	0.163
NaiveBayes	89.18%	0.934	0.892	0.903	0.971	0.6534	0.12	0.035
RF (5 trees)	98.98%	0.99	0.99	0.99	0.996	0.9593	0.011	0.004
RF (10 trees)	99.02%	0.991	0.99	0.99	0.997	0.9606	0.011	0.001
RF (20 trees)	99.08%	0.991	0.991	0.991	0.997	0.963	0.011	0
RF (50 trees)	99.09%	0.991	0.991	0.991	0.998	0.9632	0.011	0

### 4.3.1 Analysis of the effect of MetaCost on classification performance

The effect of applying the MetaCost wrapper to each classification model was assessed in terms of changes to the rates of false positive and false negative misclassifications. Figure 4.8 shows the FN rates for each classifier with and without the application of MetaCost. All the models show a significant reduction in FN rates, with the exception of Decision Stump, where the the FN rate reduction was insignificant and the RF models, where the FN rate was very low for 5 trees and zero for 10 and 20 trees. The most significant reduction was for ADTree, with an FN rate of 0.14 for the cost insensitive

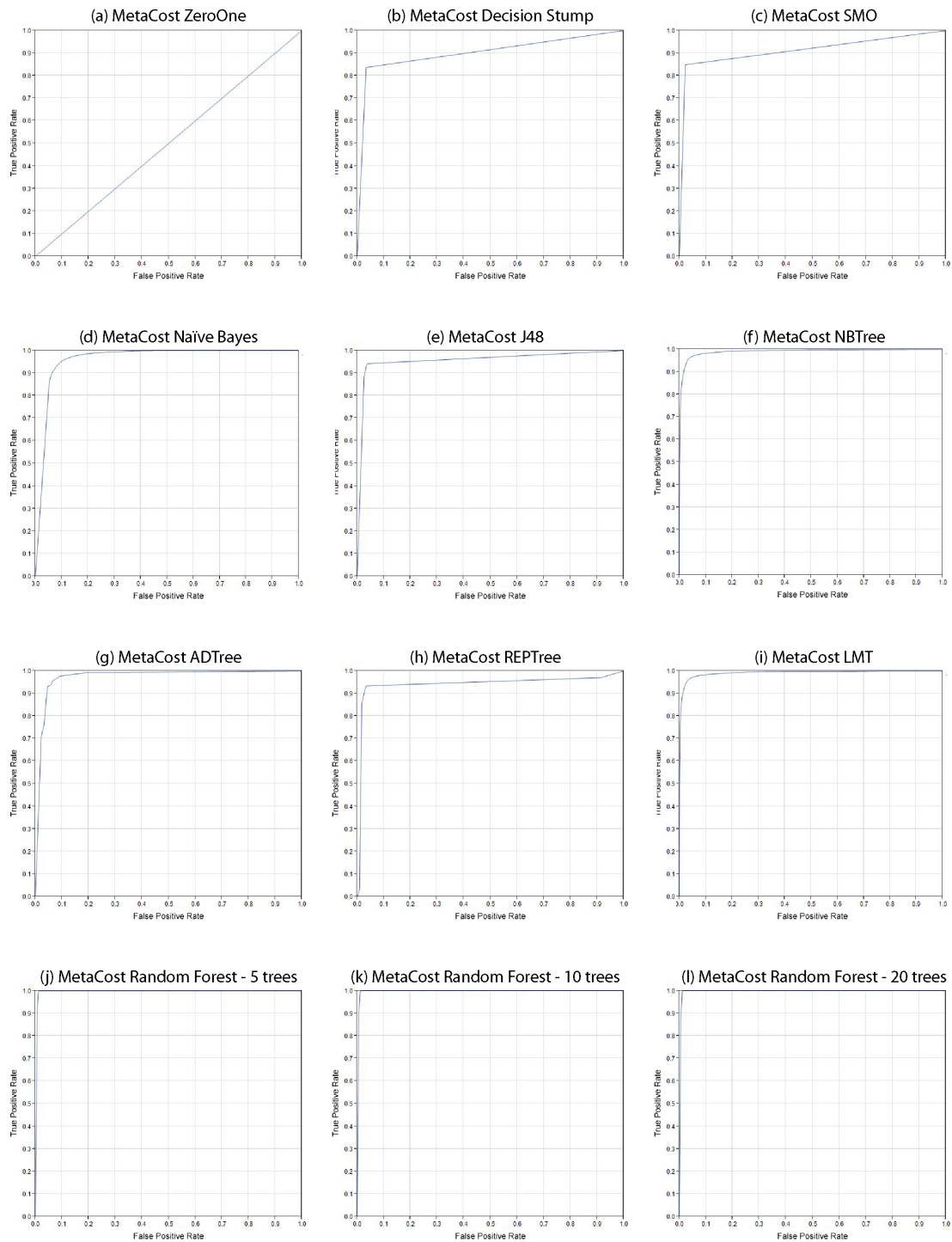


Figure 4.7: Examples of ROC charts for MetaCost models from Table 4.3

model and 0.022 using MetaCost.

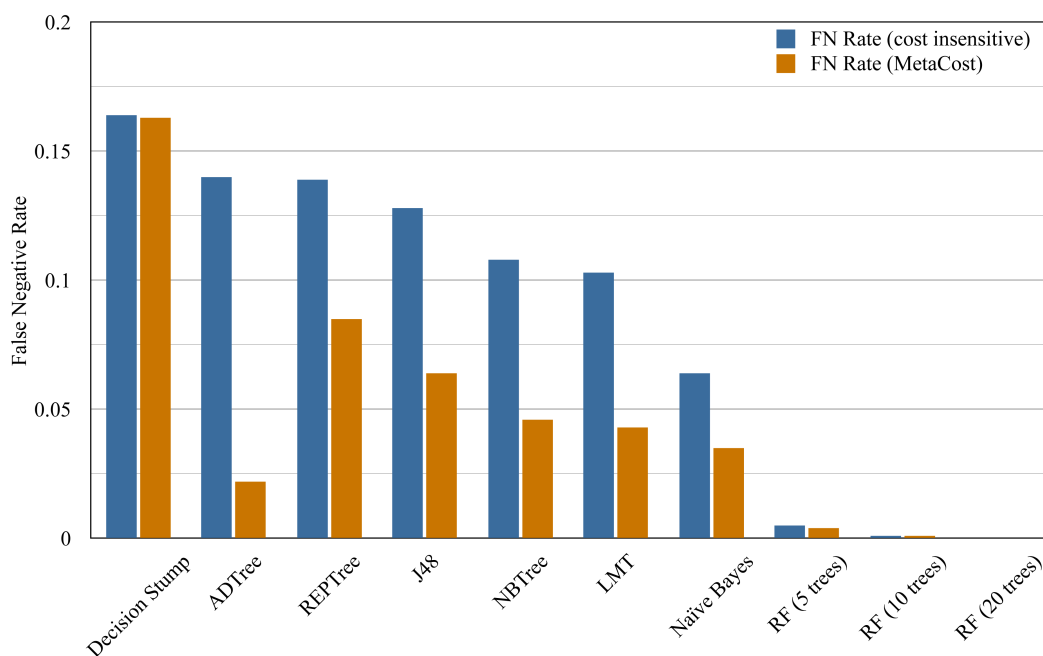


Figure 4.8: A comparison of false negative rates for cost-insensitive and MetaCost classification models

Figure 4.9 shows the FP rates for each classifier with and without the application of MetaCost. With the exception of Decision Stump, all models produced a higher FP rate when MetaCost was applied. Again, the ADTree classifier showed the largest increase in FP rate. All three RF models showed increased rates of FP but as the FN rates were close to zero for the cost-insensitive models, the application of MetaCost in these cases incurred a penalty in terms of increased FP rates without the benefit of a corresponding reduction in FN rates.

The overall effect of cost sensitivity is summarised in Figure 4.10 showing the change in FP and FN rates for each model.

## 4.4 Variable Importance

In the domain of structure identity and purity by LC-MS under investigation in this project, the two main variables that determine the outcome of an analysis are percent purity by UV and spectral purity by MS. It is these two values that are considered in the current system to label an analysis as pass or fail, with results that pass other criteria

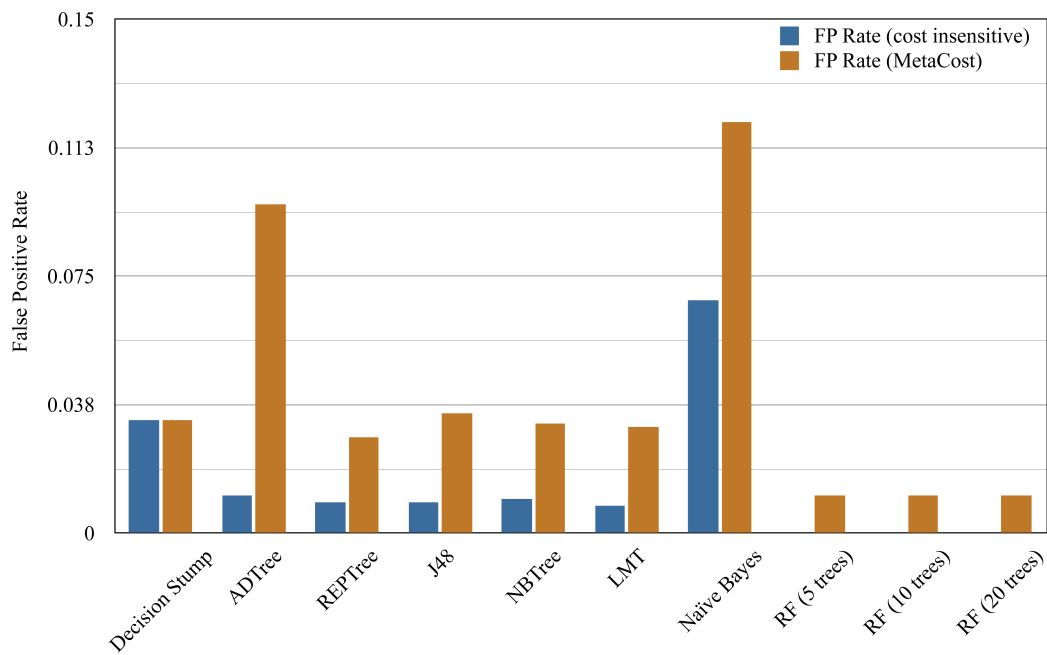


Figure 4.9: A comparison of false positive rates for cost-insensitive and MetaCost classification models

