Assessing the discoverability of variant proteins causing rare forms of paediatric diabetes

by

Lorensha Naidoo

Thesis presented in fulfilment of the requirements for the degree of
Master of Science in the Faculty of Science at
Stellenbosch University

Supervisor: Prof. Hugh-George Patterton
Co-supervisor: Prof. Marc Vaudel

December 2025

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

December 2025

Abstract

Monogenic diabetes is a rare form of paediatric diabetes caused by a pathogenic variant occurring in a single gene associated with insulin production from pancreatic β-cells, resulting in hyperglycaemic complications. Maturity-onset diabetes of the young (MODY) is a subtype of monogenic diabetes, accounting for 2 – 5% of diabetic cases. Patient classification has revealed undiagnosed symptomatic cohorts speculated to result from unknown genetic variants. Identifying these variants is essential for advancing precision medicine and providing specialised medical care.

Proteogenomics allows for the identification of alternative forms of proteins resulting from genomic variation. Protein samples are processed using mass spectrometry to obtain peptide sequences that are then annotated to a sequence database using search engines, e.g. SEQUEST and X!Tandem. Peptide-spectrum matches (PSMs) with varying confidence scores are produced; however, there is no clear distinction between correct and incorrect PSMs. Additionally, distinguishing variant PSMs from canonical PSMs remains challenging due to their low frequency and sequence similarity.

The target-decoy approach (TDA) is a common method for classifying correct and incorrect PSMs and is used in existing PSM processing tools, such as Percolator. Target sequences are peptide sequences from proteomic databases, while decoy sequences are artificially generated to serve as a null model for error rate estimation. To the knowledge of this work, the TDA has not been implemented towards improving the discrimination performance of variant PSMs.

To this end, this study conducts an exploratory analysis to improve the classification of canonical and variant PSMs using the TDA. To achieve this, a machine learning (ML) classification pipeline named Nagilums Tree written in Python, is designed to classify a PSM dataset consisting of target and decoy labelled spectra characterised by multiple scoring functions issued during annotation. ML base models are built using Scikit-learn, which produces a prediction probability $p(-1)$ test statistic that is used to rank classified spectra in order of significance. Spectra ranking is necessary for statistical inference and estimating error rate metrics for the three implementation tasks investigated in this work.

First, the discrimination performance of five decision tree ensemble architectures (Random Forest, Gradient Boosting, Histogram Gradient Boosting, Extra Trees and XGBoost) is evaluated. Second, the novel concept of decoy variant PSMs is investigated by processing canonical and variant PSMs of proteomic data from pluripotent stem cells induced into pancreatic β-cells with an HNF1A (MODY3) variant, and its performance is compared to the classical TDA. Lastly, the

two decoy strategies are compared in their ability to produce statistically significant MODY-associated PSMs by exploring a novel Bayesian inferred posterior error probability (PEP) method.

Ultimately, Extra Trees produced the most reassuring performance and was used for the second task. The decoy variant PSM strategy improved the probability fitness of the variant PSMs; however, the PEP estimates did not identify the HNF1A variant in either decoy method. Although MODY gene expression was inconclusive, the decoy variant concept proved to be an optimistic starting point for future research. The influence of proteogenomic strategies, ML and statistical inference is discussed for future implementation.

Acknowledgements

I would like to express my deepest gratitude towards the following people and institutions for their continued support of this research project.

This project has been a life ambition for many years, and so I give a special thanks to my supervisor, **Prof. Hugh G. Patterton**, whose guidance, patience, and expertise made this project possible in support of my academic journey.

An equally special thanks to my co-supervisor **Prof. Marc Vaudel**, who not only helped me to create this project but also provided me with the incredible opportunity to visit the University of Bergen (UiB), the Mohn Centre of Diabetes and Precision Medicine, and the Computational Biology Unit (CBU) in Bergen, Norway. I especially thank his lovely family, who helped me through a particularly cold winter in Bergen.

I extend my appreciation to my fellow colleagues of Dr. Vaudel's research team, **Dafni Skiadopoulou** , **Jakub Vašíček** and **Dr Ksenia Kuznetsova**, for providing the data used in this work, their invaluable input to the project and their support during my time abroad. I thank **Dr. Divya Tallapragada**, whose work regarding MODY patient classification for the Norwegian diabetes registries was provided for this project. Their contribution to this project is referenced throughout this thesis.

I am also thankful to **Stellenbosch University** for providing the funding necessary for this research. I thank the **Semester Exchange committee** for accepting me as an exchange student and providing the opportunity to travel abroad.

I thank **SANORD**, as the recipient of the Brian O'Connell Scholarship Program, for making my adventure to Norway possible. I hope to continue my dedication to partake in science for the betterment of life for all.

For my loving grandparents, to whom I miss dearly and owe everything.

I thank you all

We are all time travellers, journeying together into the future.
Let us work together to make that future a place we want to visit.
Be brave, be curious, be determined.
Overcome the odds.
It can be done.

-STEPHEN HAWKIN
BRIEF ANSWERS TO BIG QUESTIONS

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AUC	Area under the curve
AUROC	Area under the receiver operator characteristic
D	Decoy (PSM)
DNA	Deoxyribonucleic acid
FDR	False discovery rate
FWER	Family-wise error rate
FN	False negative
FP	False positive
FPR	False positive rate
GCK	Glucokinase
GWAS	Genome-wide association study
HLA	Human leukocyte antigen
HNF	Hepatocyte nuclear factor
iPSC	Induced pluripotent stem cell
LC-MS/MS	Liquid chromatography-tandem mass spectrometry
MAF	Minor allele frequency
MCDPM	Mohn Center for Diabetes and Precision Medicine
ML	Machine learning
MODY	Maturity-onset diabetes of the young
NDM	Neonatal diabetes mellitus
NT	Nagilums Tree
PEP	Posterior error probability
PR	Precision-recall
PSC	Pluripotent stem cell
PSM	Peptide-spectrum match
ROC	Receiver operator characteristic
SNPs	Single nucleotide polymorphisms
SVM	Support vector machine
T	Target (PSM)