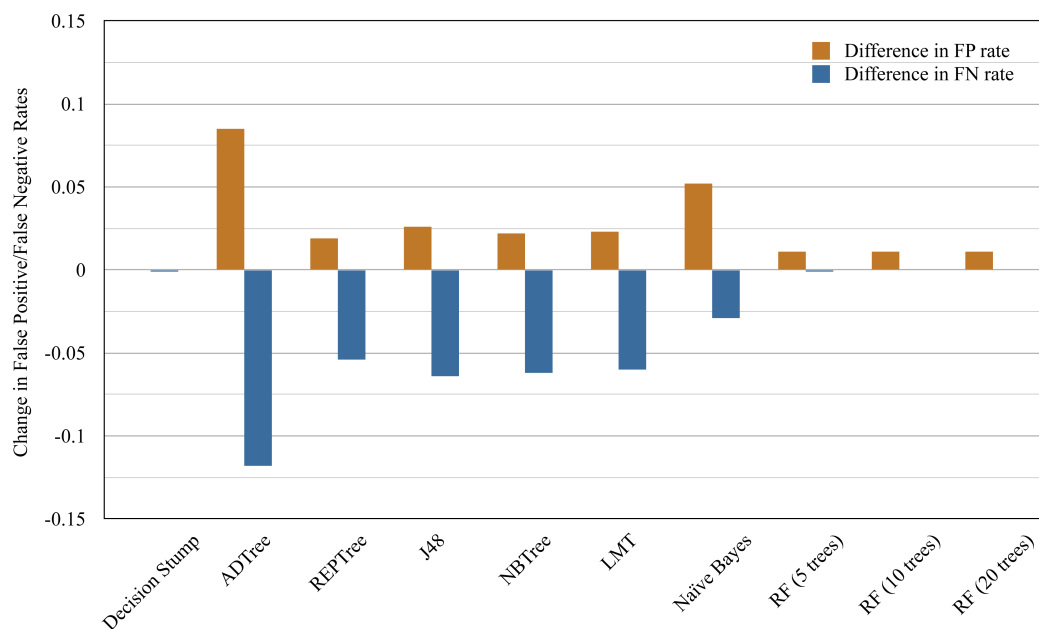


Figure 4.10: Changes in false positive and false negative rates for cost-insensitive and MetaCost classification models



for UV peak height, UV peak width and CAD peak area labelled as automatically passed (see Figure 2.4). Variable importance was assessed to establish if indeed these five criteria are the most powerful as predicting the class outcome of analysis events, or if any of the other attributes also make a contribution. Such measures of variable importance were investigated using two techniques; empirically using a measure of chi-squared using JMP software package and by applying an increasing amount of pre-pruning and measuring which attributes were eliminated from DTs. More important attributes remained as the amount of pre-pruning increased.

#### 4.4.1 Variable Importance using chi-squared

A classification and regression DT model was generated using JMP, similar to the SimpleCART implementation in WEKA. The DT generated is shown in Figure 4.11. The number of splits in which each variable was included, and the contribution measured using the  $G^2$  (likelihood-ratio chi-square), similar to that discussed in Section 3.3.2.5 on pruning. Table 4.4 lists each variable, ordered according to the  $G^2$  value in descending order. As expected, the two variables with the largest contribution to class outcome were *percent\_purity\_uv* (748030.80) and *horz\_max\_ms* (240759.26), with *get\_cad\_percent* also scoring highly (47612.42). In addition, *get\_closest\_peak* and *uv\_peak\_height* made significant contributions to classification, with *cad\_start*, *retention\_time* and *cad\_stop* also contributing to the partitioning of data. A notable finding in these results is the zero values for two of the criteria that are used to confirm an auto-pass in the current LC-MS system, namely *uv\_peak\_width* and *cad\_peak\_area*, suggesting that the inclusion of these two attributes in the auto-pass function is redundant.

#### 4.4.2 Variable Importance using Elimination by Pre-pruning

A second approach to establishing variable importance was performed by generating a number of J48 DTs using 10-fold cross validation. The level of pre-pruning was incrementally increased using the minimum number of instances per leaf node threshold, from 50 to 225 increasing by 25 each step. For each set the number of occurrences of each attribute were recorded across all folds. The results are summarised in Table 4.5.

As with the chi-squared assessment of variable importance in Section 4.4.2 the most important variables appear from these results to be *percent\_purity\_uv*, *horz\_max\_ms* and *get\_cad\_percent*, which all appeared in the DT for all folds and for all levels of pruning. The attributes *retention\_time* and *get\_closest\_peak* showed a reduction in the

Table 4.4: Variable Importance measured using Chi-squared

Attribute	Number of splits	Chi-squared
percent_purity_uv	7	748030.80
horz_max_ms	6	240759.26
get_cad_percent	8	47612.42
get_closest_peak	3	4271.20
uv_peak_height	1	2286.99
cad_start	2	1564.71
retention_time	1	889.83
cad_stop	1	569.35
num_of_peaks	0	0.00
uv_peak_width	0	0.00
cad_peak_area	0	0.00
cad_peak_width	0	0.00
dad_start	0	0.00
dad_stop	0	0.00

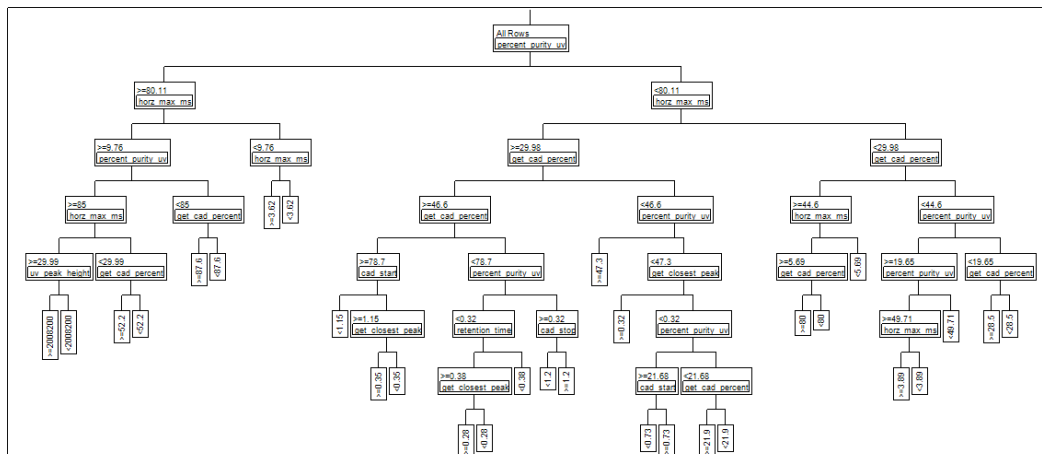


Figure 4.11: Classification and regression tree generated by JMP

Table 4.5: Performance of Classification Models

Attribute	MinNum=50	MinNum=75	MinNum=100	MinNum=125	MinNum=150	MinNum=175	MinNum=200	MinNum=225	MinNum=250
percent_purity_uv	10	10	10	10	10	10	10	10	10
horz_max_ms	10	10	10	10	10	10	10	10	10
get_cad_percent	10	10	10	10	10	10	10	10	10
retention_time	10	10	8	6	7	9	7	6	2
get_closest_peak	10	10	10	9	7	5	3	2	1
num_of_peaks	8	8	4	1	1	0	0	0	0
uv_peak_width	6	6	2	0	1	2	0	0	0
uv_peak_height	4	3	2	1	0	0	1	1	0
cad_peak_width	6	2	0	0	2	1	0	0	0
cad_peak_area	0	0	0	0	0	0	0	0	0

number of folds in which they occurred but both were present in all levels of pruning. No models contained a split using *cad\_peak\_area* which supports the assertion that this variable offers no value in determining the class outcome. Other results appear to contradict the chi-squared values calculated during the construction of the CART tree. Both *uv\_peak\_width* and *cad\_peak\_width* had a chi-squared value of zero but were included in some of the J48 models. They were however, eliminated by the time a minimum instances threshold of 200 had been applied, suggesting that the importance of these variables was low. As a cost complexity pruning method was applied to the CART tree in Section 4.4.2 it is assumed that any nodes where these two attributes were involved were pruned prior to the chi-squared values being calculated. From the earlier data on unpruned DTs it is apparent that whilst attributes that have little actual contribution to class outcomes will not be selected at the higher nodes in a decision tree, further down towards the bottom the amount of data on which attributes are selected for splitting is reduced. As a result, non-contributing attributes can by chance appear to be the preferred attribute on which to split. This may account for the inclusion of some attributes with zero chi-squared values when the amount of pruning is low.

### 4.4.3 Random Forest Implementation using Pipeline Pilot

From the initial experiments using the WEKA software platform, the highest performing classification model was found to be the RF algorithm (see Table 4.2). Further experiments were carried out using Pipeline Pilot to investigate the performance of the RF available in this software. The motivation for this was that much of the existing LC-MS system is implemented in Pipeline Pilot so the barrier to future integration with the current operational workflow would be low. In addition, the Pipeline Pilot model was capable of outputting confidence statistics on all class predictions. This feature was particularly desirable as it allows the partitioning of results into high confidence predictions (auto-pass and auto-fail) and low confidence predictions that require manual review, one of the aims of this project.

The maximum amount of available data, *Set\_Max* (Table 3.1) was used to train and test the RF models in Pipeline Pilot. Forests were constructed using 5, 10, 20, 50, 100 and 200 trees in the ensemble. The results of these experiments are shown in Table 4.6. All RF models showed high performance, with only the 5 tree model producing any false negative classifications and even then the FN rate was 0.0054. The remaining RF ensembles showed very little difference in performance when the number of trees was increased supporting the assertion that most of the gains in accuracy occur with the first few members of the ensemble [Leo01].

The RF models constructed using Pipeline Pilot were also assessed in terms of the confidence measures with which predictions of class were made. An example of the distribution of confidences from these RF models (10 tree version) is included in Appendix D. The data for all models is summarised in Figure 4.12, which shows the percentage of predictions that had a confidence value in excess of 90% (represented as  $>0.9$ ). A single-tree classification and regression model was included for comparison, labelled *RF\_TreeModel* in Figure 4.12. All RF models were able to make more confident predictions of class outcomes compared to the single tree model. The proportion of high confidence predictions declined as the number of trees in the ensemble was increased, with the 5 tree ensemble showing the highest proportion of high confidence predictions.

Table 4.6: Performance of Random Forest Models using Pipeline Pilot

Model	Accuracy	Precision	Recall	F-measure	FP Rate	FN Rate	TP	FP	TN	FN
RF_5Trees	95.99%	0.796	0.962	0.871	0.0347	0.0054	104,476	26,825	637,724	4,151
RF_10Trees	96.06%	0.774	1	0.873	0.039	0	104,446	30,483	638,247	0
RF_20Trees	96.07%	0.775	1	0.873	0.0393	0	104,458	30,389	638,329	0
RF_50Trees	96.08%	0.77	1	0.873	0.0392	0	104,471	30,334	638,371	0
RF_100Trees	96.09%	0.776	1	0.874	0.0391	0	104,442	30,208	638,526	0
RF_200Trees	96.09%	0.776	1	0.874	0.0391	0	104,438	30,216	638,522	0

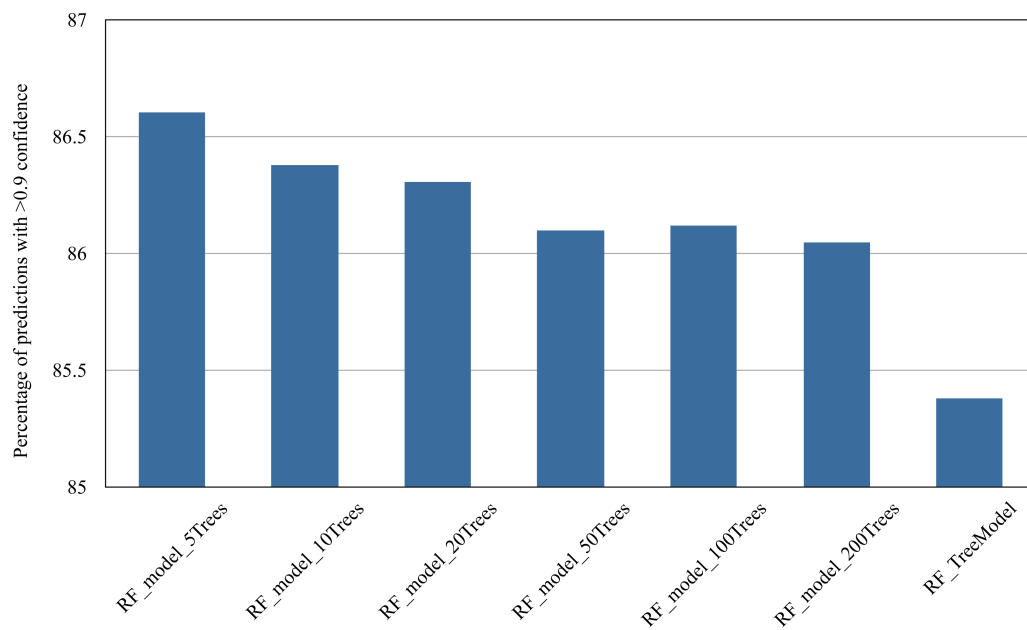


Figure 4.12: Effect of Random Forest tree number on the percentage of high confidence ( $>0.9$ ) predictions

# Chapter 5

## Discussion

### 5.1 Discussion of Results

Quality assurance activities on new compounds entering the screening collection at AstraZeneca are an essential activity to allow Compound Management to monitor, maintain and improve the overall quality of the screening collection. Low quality samples entering screening have the potential to disrupt hit identification projects by introducing spurious results. Lead identification compounds are generally discovered from a cluster of structurally similar, active compounds rather than a single hit, or singleton. Despite this, it is important that the samples under investigation match the structures against which they are registered into the corporate compound database. Similarly, impurities present in research compounds make the difficult job of establishing structure-activity relationships even more challenging.

As discussed in Chapter 2, the ideal scenario would be the existence of a universal method for identifying the structure and purity of research compounds and whilst the combination of LC and MS is a ubiquitous technique for addressing this need, it is not a panacea for compound quality. Although the majority of compounds can be successfully analysed using this approach, there are a small minority of structures where this method is less effective. Ideally, an analyte would show a single, well defined peak in the chromatography with high UV absorption and would be successfully ionised and detected by MS, forming one of the simple adduct patterns the spectrometer has been programmed to look for. Figure 2.2 shows the number of peaks integrated for analyses and confirms that this clear chromatographic separation of a single peak is rarely the case, with 5 integrated peaks being much more common. A typical LC-MS result from

this domain is less complex than in some other application of LC-MS, notably the various omics. In proteomics studies the integrated peaks can number in the hundreds, which then have to be matched to known ‘fingerprints’ often using reference databases [YI98], [BGMY04], [Sun04]. The comparative simplicity of the LC-MS results under investigation in this project is exposed by the fact that a brief review by a domain expert is often all that is required to confirm the outcome as a pass or fail. This situation is reflected in the data mining activities studied in this project. The domain of analysis by LC-MS is clearly ‘solvable’ using KDD and data mining, with even very simple rules such as Decision Stump showing high accuracy at predicting outcomes. It is, however, important to realise that as mentioned, most analyses produce multiple peaks and only one peak per analysis can by definition be considered a pass resulting in highly imbalanced data. This means that an obviously useless model that predicted all peaks as fails would be correct in approximately 85% of cases. The real battleground for the development of high performing classification models is therefore, not in the classification of the majority of cases, but for the outliers at the fringe where the situation is more complex and requires more sophisticated algorithms that can examine the subtleties of the data to make accurate predictions. This has been successfully achieved by including the additional derived attributes discussed in Section 3.2.3.6 when constructing classification models.

### 5.1.1 A Review of the Current Auto-pass Function

By far the most important attributes to consider are the percent purity by UV and the spectral purity by MS. This assumption was supported throughout this project by the construction of all classification models and by the investigations into variable importance discussed in Section 4.4. As described by Figure 2.4, it is these two critical attributes that dictate the initial labelling of results as passed or failed, subsequently confirmed or overwritten during the review stage. The original approach was to use three other attributes as ‘supporting evidence’ of a pass result (UV peak height, UV peak width and CAD peak area) with any analysis satisfying all five criteria labelled as an automatic pass. These tests were performed on the single peak most likely to represent a pass, identified by the ‘peak of interest’ function described in Section 3.2.3.5.

The philosophy applied to the original design of this process was one of caution. The inclusion of these five features in the application of an auto-pass label were thought to provide strong evidence that the results of the analysis confirmed the purity and structural identity of the analyte. In addition, the threshold values applied to these



features were set deliberately high, increasing the number of samples that would be diverted for manual review but lessening the risk of false positives. The algorithms generated in this project have revealed this to be the case, particularly with regard to the value of spectral purity, which must currently exceed 30% for a sample to be considered a pass. The classification models generated often partitioned data on a figure of approximately 8-11% spectral purity, with further splitting on other attributes in lower nodes. Figure 5.1, Figure 5.2 and Figure 5.3 all show extracts from the J48, J48 with bagging and REPTree models. The full decision trees are included in Appendix E. In all three cases, the split on spectral purity (*horz\_max\_ms*) uses a value of 9.38%, 8.17% and 8.93% respectively, with further splitting using other attributes further down the tree. When used in combination with these other attributes these much lower values for spectral purity were found to be optimal at accurately classifying the data, supporting the assertion that the value of 30% used historically was indeed overly conservative.

```

horz_max_ms > 9.38
|  |  percent_purity_uv <= 84.97
|  |  |  get_cad_percent <= 87.1
|  |  |  |  percent_purity_uv <= 83.3: fail (1578.0/619.0)
|  |  |  |  percent_purity_uv > 83.3: pass (595.0/240.0)
|  |  |  get_cad_percent > 87.1: pass (673.0/210.0)
|  |  percent_purity_uv > 84.97: pass (37622.0/711.0)

```

Figure 5.1: Extract from J48

```

percent_purity_uv > 79.02
|  horz_max_ms <= 8.17: fail (7902.0/665.0)
|  horz_max_ms > 8.17
|  |  percent_purity_uv <= 84.97
|  |  |  get_cad_percent <= 86.7
|  |  |  |  percent_purity_uv <= 83.22: fail (1442.0/563.0)
|  |  |  |  percent_purity_uv > 83.22: pass (619.0/253.0)
|  |  |  get_cad_percent > 86.7: pass (681.0/220.0)
|  |  percent_purity_uv > 84.97: pass (37833.0/786.0)

```

Figure 5.2: Extract from J48 with Bagging

What is perhaps more interesting is that the most important variables that were found to contribute to the classification were not the five originally used in the auto-pass function, but were some of the derived attributes; *get\_cad\_percent* and *get\_closest\_peak* (see Section 3.2.3.6 and Table 4.4). Although these attributes are applied to each peak

```
percent_purity_uv >= 79.02
|  horz_max_ms < 8.93 : fail (8064/712) [3951/396]
|  horz_max_ms >= 8.93
|  |  percent_purity_uv < 85
|  |  |  get_cad_percent < 87.25 : fail (2142/976) [993/450]
|  |  |  get_cad_percent >= 87.25 : pass (643/201) [335/101]
|  |  percent_purity_uv >= 85 : pass (37725/749) [18867/395]
```

Figure 5.3: Extract from REPTree

individually, they both have the characteristic of placing each peak ‘in the context’ of the other peaks in an analysis event.

The inclusion of these attributes was a recommendation following early discussions with domain experts. The attribute *get\_cad\_percent* is an alternative representation of the value of *cad\_peak\_area*, expressed as a percentage of the total CAD area accounted for by each peak rather than using arbitrary units which may vary between different instruments. The results show that this was indeed a powerful attribute, with only *percent\_purity\_uv* and *horz\_max\_ms* having higher chi-squared measures of variable importance (Table 4.4).

The attribute *get\_closest\_peak* was included in an attempt to identify a particular type of issue. The LC-MS instruments perform the task of identifying each eluted peak when processing raw data. Although this is generally successful, sometimes a slight aberration or ‘shoulder’ during chromatography can cause the instrument to incorrectly divide a single peak into two. This effect is described in the simplified representation of a chromatogram in Figure 5.4. The *get\_closest\_peak* attribute measures the distance in retention time from each peak to the closest peak, eluted either before or after in the chromatogram and was used to try to identify these types of integration problems. The chi-squared value for this attribute suggests that it was successful in doing this, with a large *get\_closest\_peak* increasing the likelihood that a peak was predicted as a fail.