T1D	Type 1 diabetes
T2D	Type 2 diabetes
TN	True negative
TP	True positive
TPR	True positive rate

Chapter 1: Literature Review

1.1 Introduction

Medical practice finds its roots emerging from a foundation originating well over 3000 years ago. Ancient civilisations developed rudimentary techniques to diagnose ailments afflicting the human condition (Jouanna, 2012). Subsequently, specialised treatments were practised, such as localising conditions to human anatomy, as an early approach to precision medicine. The 'father of Western medicine' Hippocrates, later reinvented these practices into modern standards by including research rationale (Sykiotis *et al*., 2005). Twenty-five centuries later, these fundamentals would advance human healthcare and be in practice today. However, in more recent times medical practitioners sought to create a health care system accessible to the wider population and adopted a generalised approach (Kurland and Molgaard, 1981). The ability to archive patient data allowed practitioners to analytically develop standardised therapeutic procedures.

After the discovery of the double helical model in 1953 by Francis Crick, J.D. Watson, Rosalind Franklin and M.H.F. Wilkins (Crick, 1954), it would be just 20 years later when recombinant DNA revolutionised medical research. The birth of Sanger sequencing shortly after in 1977 (Sanger *et al*., 1977) led to the global initiative of the Human Genome Project in 1990 (Watson, 1990). Completed in 2021 (Nurk *et al.*, 2022), this 13-year project opened the floodgates for human health research and redefined the possibilities of precision medicine. The ability to understand genetic factors influencing biological processes of complex diseases through Genome-wide association studies (GWAS) has provided a more accurate measure for diagnostics, rather than relying on observing phenotypical symptoms. The genetic diversity amongst human populations can be recognised, and health conditions with overlapping traits, including rare subtypes with unique markers, can be identified (Adeyemo *et al*., 2009; Gudbjartsson *et al*., 2015; Chen *et al*., 2019). More importantly, targeted therapies can be developed and administered in this new era of personalised medicine. This is especially beneficial for the complex field of non-communicable disorders, such as cancer, cardiovascular diseases and diabetes.

1.2 Diabetes mellitus

Glucose is a primary energy molecule responsible for cellular function in vertebrate species. It enters the bloodstream either through ingestion or by metabolising stored compounds, such as fats and proteins (Tirone and Brunicardi, 2001). The islet of Langerhans, situated in the pancreas, consists of the major cells producing the hormones responsible for glucose regulation: α-cells (glucagon secretion), β-cells (insulin), δ-cells (somatostatin), γ-cells (pancreatic polypeptide) and ε-cells (ghrelin) (Steiner *et al.*, 2010). The biological process of these cells is stimulated by the presence or absence of glucose (**Figure 1.1**).

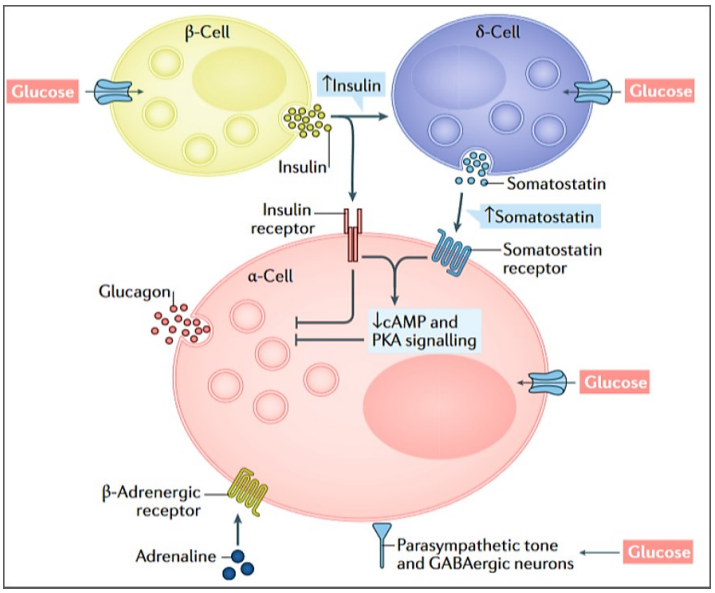


Figure 1.1 Pancreatic cellular interaction during glucose regulation. The biochemical response of pancreatic cells is activated by glucose. The α-cells process stored glucagon, and the β-cells release insulin. This unmodified figure is from Gromada *et al.* (2018).

The more important cells are the α-cells, which metabolise stored glycogen into glucose within the liver during the absence of digested glucose in the blood (Barg, 2003), and the β-cells, which release insulin. Insulin is a protein molecule required by tissue cells (e.g. fat and muscles) to activate glucose receptors. Then, glucose and interacting molecules can enter the cells as an energy source, driving metabolic processes (Stefan *et al.*, 1987). Absence of insulin results in elevated glucose levels (hyperglycaemia) in the blood (plasma) to which the extent negatively affects human health. Complications arising from severe hyperglycaemia include microvascular conditions, such as nephropathy (kidney dysfunction), retinopathy (eye loss) and neuropathy (nerve damage), and macrovascular conditions, such as cardiovascular (heart), cerebrovascular (stroke) and peripheral vascular (limb function) (Krentz *et al.*, 2007). Diabetes mellitus is a chronic metabolic condition developing from insulin dysfunction or poor regulation.

In 2021, the International Diabetes Federation reported that 537 million people globally were diagnosed with diabetes (IDF, 2021). Cases are continually increasing with the development of socio-economic landscapes. The disorder, however, has been acknowledged for thousands of years. Ancient civilisations, such as the Indians, Egyptians and Greeks, reported the urine of certain individuals as sweet (mellitus being Latin for 'sweet' or 'honeyed') with an inclination to attract insects and animals (Trowell, 1982). Pre-diabetics (mild hyperglycaemia) were considered cured once nature no longer held interest in early prognostic evaluations. Modern day diagnostics monitor glucose in the blood, plasma and serum under different conditions, such as after consuming a glucose liquid (Oral Glucose Tolerance Test), after a fasting period (Fasting Plasma Glucose), and glucose-bound oxygenated haemoglobin levels (HbA1c) (Kim *et al*., 2019). Although a combination of glucose tests is typical of diabetes diagnostics, patients are often misdiagnosed regarding the type of diabetes.

Previously, diabetes had been classified into three main subtypes: type 1 diabetes (T1D), type 2 diabetes (T2D) and gestational diabetes (T2D conditions developing during pregnancy). T1D is an immune system response that detects β-cells as foreign bodies, resulting in their deterioration (Principi *et al*., 2017). T2D occurs from the progressive dysfunction of β-cells, resulting in poor insulin secretion or cellular insulin resistance (Udler *et al*., 2018). The development of GWAS has provided the ability to identify prominent genetic factors influencing the biological processes of human diseases. Owing to this, the pathophysiology of diabetes has been redefined as a spectrum, which identifies several subtypes as depicted in **Figure 1.2[a]**. Attributes, such as genetic markers, molecular factors and age of onset, are now considered to redefine the heterogeneity of diabetes. The influence of environmental factors is considered in the modern scope of precision medicine.

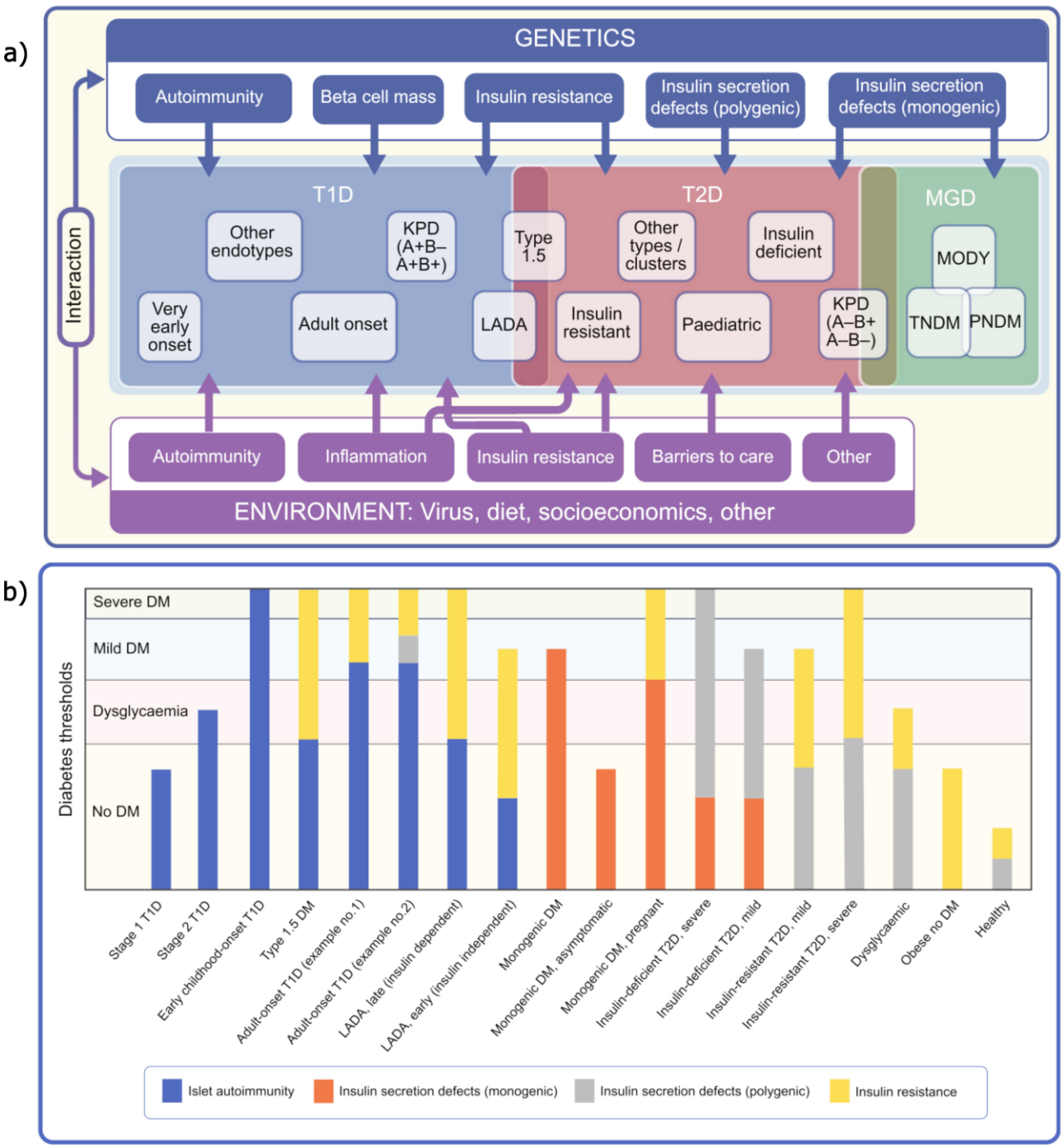


Figure 1.2 The heterogeneity of diabetes. Diabetes subtypes are classified by genetic dysfunction and biochemical onset. **(a) The spectrum of diabetes.** Classification of subtypes is within the broader divisions of T1D, T2D and monogenic diabetes (MGD). The influence of environmental factors is considered. **(b) The composition of biochemical manifestations of diabetes subtypes.** The severity of the diabetes subtype influences its complexity and treatment response. Both of these unmodified figures are from Redondo *et al.* (2020).

The ability to gauge the severity of the subtypes with shared phenotypic symptoms by identifying their characterising factors (**Figure 1.2[b]**) has drastically improved clinical care. This is due to the expression of certain alleles that serve as diagnostic markers; for example, the human leukocyte antigen (HLA) is used to identify T1D. GWAS has identified alternative non-HLA genomic variants, and in combination with HLA, clinical researchers have been able to determine risk factors across age groups and prescribe targeted treatment plans (Yamashita *et al*., 2011). Several rare types of diabetes that had been previously misclassified as dominant class types (T1D and T2D) are now as well able to be better acknowledged. This is the case for 'special' types, which are those occurring from therapeutic drawbacks, communicable conditions (bacterial and viral infections) (Principi *et al.*, 2017) and non-communicable conditions, e.g. cancer (Habib *et al*., 2012). A prominent rare subtype that is currently receiving more attention in recent years is paediatric monogenic diabetes.