In conclusion, the application of the data mining approach investigated in this study negates the need to identify the ‘peak of interest’, as it is applied to every peak, and increases the predictive capability of the current system for classifying the outcome of analyses.

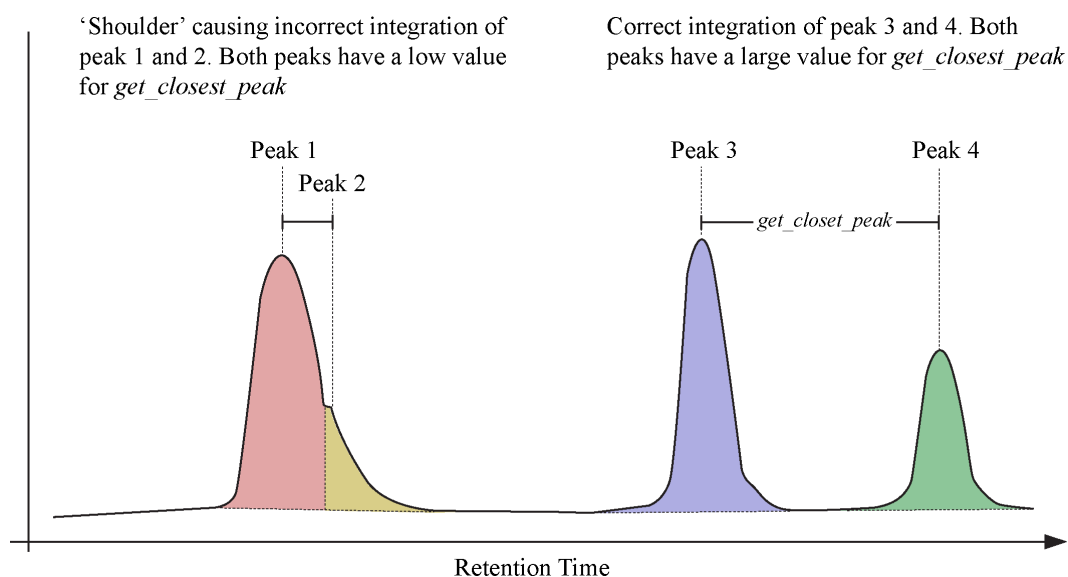


Figure 5.4: Detection of peak integration errors using the *get\_closest\_peak* attribute

### 5.1.2 Application of Classification Models into the LC-MS Workflow

The highest performing classification model investigated in this project was the Random Forest, with an accuracy close to 100% when using 20 and 50 tree ensembles. A confidence measure on the predictions made by a Random Forest model will allow a new workflow system to automatically label analytical results as pass or fail with far fewer analyses requiring a manual review. As predictions are made on each individual peak, the results of these peaks could be aggregated into an overall result for the analysis. For example, if a single peak from an analysis has a predicted pass with a high confidence, the analysis can be labelled as passed. If all peaks are predicted fails with a high confidence the analysis can be labelled as failed. If any of the peaks have predictions where the confidence level is low, these analyses can be flagged as requiring a manual review by an LC-MS technician. Preliminary analysis using a Random Forest model in Pipeline Pilot show the results of this strategy on the data currently stored in the operational LC-MS database (Section 4.4.3).

In order to assess the impact of a system as described above an analysis was performed on the test data used to evaluate the Random Forest (10 trees) in Pipeline Pilot. All 773,176 peaks were grouped into their respective samples, producing 175,893

unique analyses. Various confidence thresholds, referred to as *min\_conf*, were investigated to test the effect on the number of analyses that would require a manual review. Each group was then tested to see if any 'pass' predictions were present. If found the samples were labelled as *auto-pass* if the confidence on pass predictions was  $>min\_conf$  and *pass-requires-review* if  $<min\_conf$ . For samples where all peaks were predicted as failed, the value of the prediction with the lowest confidence was tested, with  $>min\_conf$  labelled as *auto-fail* and those with at least one prediction where the confidence was  $<min\_conf$  labelled as *fail-requires-review*. The results are summarised in Figure 5.5.

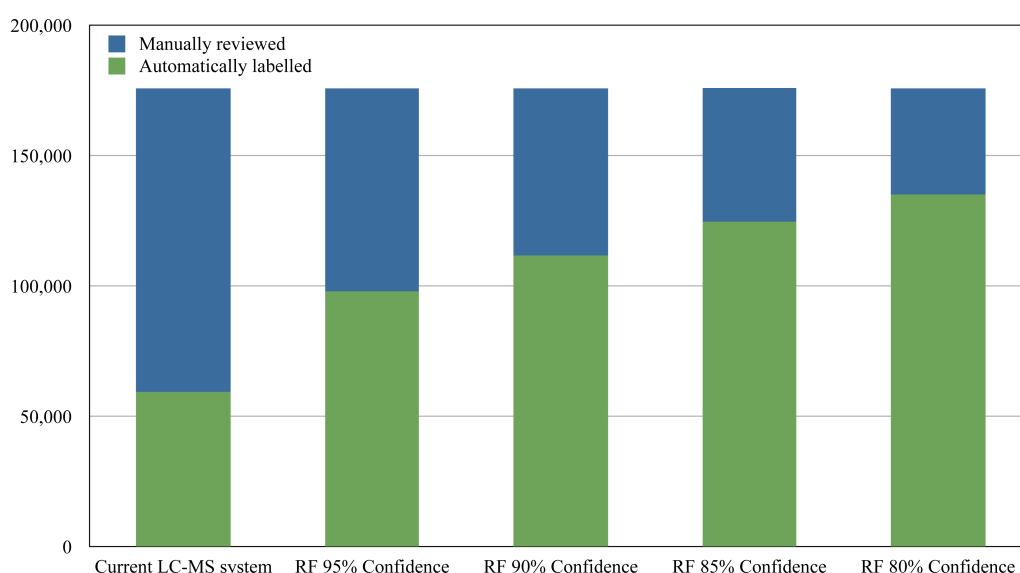


Figure 5.5: Comparison of automatic and manual reviews for current LC-MS system versus Random Forest approach

For comparison, the current LC-MS system was able to automatically pass 59,497 analyses with the remaining 116,396 (66.2%) requiring a manual review. When the RF model is applied to the LC-MS test data, the number of analyses requiring a manual review drops to 77,840 (44.3%) with *min\_conf* set at 95% and 40,688 (23.2%) when *min\_conf* is relaxed to 80%. Following discussions with domain experts, a figure of 90% for *min\_conf* was considered a suitable compromise between accuracy and a reduction in the number of manual reviews. This figure could be adjusted to suit the requirements of specific LC-MS processes, however, as the importance of accuracy may vary for different activities.

When *min\_conf* was set to 90% the number of analyses requiring manual review

dropped to 64,087 (36.4%), a 45% reduction in the number of manual reviews. Within the 111,806 analyses automatically passed there would have been no FN misclassifications and only 2,848 (2.5%) FP results. Clearly this is a significant reduction in the amount of resource required in terms of reviews by expert LC-MS technicians. For the data included in this test set alone, 52,309 analyses would have been removed from the manual review process.

#### 5.1.2.1 Quantifying the Effect of RF Models on CM Resources

The introduction of the current LC-MS system in CM UK, AstraZeneca allowed a stepwise increase in throughputs compared to the previous manual workflow, which offered no opportunities to automatically pass or fail results. The number of samples analysed per month reached 10,000 after the first three months of operation. The past few years have seen a steady rise in throughputs with the number of analyses per month now averaging around 25,000. This rise may be due to several factors, including enhancements to the LC-MS system, familiarity of users with the review GUI and an increase in knowledge and experience of some users. Most of the reviewing has been carried out by two domain experts, with some assistance from other staff. When combined, the estimated time spent performing these reviews is 15 hours per week, or 40% of a full-time employee (FTE). This estimate is based on a domain expert with in-depth knowledge and experience of how to interpret LC-MS analytical data. The figure of 15 hours could be considerably higher for less experienced LC-MS technicians. The application of the RF Model using a confidence threshold of 90% on predictions would reduce this time from 15 hours to 8.25 hours, or from 40% of one FTE to 22%. The total cost of an AstraZeneca FTE is estimated to be \$250,000 per year. The recommendations from this project would therefore equate to an annual saving of \$45,000. Alternatively, if the same amount of resource was assigned to the manual review process, the current number of analyses per month could potentially rise from 25,000 to over 45,000 due to the reduced number of analyses entering the manual review process.

Aside from the savings in resources, other benefits are expected resulting from the implementation of the RF model. The manual review process requires a high level of concentration and the repetitive nature of this work can lead to fatigue and occasional mistakes in interpretation even for the most experienced technicians. As discussed in Section 3.3.4.3, Random Forests are robust in terms of outliers as anomalies in a particularly DT are often 'outvoted' by the other trees in the ensemble. The RF model could therefore be expected to outperform domain experts and reduce the number of

misclassifications on future data.

### 5.1.3 Data Mining as a Complimentary Approach to Quality Assurance

The main objective of quality assurance by LC-MS performed by CM is to confirm the quality of compounds in the corporate collection. The data gathered by these activities has also been used for more investigative research into compound stability. As well as the binary decision on quality, which has been the focus of this project, other annotations are added to LC-MS results, such as the mass identified if found to be different than that expected. This has allowed work on degradation patterns to be investigated which has looked for links between particular chemical structures and instability in DMSO [CSA], [CS09]. The classification of LC-MS results using the data mining methods discussed are extremely useful for automating the routine quality assessment of compounds but are not capable of identifying the degradation patterns identified by domain experts during data review. As such, the data mining approach to classification should be considered a complementary system rather than a complete replacement of the current system. Instead, it presents an opportunity to split LC-MS activities into a two-tier approach. These two distinct activities are described below:

- Tier 1: Routine quality assurance activities to confirm the structural identity and purity of research compounds - these activities will utilise a Random Forest classification model as reported by this project to provide a binary decision on purity of compounds by LC-MS
- Tier 2: Experiments on compound instability - LC-MS analysis will be performed on targeted sets of compounds and reviewed manually to investigate degradation patterns such as redox reactions. This will involve a manual review by a domain expert to annotate results with the mass found and therefore allow an investigation into specific chemical degradation.

By decoupling these two distinct processes, the routine activities in tier 1, which account for the vast majority of LC-MS activities, can be further automated freeing up resources for tier 2 activities.

### 5.1.4 Further Work

The predictions of LC-MS outcomes sought in this project are binary decisions on pass or fail. As discussed previously this does not capture the full value of LC-MS activities carried out by the UK Compound Management at AstraZeneca and the automating of this process without the need for expert review as described does come at a cost; the binary classification models are not capable of distinguishing the different reasons for failures. There are, however, some steps that can be taken to address this issue. For example, if an analysis was labelled as an *auto-fail* it could easily be checked to see if any of the peaks for that analysis had a high value for purity by UV. By applying this test, it would be possible to make a distinction between analyses where something was found but it was not the correct mass and analyses where no substance was detected. Similarly, data for *auto-fail* analyses could be examined to identify instances when the analysis was not performed at all due to errors with the instruments. If there were no integrated peaks or if all peaks were of particularly low quality (very low spectral purity) this may indicate that the analysis itself has failed, a different situation from the analysis being performed and producing a *fail* outcome. Rather than being flagged as requiring a manual review, these analyses could be marked as erroneous results, triggering a second analysis of the sample. This approach would further reduce the number of analyses requiring manual review.

The next challenge in this domain is to not only be able to answer the question “was the expected compound present?” but rather “from the analytical data, what was the compound present?”. Previous activities by CM staff have suggested that many failed LC-MS analyses are due to chemical changes and degradations in DMSO, such as oxidation or hydrolysis [CSA]. The results of LC-MS analyses can often be indicative of these types of reactions, producing patterns that are recognisable to domain experts. By applying a KDD/data mining approach to data which further describes both the chemical characteristics of the compounds being tested and the analytical results of LC-MS, it may be possible to develop models capable of identifying compounds that have degraded by reconstructing a value for molecular mass from the patterns of degradation seen in LC-MS results. A simple example to illustrate this point would be to check failed analyses for the presence of halogens such as chlorine or bromine. Such elements produce recognisable patterns in LC-MS data following a degradation that could help to identify what the compound was before the reaction took place. Being able to identify structure-stability relationships at the molecular level may allow a data mining approach to be developed that is capable of answering the question “what

was the compound present". Success in this endeavour would combine the ability of data mining to automate LC-MS interpretations and the valuable information on types of failures, currently only available through the manual review process. This would allow the two tiers described in Section 5.1.3 to be merged into a highly automated, high performance system that would maximise the value of data gleaned from quality assurance by LC-MS.



# Bibliography

- [AMBC04] A Amo, J Montero, G Biging, and V Cutello. Fuzzy classification systems. *European Journal of Operational Research*, 156(2):495–507, July 2004.
- [BFOS84] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. wadsworth & brooks. *Monterey, CA*, 1984.
- [BGMY04] M Bern, D Goldberg, W H McDonald, and J R Yates. Automatic Quality Assessment of Peptide Tandem Mass Spectra. *Bioinformatics*, 20(Suppl 1):i49–i54, July 2004.
- [Bre96a] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [Bre96b] Leo Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996.
- [BTBa01] R Burbidge, M Trotter, B Buxton, and et al. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Computers & chemistry*, 2001.
- [CLM08] A. Cufoglu, M. Lohi, and K. Madani. Classification accuracy performance of Naïve Bayesian (NB), Bayesian Networks (BN), Lazy Learning of Bayesian Rules (LBR) and Instance-Based Learner (IB1)-comparative study. *Computer Engineering & Systems, 2008. ICCES 2008. International Conference on*, pages 210–215, 2008.
- [CPL07] G Chen, BN Pramanik, and YH Liu. Applications of LC/MS in structure identifications of small molecules and proteins in drug discovery

- Chen - 2007 - Journal of Mass Spectrometry - Wiley Online Library. ... *of mass spectrometry*, 2007.
- [CS09] Isabel Charles and Ian Sinclair. Journal of Biomolecular Screening. *Journal of Biomolecular Screening*, pages 1–8, June 2009.
- [CSA] Isabel Charles, Ian Sinclair, and Daniel Addison. The Capture and Exploration of Sample Quality Data to Inform and Improve the Management of a Screening Collection. *Journal of Laboratory Automation*, (Publication Pending).
- [CV05] Mario Cannataro and Pierangelo Veltri. SpecDB: a database for storing and managing mass spectrometry proteomics data. In *WILF'05: Proceedings of the 6th international conference on Fuzzy Logic and Applications*. Springer-Verlag, September 2005.
- [DAa07] K Dettmer, PA Aronov, and et al. Mass spectrometrybased metabolomics. *Mass Spectrometry ...*, 2007.
- [Dea82] Michael AB Deakin. The development of the laplace transform, 1737–1937 ii. poincaré to doetsch, 1880–1937. *Archive for History of Exact Sciences*, 26(4):351–381, 1982.
- [Dom99a] P Domingos. MetaCost. In *Proceedings of the fifth ACM SIGKDD international ...*, 1999.
- [Dom99b] Pedro Domingos. MetaCost: a general method for making classifiers cost-sensitive. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Request Permissions, August 1999.
- [DVMWK99] BD Duléry, J Verne-Mismer, E Wolf, and C Kugel. Analyses of compound libraries obtained by high-throughput parallel synthesis: strategy of quality control by high-performance liquid chromatography, mass spectrometry and nuclear magnetic resonance techniques 10.1016/S0378-4347(98)00570-2 : *Journal of Chromatography B: Biomedical Sciences and Applications* — ScienceDirect.com. ... *of Chromatography B: ...*, 1999.

- [EB00] PJ Eddershaw and AP Beresford. ScienceDirect.com - Drug Discovery Today - ADME/PK as part of a rational approach to drug discovery. *Drug discovery today*, 2000.
- [EGJ09] Daniel Eriksson, Sven Glansberg, and Johan Jörtsö. Cost-sensitive Classifiers. 2009.
- [FCL<sup>+</sup>03] Eliza N Fung, Inhou Chu, Cheng Li, Tongtong Liu, Anthony Soares, Richard Morrison, and Amin A Nomeir. Higher-throughput screening for Caco-2 permeability utilizing a multiple sprayer liquid chromatography/tandem mass spectrometry system. *Rapid communications in mass spectrometry : RCM*, 17(18):2147–2152, 2003.
- [FM99] Y Freund and L Mason. The alternating decision tree learning algorithm. *MACHINE LEARNING-INTERNATIONAL . . .*, 1999.
- [FPF10] F Famili, S Phan, and F Fauteux. Data integration and knowledge discovery in life sciences. *IEA/AIE'10 . . .*, 2010.
- [FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases. *AI magazine*, 17(3):37, March 1996.
- [FS] Y Freund and R E Schapire. 1 996–Experiments with a New Boosting Algorithm. pages 148–156.
- [GA03] Leland J Gershell and Joshua H Atkins. A brief history of novel drug discovery technologies. *Nature reviews. Drug discovery*, 2(4):321–327, April 2003.
- [Gam04] J Gama. Functional trees. *Machine Learning*, 55(3):219–250, 2004.
- [GHGa99] SP Gygi, DKM Han, AC Gingras, and et al. Protein analysis by mass spectrometry and sequence database searching: Tools for cancer research in the postgenomic era. . . ., 1999.
- [GV03] Gary L Glish and Richard W Vachet. The basics of mass spectrometry in the twenty-first century. *Nature reviews. Drug discovery*, 2(2):140–150, February 2003.

- [GW08] Guangtao Ge and G William Wong. Classification of premalignant pancreatic cancer mass-spectrometry data using decision tree ensembles. *BMC Bioinformatics*, 9(1):275, 2008.
- [HBB<sup>+</sup>08] John G Houston, Martyn N Banks, Alastair Binnie, Stephen Brenner, Jonathan O’Connell, and Edward W Petrillo. Case study: impact of technology investment on lead discovery at Bristol-Myers Squibb, 1998-2006. *Drug discovery today*, 13(1-2):44–51, January 2008.
- [Hol93] Robert C Holte. Machine Learning, Volume 11, Number 1 - Springer-Link. *Machine Learning*, 11(1):63–90, 1993.
- [IXC02] J Isbell, R Xu, and Z Cai. Realities of High-Throughput Liquid Chromatography/Mass Spectrometry Purification of Large Combinatorial Libraries: A Report on Overall Sample Throughput Using Parallel Purification - Journal of Combinatorial Chemistry (ACS Publications). . . . of *Combinatorial Chemistry*, 2002.
- [JCSa00] KH Jarman, ST Cebula, AJ Saenz, and et al. An algorithm for automated bacterial identification using matrix-assisted laser desorption/ionization mass spectrometry. *Analytical . . .*, 2000.
- [JDIFx09] Chen Jin, Luo De-lin, and Mu Fen-xiang. An improved ID3 decision tree algorithm. In *Computer Science & Education, 2009. ICCSE '09. 4th International Conference on*, pages 127–130, 2009.
- [Kas80] G V Kass. JSTOR: Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 29, No. 2 (1980), pp. 119-127. *Applied statistics*, 1980.
- [KCW01] JN Kyranos, H Cai, and D Wei. High-throughput high-performance liquid chromatography/mass spectrometry for modern drug discovery 10.1016/S0958-1669(00)00176-2 : Current Opinion in Biotechnology — ScienceDirect.com. *Current opinion in . . .*, 2001.
- [KD95] E B Kong and T G Dietterich. Error-correcting output coding corrects bias and variance. . . . of *the Twelfth International Conference on Machine . . .*, 1995.