Monogenic diabetes is subdivided into two main categories separated by age of onset. The first is neonatal diabetes mellitus (NDM), which develops in infants within the first 6 months after birth (Beltrand *et al*., 2020). The second is maturity-onset diabetes of the young (MODY), a highly prevalent form of paediatric diabetes developing in age groups between infancy, adolescence and young adulthood.

1.3 Discovery and background of MODY

1.3.1 Brief history

Familial inherited diabetes was conceptualised in 1928, in which mild diabetic cases were investigated (Cammidge, 1928). Later throughout the 1950s, Fajans and Conn (1954) reported hyperglycaemic conditions in younger populations whilst investigating average blood glucose levels across age groups. After recording the extensive familial diabetes history of a 70-year-old patient, the research of Fajans uncovered the patterns of inheritance in 1958 (Fajans and Conn, 1958; Fajans and Bell, 2011). Fajans and Tattersall then demonstrated the occurrence of asymptomatic diabetes in children and young adults presenting mild hyperglycaemia despite displaying atypical phenotypes of diabetic conditions at the time, e.g. non-obesity (Tattersall *et al*., 1975). In 1964, Fajans coined the term 'maturity-onset type diabetes of childhood or of the young', aimed to distinguish age of onset as juvenile (now T1D) and maturity-onset (now T2D) (Fajans and Bell, 2011). Later, the term MODY was formally abbreviated, and its characteristic features for diagnosis, such as the patterns of familial inheritance and symptomatic traits, were established (Tattersall *et al.*, 1975). In recent years, clinical practice has clearly defined the biochemistry of MODY, with ongoing research revealing new insights into the condition.

1.3.2 Genetic factors characterising MODY

The aetiology of monogenic diabetes is described by a heterozygous gene dysfunction, i.e. a pathogenic variant on a single allele within a chromosome (Jackson *et al*., 2018). There are currently 14 genes known to be associated with MODY as listed in **Table 1.1**, which was created using the combined information from the works of Urakami (2019) and Broome *et al*. (2021). Dysfunction of the glucokinase gene (GCK) and hepatocyte nuclear factor (HNF) are responsible for the majority of MODY cases. Additional rare MODY subtypes are being uncovered in on-going research, such as by Patel *et al*. (2017) and Mohan *et al*. (2018). Although these new discoveries are exciting and interesting, the scope of this work will focus on the 14 MODY genes established in academic literature for brevity. A single variant (haploinsufficiency) (Garin *et al.*, 2008) in one of these genes disrupts β-cell function, causing insulin suppression. The term 'variant' describes changes in the deoxyribonucleic acid (DNA) sequence that influence gene functionality and protein expression. The term 'mutation' is often used in literature, referring to the same process; however, regarding the subject of human health, the term 'variant' is the person-first terminology and will be used hereon.

Table 1.1 Characteristics of the 14 MODY genes.

Gene (MODY)	Frequency (%)	Gene Function	Aetiology	Treatment
HNF4A (MODY1)	5	Transcription factor	β-cell dysfunction	Sulfonylurea, insulin, GLP-1 RA
GCK (MODY2)	15 - 20	Enzyme	Glucose-sensing defect	No treatment
HNF1A (MODY3)	30 - 50	Transcription factor	β-cell dysfunction	Sulfonylurea
PDX1 (MODY4)	<1	Transcription factor	β-cell dysfunction	Sulfonylurea, OHA, insulin
HNF1B (MODY5)	<1	Transcription factor	β-cell dysfunction	Sulfonylurea, insulin,
NEUROD1 (MODY6)	<1	Transcription factor	β-cell dysfunction	Diet, Sulfonylurea, OHA, insulin
KLF11 (MODY7)	<1	Transcription factor	β-cell dysfunction	Insulin
CEL (MODY8)	<1	Molecular functioning of the pancreas	Pancreas endocrine and exocrine dysfunction	Diet, OHA, sulfonylurea, insulin
PAX4 (MODY9)	<1	Transcription factor	β-cell dysfunction	OHA, Sulfonylurea, insulin
INS (MODY10)	<1	Insulin precursor	Insulin gene variant	Diet, OHA, sulfonylurea, insulin
BLK (MODY11)	<1	Tyrosine kinase functionality	Insulin secretion defect	Diet, OHA, sulfonylurea, insulin
ABCC8 (MODY12)	<1	Facilitates insulin release	ATP-sensitive potassium channel dysfunction	Sulfonylurea, SGLT-2 inhibitors, insulin
KCNJ11 (MODY13)	<1	Facilitates insulin release	ATP-sensitive potassium channel dysfunction	OHA, sulfonylurea, insulin
APPL1 (MODY14)	<1	Insulin signalling	Insulin secretion defect	Diet, OHA, sulfonylurea, insulin

1.3.2.1 Types of genomic variations

Single nucleotide polymorphisms (SNPs) are defined as a variant occurring on a single nucleotide (adenine, thymine, cytosine or guanine) within the nucleic acid sequence of the genome (Jackson *et al.*, 2018). The frequency of SNPs differs among population groups (Choudhury *et al.*, 2014), which can serve as markers in drug response for personalised medicine. Gene alterations manifest as various types, such as insertions or deletions (indels), short tandem repeats, and copy number variants (**Figure 1.3[a]**). Other structural variants are produced from sequence translocation, inversion and duplication. The impact of a nucleotide change returns certain consequences (**Figure 1.3[b]**) to codon regions (three nucleotides forming an amino acid). Alterations in the codon influence gene expression and produce protein products irregular to metabolic bioprocesses, which can be detrimental and cause severe health concerns.