- [Ken04] JR Kenseth. High-throughput characterization and quality control of small-molecule combinatorial libraries 10.1016/j.cbpa.2004.06.004 : Current Opinion in Chemical Biology — ScienceDirect.com. *Current opinion in chemical biology*, 2004.
- [Kor05] WA Korfmacher. Foundation review: Principles and applications of LC-MS in new drug discovery 10.1016/S1359-6446(05)03620-2 : Drug Discovery Today — ScienceDirect.com. *Drug discovery today*, 2005.
- [KV99] WA Korfmacher and J Veals. Demonstration of the capabilities of a parallel high performance liquid chromatography tandem mass spectrometry system for use in the analysis of drug discovery plasma samples - Korfmacher - 1999 - Rapid Communications in Mass Spectrometry - Wiley Online Library. *Mass Spectrometry . . .*, 1999.
- [KWGa97] M Kamber, L Winstone, W Gong, and et al. Generalization and decision tree induction: efficient classification in data mining. . . . *Issues in Data . . .*, 1997.
- [KYO03] HL Koh, WP Yau, and PS Ong. Current trends in modern pharmaceutical analysis for drug discovery 10.1016/S1359-6446(03)02846-0 : Drug Discovery Today — ScienceDirect.com. *Drug discovery today*, 2003.
- [Leo01] Breiman Leo. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [LHF05] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic Model Trees. *Machine Learning*, 59(1-2):161–205, May 2005.
- [LS97] Wei-Yin Loh and Yu-Shan Shih. Split Selection Methods for Classification Trees. *Statistica Sinica* 7(1997), 815-840, 1997.
- [Min89] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243, 1989.
- [MO11] R Maclin and D Opitz. Popular Ensemble Methods: An Empirical Study. *arXiv.org*, June 2011.

- [MS06] Timothy R McJunkin and Jill R Scott. Fuzzy Logic Classification of Imaging Laser Desorption Fourier Transform Mass Spectrometry Data. *arXiv.org*, November 2006.
- [MSKW02] Larry M Mallis, Ani B Sarkahian, John M Kulishoff, and William L Watts. Open-access liquid chromatography/mass spectrometry in a drug discovery environment. *Journal of mass spectrometry : JMS*, 37(9):889–896, September 2002.
- [MT03] MK Markey and GD Tourassi. Decision tree classification of proteins identified by mass spectrometry of blood serum samples from people with and without lung cancer - Markey - 2003 - PROTEOMICS - Wiley Online Library. *Proteomics*, 2003.
- [OS84] LBJHFRA Olshen and C J Stone. Classification and Regression Trees. *Wadsworth . . .*, 1984.
- [OS96] David W Opitz and Jude W Shavlik. Generating accurate and diverse members of a neural-network ensemble. *Advances in neural information processing systems*, pages 535–541, 1996.
- [PS01] DI Papac and Z Shahrokh. Mass spectrometry innovations in drug discovery and development. *Pharmaceutical Research*, 18(2):131–145, 2001.
- [QCJ95] J Quinlan and R Cameron-Jones. Oversearching and layered search in empirical learning. *breast cancer*, 286:2–7, 1995.
- [QJL<sup>+</sup>06] W J Qian, J M Jacobs, T Liu, D G Camp, and R D Smith. Advances and Challenges in Liquid Chromatography-Mass Spectrometry-based Proteomics Profiling for Clinical Applications. *Molecular & Cellular Proteomics*, 5(10):1727–1744, May 2006.
- [Qui86] JR Quinlan. Induction of Decision trees. *Machine Learning, Volume 1, Number 1 - SpringerLink*, 1986.
- [Qui96a] J R Quinlan. Improved Use of Continuous Attributes in C4.5. *arXiv.org*, March 1996.
- [Qui96b] J Ross Quinlan. Bagging, boosting, and C4. 5. pages 725–730, 1996.

- [RŠ04] M Robnik-Šikonja. Improving Random Forests - Springer. *Machine Learning: ECML 2004*, 2004.
- [RŠK03] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69, 2003.
- [Sa06] H Shin and et al. A machine learning perspective on the development of clinical decision support systems utilizing mass spectra of blood samples. *Journal of Biomedical Informatics*, 2006.
- [SG09] Ian Sinclair and Richard Gallagher. Charged Aerosol Detection: Factors for consideration in its use as a generic quantitative detector. *Chromatography Today*, 1(3):1–5, October 2009.
- [Shi01] YG Shin. Analysis and screening of combinatorial libraries using mass spectrometry - Shin - 2002 - Biopharmaceutics & Drug Disposition - Wiley Online Library. *Biopharmaceutics & drug ...*, 2001.
- [Shi07a] H Shi. Best-first decision tree learning. 2007.
- [Shi07b] Haijian Shi. Best-first decision tree learning. 2007.
- [SMF<sup>+</sup>06] J Salmi, R Moulder, J J Filen, O S Nevalainen, T A Nyman, R Lahesmaa, and T Aittokallio. Quality classification of tandem mass spectrometry data. *Bioinformatics*, 22(4):400–406, February 2006.
- [SSQ<sup>+</sup>07] Yahui Su, Jing Shen, Honggang Qian, Huachong Ma, Jiafu Ji, Hong Ma, Longhua Ma, Weihua Zhang, Ling Meng, Zhenfu Li, Jian Wu, Genglin Jin, Jianzhi Zhang, and Chengchao Shou. Diagnosis of gastric cancer using decision tree classification of mass spectral data. *Cancer Science*, 98(1):37–43, January 2007.
- [Sti84] B Stine. Classification and regression trees. 1984.
- [Sun04] W Sun. AMASS: Software for Automatically Validating the Quality of MS/MS Spectrum from SEQUEST Results. *Molecular & Cellular Proteomics*, 3(12):1194–1199, September 2004.

- [Süß99] RD Süßmuth. Impact of mass spectrometry on combinatorial chemistry 10.1016/S0378-4347(98)00513-1 : Journal of Chromatography B: Biomedical Sciences and Applications — ScienceDirect.com. *Journal of Chromatography B: Biomedical Sciences . . .*, 1999.
- [Swe02] Jonathan Sweedler. The continued evolution of hyphenated instruments. *Analytical and Bioanalytical Chemistry*, 373(6):321–322, July 2002.
- [TKF<sup>+</sup>03] GK Taylor, YB Kim, AJ Forbes, F Meng, and et al. Web and database software for identification of intact proteins using “top down” mass spectrometry. *Analytical . . .*, 2003.
- [Wa05] IH Witten and et al. Data Mining: Practical machine learning tools and techniques. *books.google.com*, 2005.
- [WAF<sup>+</sup>03] B Wu, T Abbott, D Fishman, W McMurray, and et al. Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. . . ., 2003.
- [WF05] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.
- [WFT<sup>+</sup>99] Ian H. Witten, Eibe Frank, Leonard E. Trigg, Mark A. Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with Java implementations. 1999.
- [Wil92] L Wilkinson. Tree structured data analysis: AID, CHAID and CART. In *Proceedings of Sawtooth; Software Conference*, 1992.
- [Wol92] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.
- [WR11] Chris Walsh and Stuart Ratcliffe. Automated Compound Management Systems: Maximizing Return on Investment Through Effective Life-Cycle Planning. *Journal of Laboratory Automation*, 16(5):387–392, October 2011.
- [XNJR02] R Xu, C Nemes, KM Jenkins, and RA Rourick. Application of parallel liquid chromatography/mass spectrometry for high throughput



- microsomal stability screening of compound libraries 10.1016/S1044-0305(01)00342-7 : Journal of the American Society for Mass Spectrometry — ScienceDirect.com. *Journal of the American ...*, 2002.
- [Yat03] I Yates. Compound Management comes of age. *Drug Discovery*, 2003.
- [YFIZ03] B Yan, L Fang, M Irving, and S Zhang. Quality Control in Combinatorial Chemistry: Determination of the Quantity, Purity, and Quantitative Purity of Compounds in Combinatorial Libraries - Journal of Combinatorial Chemistry (ACS Publications). ... *of combinatorial ...*, 2003.
- [YI98] JR Yates III. Database searching using mass spectrometry data. *Electrophoresis*, 1998.
- [YI00] JR Yates III. Mass spectrometry:: from genomics to proteomics. *Trends in Genetics*, 2000.
- [YR09] JR Yates and CI Ruse. Proteomics by mass spectrometry: approaches, advances, and applications. *Annual review of ...*, 2009.
- [ZZ07] Y Zhao and Y Zhang. Comparison of decision tree methods for finding active objects. *arXiv.org*, August 2007.

# Appendix A

## Techniques and Components of LC-MS Analysis

### A.1 Liquid Chromatography (LC)

Separation by LC instrumentation has seen a number of technological advances, which have increased both sensitivity, resolution and the speed of analysis. These improvements have been achieved primarily by reducing the size of particles packed into the column, reducing the inner diameter of the column itself and by applying a very high pressure (10-20 kp.s.i.) to the system [QJL<sup>+</sup>06]. Whilst the use of high pressures create additional technical challenges for creating robust automated LC systems, some such ‘ultra-performance’ (UPLC) instruments are now commercially available, for example the Waters ACQUITY™UPLC System (1.7  $\mu\text{m}$  sized particles operating at >10 kp.s.i.). UPLC systems are capable of high resolution, sensitivity and peak capacity and allow increased throughputs and good levels of integration. They are applicable to both compound library identification and separation of complex mixtures for metabolomics and biomarker analysis [CPL07]. All instrument configurations utilised by CM include a liquid chromatography column.

### A.2 Diode Array Detection (DAD)

DAD is a method of UV detection and is commonly used for determining purity of compound libraries. Purity is defined as the percentage by weight of the expected target compound in the sample and is calculated by comparing the area of an integrated peak as a percentage of the total area of all integrated peaks in a UV chromatogram

[YFIZ03]. All other substances detected are present as impurities. There are a number of issues around the use of DAD. The purity is a percentage of the total detection by UV. The method also assumes that all substances present are detectable by UV and that the response of each substance is equal. Neither of these assumptions are correct and support the idea that relying exclusively on one detection method is not advisable. Despite these shortcomings, the speed and ease of use of DAD detectors make them commonplace in HPLC-MS systems for purity determination [Süß99]. All instrument configurations utilised by CM include a DAD to assess purity by UV absorbance.

### **A.3 Charged Aerosol Detector (CAD)**

CAD can be used to complement, or as an alternative to, UV absorbance or evaporative light scattering detection (ELSD). It is particularly useful for monitoring the concentration of screening compounds in solution. The method has been shown to produce direct quantitation of non-volatile compounds relative to a known standard and has also been successfully integrated into a high-throughput workflow [SG09]. Similarly to UV detection by DAD, CAD produces a chromatogram in which peaks are integrated. The LC-MS system developed by CM includes the ability to provide quantitative measurements of concentration. Substances of a known concentration are processed and stored within the LC-MS database. Each such standard has a timestamp for when it was generated and the concentration measurement as determined by CAD for each analysis is calibrated against the most recent standard on a given instrument.

### **A.4 Mass Spectrometry (MS)**

MS is now a ubiquitous technique in drug discovery due to its speed and sensitivity and the fact that it can be automated, allowing for high-throughput analysis [KYO03]. One use of MS is to measure the molecular mass (MW) of compounds via their mass to charge ratios, but it can also be used to provide some structural information via methods such as tandem MS (MS/MS) [CPL07].

MS works by determining the mass-to-charge ( $m/z$ ) ratio, represented as atomic mass units (mu) per unit charge, which is plotted against ion abundance to create a mass spectrum. The peaks on this spectrum show the relative abundances of the various components of the sample being analysed, or more accurately, the ions of these components created during analysis. Spectra can be generated simultaneously for both

ion modes (positive and negative) and the interpretation of these spectra can reveal information about the structural identity, purity and fragmentation of the sample being analysed [GV03]. The basic components of a MS system are an ionisation source, a mass analyser and a detector.

Analytes, the substance being analysed, are converted into gas-phase ions by one of a number of ionisation techniques. *Electrospray ionisation* (ESI) involves passing a solubilised sample through a small capillary which is at a potential difference to a counter electrode [GV03]. This electrostatic spray creates an aerosol of charged droplets containing the analyte and the solvent. Analyte ions are freed from the solvent and are sent to the mass analyser. The analyte ions are either positively charged (protonation) or negatively charged (deprotonation) depending on their acid/base characteristics. One of the main advantages of ESI is the ability to couple it with liquid separation techniques, allowing each separate component eluted to be analysed using MS. This provides fast and accurate analysis of even complex compounds however a disadvantage of its flowing nature means some sample is wasted [GV03].

*Atmospheric-pressure chemical ionisation* (APCI) is an ionisation technique with some similarities to ESI but involves the discharge of a corona from a very fine needle. Rather than a voltage, a nebulizing gas and heat are used to ionise the analyte [GV03] but the effect is still to produce the ions that are assayed by the MS system [Kor05].

All instruments utilised by CM include ESI ionisation but APCI is only available on certain instruments.

## A.5 Mass Analysers

Mass analysers fall into two general types; those that are optimised for accuracy and those that provide precision/resolution. When the focus is compound identity, accuracy of measurement is more important as the closer the measured mass is to the expected mass, the greater the confidence of a correct identification. Resolution is more critical in situations where there may be two ions with very similar masses as without high resolution, the system may not distinguish between the two peaks and instead report an average mass, which would be incorrect for both [GV03].

The *Time-of-Flight* (TOF) mass analysers measure differences in the velocity of different ions. Theoretically, all ions are formed at the same time in the ion source. They are then accelerated down a drift tube and reach the detector at different times, depending on their velocities which are inversely related to the square root of mass.

TOF detectors have well-defined start times, so are suitable for use with pulse ionisation methods such as *Matrix-assisted laser desorption/ionisation* (MALDI) [GV03].

*Sector analysers* use both an electric and magnetic section to disperse ions according to their  $m/z$  ratio and kinetic energy to charge ratio. Once separated, these ion streams can then be focused onto the detector. *Quadrupoles* have been the most popular mass analyser for use with GC-MS and LC-MS due to their relatively low cost, ease of automation and low energy usage. Quadrupoles achieve mass separation by means of radio frequency (rf) voltages and direct current (DC) voltages applied to four rods. The size of the quadrupole and the rf frequency are kept constant, so that ions of different  $m/z$  can be sequentially allowed to reach the detector by increasing the magnitude of the dc voltages [GV03]. All instruments utilised by CM include a quadrupole mass analyser.

## **Appendix B**

### **LC-MS Data management system**

**LCMS Console**

Filter by Batch reference: All batches

**Submitted Plates: awaiting analysis**

Plate BC	Reference	Date Created	COSMOS ID	#wells
1054085180	set 0 to 7 score 2 to 1	17-FEB-12	UKAP148789-001	96
1054085197	set 0 to 7 score 2 to 1	17-FEB-12	UKAP148789-001	96
1054085166	set 0 to 7 score 2 to 1	17-FEB-12	UKAP148789-001	96
1054085173	set 0 to 7 score 2 to 1	17-FEB-12	UKAP148789-001	96
1054085159	set 0 to 7 score 2 to 1	17-FEB-12	UKAP148789-001	96
1054085135	set 0 to 7 score 2 to 1	17-FEB-12	UKAP148789-001	95

Plate count = 201

**Analysed Plates: awaiting review**

Plate BC	Reference	Date Created	Date Analysed	Meth ID	Instr	#reviews
1051850750	set 0 to 7 score 4 to 2	01-FEB-12	16-FEB-12	9	Agilent1	72
1053390599	8 to 11 score greater than 4 reruns	14-FEB-12	16-FEB-12	11	Agilent2	91
1051850767	set 0 to 7 score 4 to 2	01-FEB-12	16-FEB-12	9	Agilent1	62

Plate count = 14

**Reviewed Plates: ready for reporting**

Plate BC	Reference	Date Created	Date Reviewed	Meth ID	Instr
1054084695	moln vessel check	17-FEB-12	17-FEB-12	10	HTS1
1051850798	set 0 to 7 score 4 to 2	01-FEB-12	17-FEB-12	9	Agilent1
1051850231	set 0 to 7 score 4 to 2	01-FEB-12	17-FEB-12	10	HTS1
1051850255	set 0 to 7 score 4 to 2	01-FEB-12	17-FEB-12	11	Agilent2

Plate count = 37

PLATE MAP: 1051850798

Key for well colours:

- Not yet analysed
- Passed analysis criteria
- No peak data
- Failed: requires review
- Failed: has been reviewed

COSMOS ORDER

CONTAINER

JOB BATCH REFERENCE

INSTRUMENT & METHOD

Console Review Form Batch Report Data search

Figure B.1: Workflow Management System

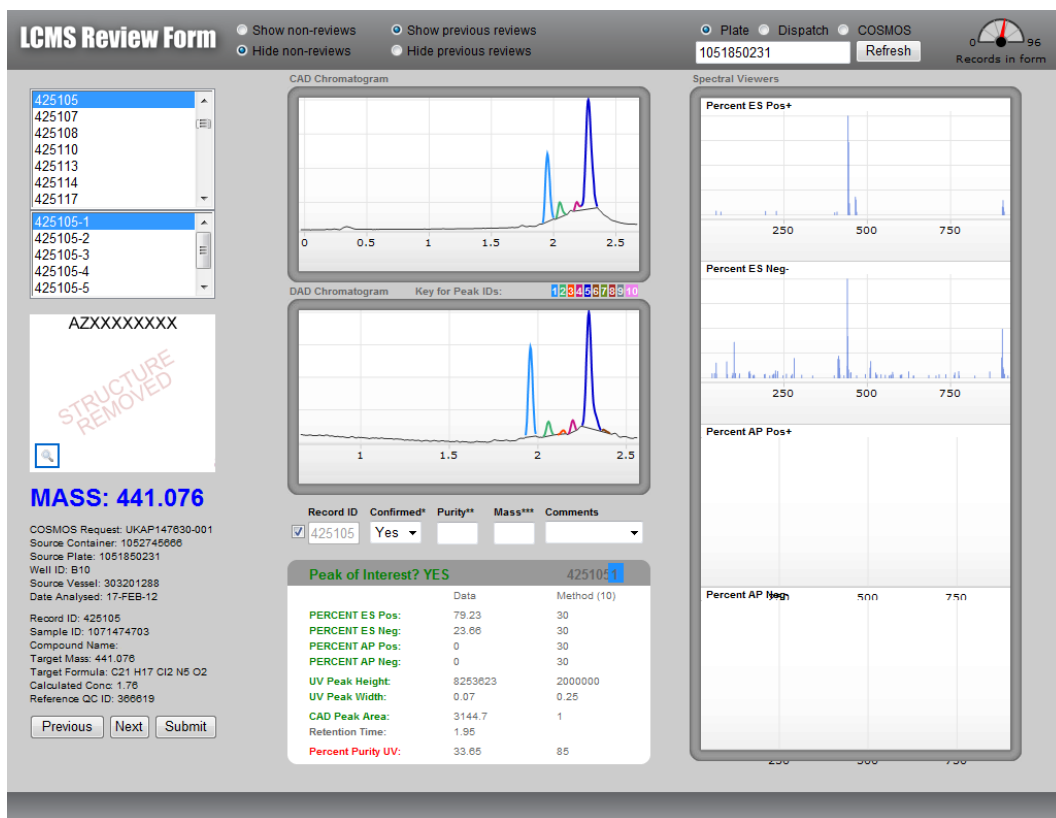


Figure B.2: Analysis review system



### LCMS Data Search

Select a Filter

Plate Barcode: 1051850828

Record IDs in search results:

426613  
426614  
426615  
426616  
426617  
426618  
426619

[Click to open Visualizer...](#)

SEARCH RESULTS: (total records in result = 96)

Id	Plate	Well	Container	Sample Id	AZ Name	Meth	Source	Date Processed	Review?
426613	1051850828	A01	1052745666	32116470	AZ10268673	11	001430036	2012-02-17	Required
426614	1051850828	A02	1052745666	1066924519	AZ12669850	11	304256081	2012-02-17	Required
426615	1051850828	A03	1052745666	215389782	AZ12198834	11	301251410	2012-02-17	Required
426616	1051850828	A04	1052745666	214502639	AZ12202396	11	002719207	2012-02-17	Required
426617	1051850828	A05	1052745666	26126818	AZ11301626	11	002169741	2012-02-17	Required
426618	1051850828	A06	1052745666	1025277460	AZ10825555	11	305077259	2012-02-17	Required
426619	1051850828	A07	1052745666	1099854634	AZ13441879	11	306248815	2012-02-17	Required
426620	1051850828	A08	1052745666	1089938739	AZ13144358	11	304943499	2012-02-17	Required
426621	1051850828	A09	1052745666	1095164248	AZ13300191	11	304607006	2012-02-17	Required
426622	1051850828	A10	1052745666	214307250	AZ12133357	11	304162393	2012-02-17	Required
426623	1051850828	A11	1052745666	215621939	AZ12230347	11	305512165	2012-02-17	Required

EXPORT RESULTS:





 Console
  Review Form
  Batch Report
  Data search

Figure B.3: Search engine

# Appendix C

## Peak of Interest function

```
CREATE OR REPLACE FUNCTION LCMSUSER.GET_SAMPLE_POI_V4 (this_rec in INTEGER)
RETURN NUMBER IS

    cursor c is
        select vvsp.peak_id
        from   v_glo_sample_peak_with_max vvsp
              ,glo_sample_record         gsr
              ,glo_method                 gm
        where gsr.record_id              = vvsp.record_ref
        and   gm.method_id               = gsr.method_ref
        and   vvsp.max_ms_purity >= gm.spectral_purity
        and   vvsp.record_ref            = this_rec -- parameter
        order by vvsp.percent_purity_uv desc nulls last
              , vvsp.cad_peak_area      desc nulls last;

    tmpVar NUMBER;

BEGIN
    open c;
    fetch c into tmpVar;
    close c;
    RETURN tmpVar;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
    WHEN OTHERS THEN
        if (c%isopen) then
            close c;
        end if;
        RAISE;
END GET_SAMPLE_POI_V4;
/
```

## **Appendix D**

# **Confidence measures for Random Forest Models**

124 APPENDIX D. CONFIDENCE MEASURES FOR RANDOM FOREST MODELS

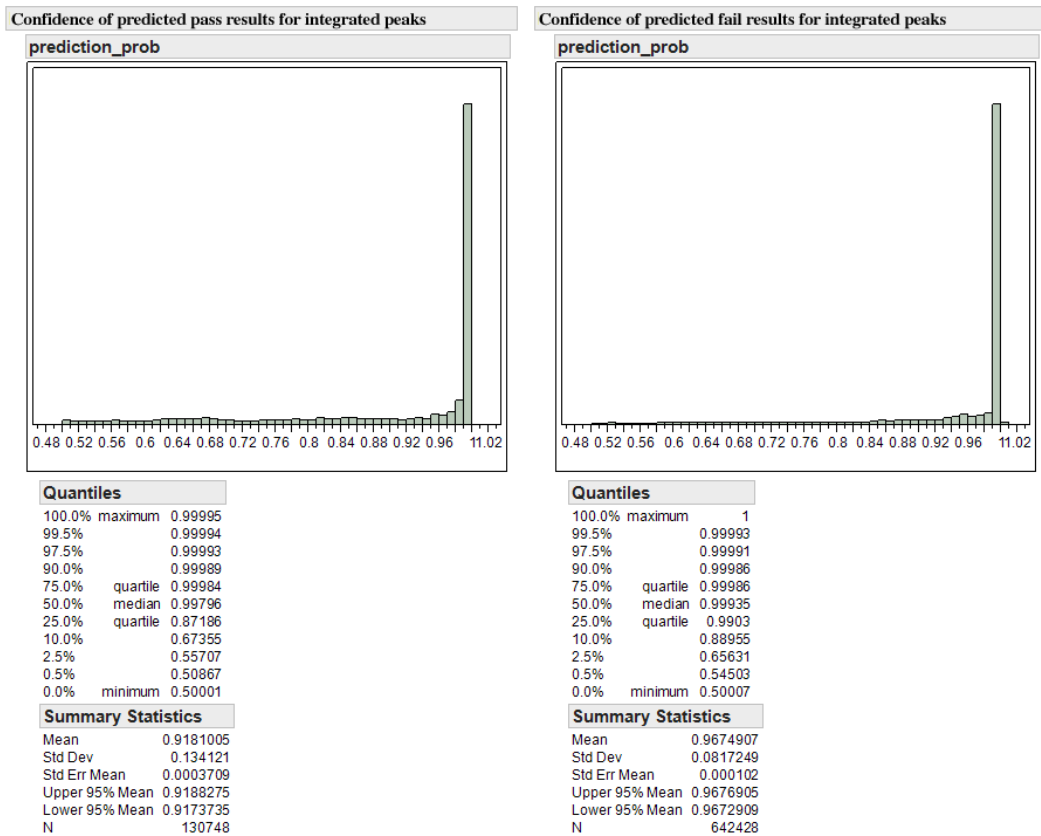


Figure D.1: Distribution of confidences for predictions by Random Forest (10 trees)

# **Appendix E**

## **Full Decision Trees Discussed in Chapter 5**

J48 pruned tree

-----

```

percent_purity_uv <= 78.71
| horz_max_ms <= 29.93: fail (248477.0/1245.0)
| horz_max_ms > 29.93
| | get_cad_percent <= 48.9
| | | percent_purity_uv <= 44.51: fail (23477.0/1148.0)
| | | percent_purity_uv > 44.51
| | | | get_closest_peak <= 0.04: pass (551.0/246.0)
| | | | get_closest_peak > 0.04: fail (2606.0/813.0)
| | | get_cad_percent > 48.9
| | | | get_cad_percent <= 80.8
| | | | | get_closest_peak <= 0.6
| | | | | | uv_peak_width <= 0.03
| | | | | | | num_of_peaks <= 5
| | | | | | | | retention_time <= 0.25: fail (626.0/246.0)
| | | | | | | | retention_time > 0.25: pass (530.0/180.0)
| | | | | | | | num_of_peaks > 5: fail (518.0/120.0)
| | | | | | | | uv_peak_width > 0.03: fail (3974.0/891.0)
| | | | | | | | get_closest_peak > 0.6: pass (678.0/307.0)
| | | | | get_cad_percent > 80.8
| | | | | | get_closest_peak <= 0.25
| | | | | | | retention_time <= 1.18: pass (515.0/71.0)
| | | | | | | retention_time > 1.18
| | | | | | | | horz_max_ms <= 82.98
| | | | | | | | | get_cad_percent <= 90.1: fail (524.0/203.0)
| | | | | | | | | get_cad_percent > 90.1: pass (635.0/282.0)
| | | | | | | | | horz_max_ms > 82.98: pass (519.0/174.0)
| | | | | | | | get_closest_peak > 0.25: pass (1094.0/197.0)
percent_purity_uv > 78.71
| horz_max_ms <= 9.38: fail (8052.0/740.0)
| horz_max_ms > 9.38
| | percent_purity_uv <= 84.97
| | | get_cad_percent <= 87.1
| | | | percent_purity_uv <= 83.3: fail (1578.0/619.0)
| | | | percent_purity_uv > 83.3: pass (595.0/240.0)
| | | | get_cad_percent > 87.1: pass (673.0/210.0)
| | | percent_purity_uv > 84.97: pass (37622.0/711.0)

```

```

Number of Leaves :      19
Size of the tree :      37

```

Figure E.1: J48 Decision Tree output

J48 (Bagging) pruned tree

```

-----
percent_purity_uv <= 79.02
|  horz_max_ms <= 28.38: fail (247179.0/1197.0)
|  horz_max_ms > 28.38
|  |  get_cad_percent <= 46.8
|  |  |  percent_purity_uv <= 45.67: fail (24223.0/1095.0)
|  |  |  percent_purity_uv > 45.67
|  |  |  |  get_closest_peak <= 0.04: pass (516.0/220.0)
|  |  |  |  get_closest_peak > 0.04: fail (2368.0/745.0)
|  |  |  get_cad_percent > 46.8
|  |  |  |  get_cad_percent <= 80.8
|  |  |  |  |  get_closest_peak <= 0.71
|  |  |  |  |  |  uv_peak_width <= 0.03
|  |  |  |  |  |  |  num_of_peaks <= 5
|  |  |  |  |  |  |  |  retention_time <= 0.31: fail (844.0/304.0)
|  |  |  |  |  |  |  |  retention_time > 0.31: pass (543.0/181.0)
|  |  |  |  |  |  |  |  num_of_peaks > 5: fail (604.0/128.0)
|  |  |  |  |  |  |  |  uv_peak_width > 0.03: fail (4379.0/993.0)
|  |  |  |  |  |  |  |  get_closest_peak > 0.71: pass (606.0/255.0)
|  |  |  |  |  |  |  get_cad_percent > 80.8
|  |  |  |  |  |  |  |  get_closest_peak <= 0.22
|  |  |  |  |  |  |  |  |  retention_time <= 1.02: pass (504.0/62.0)
|  |  |  |  |  |  |  |  |  retention_time > 1.02
|  |  |  |  |  |  |  |  |  |  horz_max_ms <= 83
|  |  |  |  |  |  |  |  |  |  |  retention_time <= 1.86: pass (526.0/236.0)
|  |  |  |  |  |  |  |  |  |  |  retention_time > 1.86: fail (734.0/299.0)
|  |  |  |  |  |  |  |  |  |  |  horz_max_ms > 83: pass (508.0/178.0)
|  |  |  |  |  |  |  |  |  |  get_closest_peak > 0.22: pass (1233.0/223.0)
percent_purity_uv > 79.02
|  horz_max_ms <= 8.17: fail (7902.0/665.0)
|  horz_max_ms > 8.17
|  |  percent_purity_uv <= 84.97
|  |  |  get_cad_percent <= 86.7
|  |  |  |  percent_purity_uv <= 83.22: fail (1442.0/563.0)
|  |  |  |  percent_purity_uv > 83.22: pass (619.0/253.0)
|  |  |  |  get_cad_percent > 86.7: pass (681.0/220.0)
|  |  |  percent_purity_uv > 84.97: pass (37833.0/786.0)

Number of Leaves :      19
Size of the tree :      37

```

Figure E.2: J48 Bagging Decision Tree output (example from 10 trees in ensemble)

```
REPTree
=====
percent_purity_uv < 79.02
|  horz_max_ms < 29.97 : fail (248568/1180) [124162/670]
|  horz_max_ms >= 29.97
|  |  get_cad_percent < 46.85 : fail (25930/2095) [13202/1067]
|  |  get_cad_percent >= 46.85
|  |  |  get_cad_percent < 81.05
|  |  |  |  uv_peak_height < 1112645
|  |  |  |  |  retention_time < 0.38 : fail (1441/478) [693/228]
|  |  |  |  |  retention_time >= 0.38 : pass (1187/490) [607/281]
|  |  |  |  |  uv_peak_height >= 1112645 : fail (4239/994) [2149/457]
|  |  |  |  get_cad_percent >= 81.05 : pass (3304/1035) [1663/554]
percent_purity_uv >= 79.02
|  horz_max_ms < 8.93 : fail (8064/712) [3951/396]
|  horz_max_ms >= 8.93
|  |  percent_purity_uv < 85
|  |  |  get_cad_percent < 87.25 : fail (2142/976) [993/450]
|  |  |  get_cad_percent >= 87.25 : pass (643/201) [335/101]
|  |  percent_purity_uv >= 85 : pass (37725/749) [18867/395]

Size of the tree : 19
```

Figure E.3: REPTree output