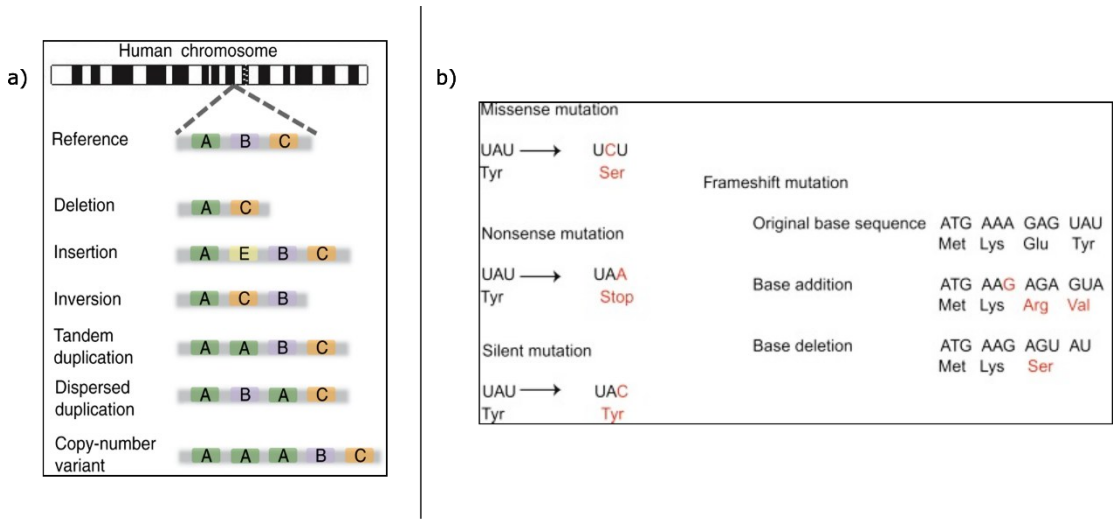


Figure 1.3 Alternative forms of genes and proteins. (a) Types of structural variants caused by nucleotide modifications. The letters (A, B, C, E) represent nonspecific nucleotides and demonstrate the modification. This unmodified figure is from (Baker, 2012). **(b) The effect of point mutations in codons.** Certain nucleotide modifications alter a codon and change the amino acid produced (missense mutation). This can either: create a non-functional protein (nonsense mutation), have no effect (silent mutation) or result in a change of protein expression (frameshift mutation). This unmodified figure is from Shen (2019).

1.3.2.2 Influence of genomic variants causing MODY

Missense SNPs occurring on genomic regions producing transcription factors increase the severity of a condition as they are responsible for manufacturing proteins from DNA at multiple genomic sites (Chang *et al*., 2018). The majority of MODY genes produce transcription factors (**Table 1.1**), and in addition to insulin suppression, interacting proteins (Laddach *et al*., 2018) are indirectly impacted, which can further worsen an individual's health. Additionally, a single gene can produce multiple different protein products (isoforms) that are expressed in human tissues at various frequencies. Two functional isoforms of HNF1A (MODY3) for example, HNF1A(A) and HNF1A(B), are highly expressed in the liver and pancreas respectively (Harries *et al*., 2009), thereby influencing tissue-specific dysfunction.

The minor allele frequency (MAF) is a measurement describing the commonality of uncommon SNPs within a given population (Manolio *et al*., 2009). In **Figure 1.4[a]** the relationship between allele (variant) frequency and phenotypic effect (penetrance) describes the prevalence of human diseases. **Figure 1.4[b]** translates this effect to the occurrence of diabetes subtypes. Common variants are associated with polygenic conditions, such as T1D and T2D, and are attributed to a combination of low-risk alleles (Manolio *et al.*, 2009). A 90% majority of diabetic cases are classified as T2D (IDF, 2021). Due to this large population, common variants are more frequent and have a good chance of being identified in GWAS. In contrast, monogenic diabetes is prevalent in 2.5 – 6.5% of paediatric diabetes (Greeley *et al*., 2022) and MODY accounts for 2 – 5% (Nkonge *et al*., 2020) of clinically diagnosed diabetic cases worldwide, within various population groups (Rafique *et al*., 2021).

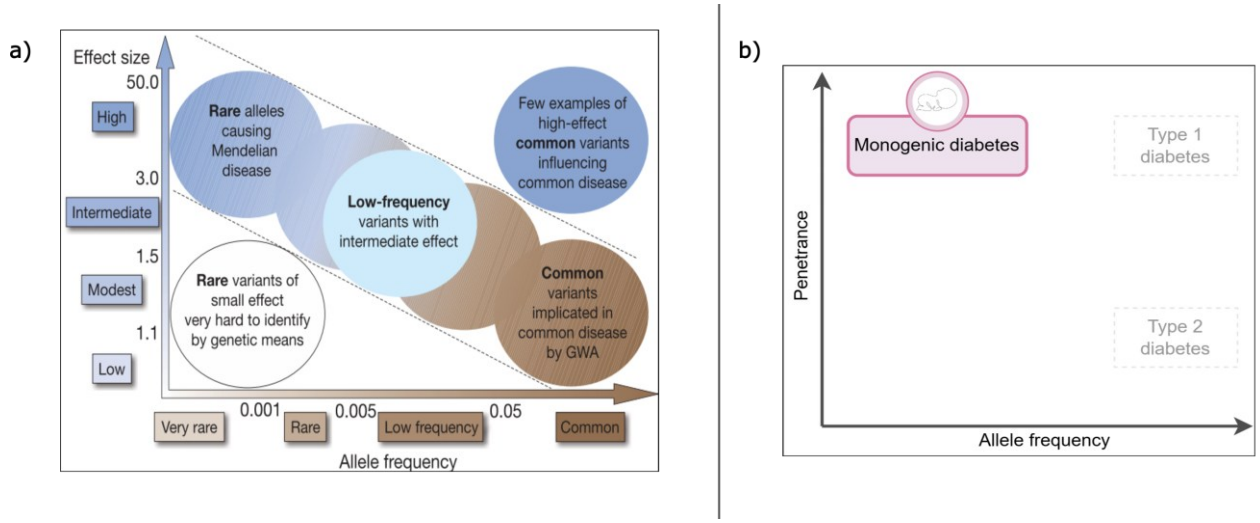


Figure 1.4. Pathogenicity of variants determined by allele frequency. (a) The influence of allele frequency on the pathophysiology of human diseases. Rare conditions correlate to less frequent allele changes as opposed to common conditions in general populations. This unmodified figure is from Manolio *et al.* (2009). **(b) The influence of allele frequency on diabetes subtypes.** An increase in allele frequency is synonymous with polygenic groups, such as type 1 diabetes and type 2 diabetes. Monogenic diabetes is a product of alleles with low frequency and high penetrance.

Rare variants defined by a MAF less than 0.5 (Manolio *et al.*, 2009) are not frequent enough in populations to be identified by genotyping arrays in GWAS. Additionally, not all variants cause MODY, e.g. HNF1A has over 1,200 variants (Valkovicova *et al.*, 2019) and further research into Indian populations has revealed 28.6% to be pathogenic (Kavitha *et al.*, 2023), whilst in different population groups (Indian and Czech) an average of 70% are potentially pathogenic (Malikova *et al.*, 2020; Kavitha *et al.*, 2023). Considering this, the frequency of relevant MODY cases is lower than expected. However, the chance for detection improves for cases with high penetrance, such as Mendelian conditions.

1.3.3 Familial inheritance of MODY

Monogenic diabetes possesses germline variants following Mendelian patterns of inheritance (**Figure 1.5)**, meaning that it is either classified as autosomal recessive or dominant. Offspring of autosomal recessive lineage have a 25% chance of being affected, resulting from two unaffected carrier (benign) parents (**Figure 1.5[a]**). A single affected parent incurs a 50% chance of producing an afflicted offspring during autosomal dominant events (**Figure 1.5[b]**). MODY is inherited through autosomal dominant events, and this increased risk has prompted the development of registries to track the familial history of diabetes within populations.

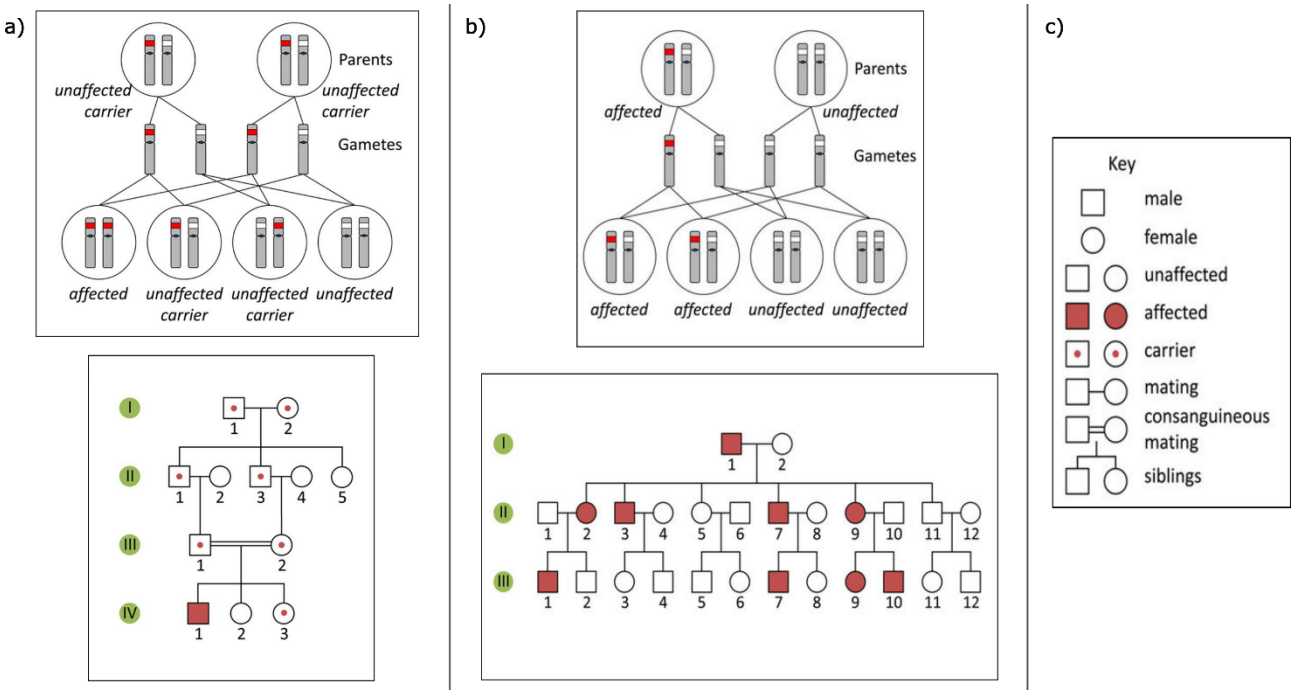


Figure 1.5 Mendelian patterns of inheritance. A pathogenic variant occurring on a single allele within the parental chromosome is passed to offspring in different inheritance patterns. Afflicted individuals are affected within generational lineage at various frequencies as depicted in the bottom figures of **(a) autosomal recessive**, and **(b) autosomal dominant** conditions. **(c) Accompanying legend.** A key to interpret the symbols of both figures. These unmodified figures are from Jackson *et al*. (2018).

1.3.4 Patient classification of MODY for precision medicine

The Mohn Centre for Diabetes and Precision Medicine (MCDPM) in Bergen, Norway, affiliated with the University of Bergen and situated in Haukeland University Hospital, was founded by Professor Emeritus Oddmund Søvik, who also began the first Norwegian diabetes registry in 1997 (Søvik *et al.*, 2013). In recent years, it has developed into a sophisticated system, archiving clinical research data for hundreds of families with hereditary diabetes into specialised units, one being for MODY managed by Professor Njølstad. Novel mechanisms explaining the development of GCK-MODY (Negahdar *et al.*, 2014) and regulation of glucokinase activity through SUMOylation (small ubiquitin-like modifier proteins responsible for targeted protein expression) (Aukrust *et al*., 2013) have been uncovered as a positive result of these registries. These records have proven necessary to identify genomic patterns for research and are expected to aid in identifying pathogenic MODY variants from new and existing patients (Irgens *et al*., 2013).

To identify MODY cases, patient classification pipelines are formulated. The pipeline provided in **Figure 1.6** is a system created for processing suspected MODY cases that have been referred to the MCDPM. The criteria for referrals to be considered include: confirmation of diabetes with supporting clinical diagnostics, relevant family history of diabetes (minimum two generations), the age of diagnosis is below 35 years of age, and negative for auto-antibodies (suggesting T1D) (Broome *et al*., 2021; Zhang *et al*., 2022). The pipeline filters out NDM subjects, and MODY is confirmed for the remaining cases. Curiously, a substantial symptomatic cohort tests negative for MODY, which is further processed to confirm the diagnostic criteria. It is suspected that these cases are a result of unknown variants that have either not been identified in clinical research or are new to a population.

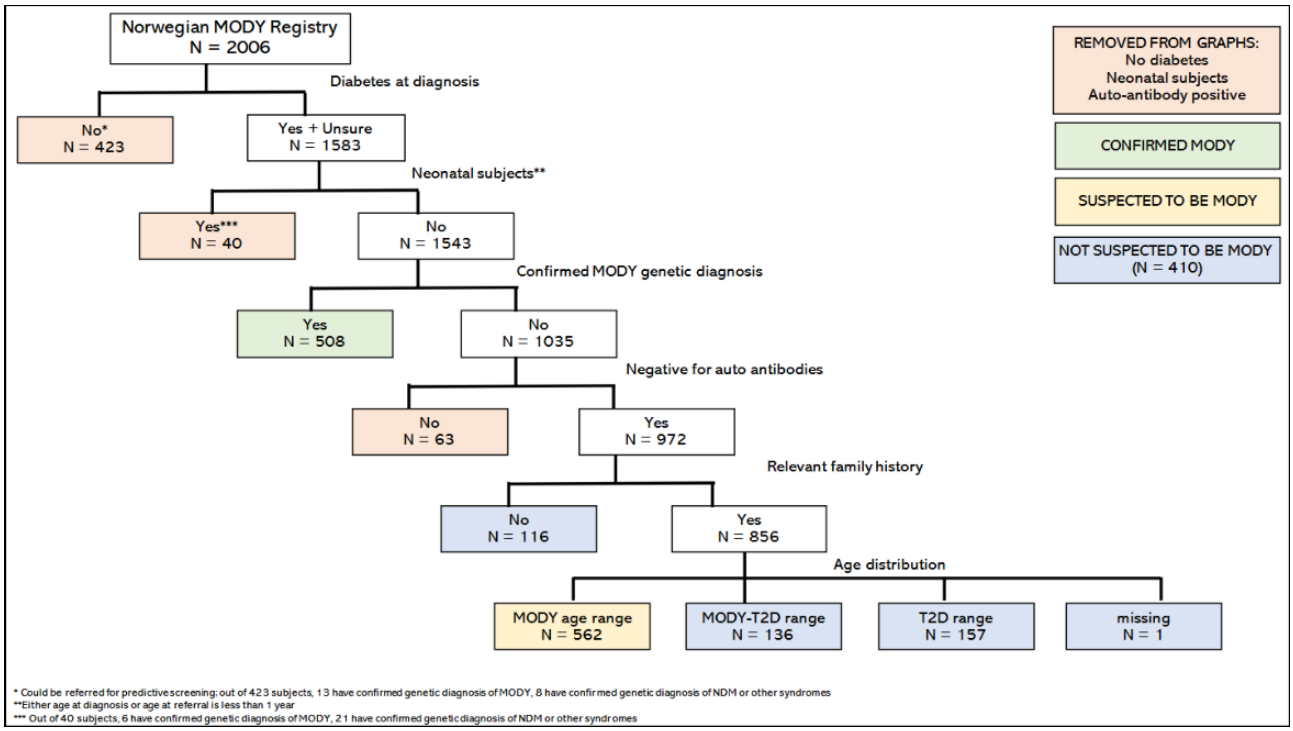


Figure 1.6 MODY patient classification pipeline. A cohort of 2,006 subjects suspected of MODY is processed for diagnostics. The colour-coded boxes in the upper right corner describe the events of the pipeline. This pipeline is used to visualise the underdetermined cohort of MODY cases (yellow box) after classification. This figure is graciously provided by Dr. Divya Tallapragada.

Interestingly, it has been revealed that the prevalence of MODY genes differs amongst populations. HNF1A (MODY3) is common in European populations, such as Norway (Irgens *et al.*, 2013) and the Netherlands (Weinreich *et al*., 2015). In Japan (Yorifuji *et al*., 2012) and Poland (Fendler *et al*., 2012), GCK (MODY2) has been largely identified; however, over 50% of cases are due to unidentified variants. A recent study in South Africa (Matsha *et al*., 2020) revealed HNF1A as the main contributor to MODY cases in a small cohort of 1,643 subjects of mixed ancestry. MODY diagnostics is predominantly conducted in European groups (Rafique *et al*., 2021) owing to their continuing interest, and an important consideration is that GWAS are limited to these populations. New variants have been identified in genomic studies in non-European groups, and the development of African ancestry gene banks (Mulder, 2017) is expected to reveal a greater variety.

Treatment regarding hereditary forms of diabetes and ongoing care is different to that of the more common T1D and T2D. There are specialised targeted therapies and treatment plans curated for the afflicted gene responsible for causing MODY, which are listed in **Table 1.1**. Each gene is responsible for certain biological processes and molecular functions that require personalised care. Insulin medication, developed by Banting *et al*. (1922), is the first line of treatment for most diabetic cases. Insulin is chronically administered via injection into the body as an at-home treatment, and for MODY, which impacts infants, adolescents and young adults, this is acknowledged as a discomfort. It has been discovered that some alternative forms of MODY can be treated with sulfonylurea (Sagen *et al.*, 2004), as a chronically administered oral drug. A clinical study conducted with MODY patients regulated with sulfonylurea revealed improved glucose regulation after 10 years (Bowman *et al.*, 2018). This implies that quality of life can potentially be improved with alternative treatment plans for certain MODY cases.

To provide personalised treatment options, patients are first required to be correctly diagnosed. Monogenic diabetes presents as mild hyperglycaemia in young people and is typically misclassified as T2D (Urakami, 2019). Considering this, it is suspected that the frequency of monogenic diabetes in populations is larger than what is currently reported. To this end, there is a need to develop methods that identify pathogenic variants causing MODY, thereby providing personalised care by correctly classifying undiagnosed individuals. The nature of this work questions whether it is possible to identify these variant protein products using existing *in silico* methods employed for protein identification. Therefore, the next chapters will explain how proteins are identified and how adopting statistical methods using bioinformatic strategies can improve variant protein identification.

1.4 How to identify proteins: proteogenomic pipeline

The proteome and genome describe the entire protein and gene content produced by an organism, respectively (Barbieri *et al.*, 2016). Proteins are the fundamental factors driving biological activity in multicellular organisms and reflect DNA modifications occurring within their complementary gene. However, not all genes are always expressed into protein products. Therefore, analysing the proteins in the blood and plasma confirms that certain genes are processed under specific conditions (Ferkingstad *et al.*, 2021). Proteome composition and disease susceptibility can be evaluated; for example, MODY genes under hyperglycaemic conditions (Malikova *et al.*, 2020). The field of proteogenomics integrates proteomics and genomics, making it an effective strategy for comprehensively understanding the relationship between genetic variations and their functional protein products.

Shotgun proteomics is a standard technique for identifying and quantifying proteins harvested during experimental clinical research. In this method, proteins are enzymatically digested into smaller peptides, which are then analysed by liquid chromatography-tandem mass spectrometry (LC-MS/MS) (Barbieri *et al.*, 2016). The peptide sequences are returned as the result. These observed spectra are then matched against a protein database to infer the peptide identification.

Theoretical protein databanks are collections of confirmed and predicted protein sequences derived from genomic or transcriptomic data. Protein databases include both the canonical proteome (most commonly annotated protein serving as reference) and its known variant protein sequences (haplotypes). Common databases include Ensembl (Harrison *et al.*, 2024) or UniProt (Bateman *et al.*, 2024). The protein sequences from these databases are computationally processed to emulate enzymatic digestion and generate peptide sequences. These theoretical spectra serve as reference sequences to which the experimental spectra are aligned, and peptide identification is inferred (Liu *et al.*, 2007). The identified peptides are mapped back to multiple corresponding proteins due to similarities. Statistical models are applied to remove ambiguity and determine the most probable protein identification (Liu *et al.*, 2007). The nature of this work will focus on peptide identification.

Peptide matching is processed using search engines, such as SEQUEST (Eng *et al.*, 1994), X!Tandem (Craig and Beavis, 2004) and Mascot (Perkins *et al.*, 1999). These tools apply various scoring algorithms that evaluate the similarity between sequences. The alignment with the best or highest scores is retained as a peptide-spectrum match (PSM). These tools are well-structured to provide the best matches and possible scores while removing poor-quality

sequences. However, they generally use a brute force approach to align every spectrum, assuming each is a true peptide (Arnold *et al*., 2006; Stevens *et al*., 2008). In actuality, errors occurring during the experimental procedure, such as incomplete enzymatic digestion, contamination and mass spectrometry miscalibration, result in poor peptide abundance. It is broadly estimated that 10 – 50% of PSMs are identified as true matches (Deutsch *et al*., 2008; Granholm and Käll, 2011; Ezkurdia *et al*., 2014).

The PSMs are returned with multiple scoring functions that are uninformative individually, and there is no clear distinction between correct and incorrect matches. To this end, these raw scores require downstream processing to improve interpretation. Additionally, discriminating between variant and canonical peptides remains challenging due to similarities between the sequences. Currently there is no means to improve the resolution of variant PSMs during downstream processing. To identify correct PSMs, statistical inference is employed to estimate error rates, i.e. the likelihood that a spectrum match is incorrect. The metrics to accomplish this are introduced next in the context of standard PSM processing. Then, in the methods of this work these techniques will be explored in the context of processing variant PSMs.

1.5 How to determine the reliability of PSMs

1.5.1 The target-decoy approach

To establish the reliability of PSMs, various strategies, broadly categorised into two approaches, have been integrated into the search engine annotation process as additional components to increase the accuracy of true identifications. The first approach involves filtering proteomic data through criteria defining the qualities of a true match and then sorting to identify significant matches. This heuristic style is applied in the works of Chen *et al*. (2005), and in established tools, such as DTASelect (Tabb *et al*., 2002) and CHOMPER (Eddes *et al*., 2002). Several probability-based approaches, such as the hidden Markov model (Wu *et al*., 2007), Gaussian distribution (López-Ferrer *et al*., 2004), and Bayesian inference as used in the well-known PSM processing tool PeptideProphet (Keller *et al*., 2002), are included in the second approach. Although either method does not apply to every hypothesis, identifying false positives is fundamentally a statistical resolution, especially for larger datasets (Cargile *et al*., 2004). An effective method, in line with the second approach, was introduced by Elias and Gygi (2007) and is known as the target-decoy search strategy or as the target-decoy approach (TDA) in this work.

Following the creation of PSMs during the proteogenomic method introduced previously, the TDA labels the peptide sequences of the theoretical dataset as 'target' PSMs. These target PSM sequences are either reversed or shuffled to create fake sequences labelled as 'decoy' PSMs (**Figure 1.7**). Alternatively, Markov modelling can increase the randomness of decoy PSMs (Colinge *et al*., 2003), although either option is considered acceptable. Decoy PSMs that are identical to the target spectra are removed. The target and decoy PSMs are combined to create the theoretical dataset that is matched with experimental sequences using a search engine to produce a composite PSM dataset (**Figure 1.8[a, b]**). Target PSMs are interpreted as a mixture of correct non-random and incorrect random identifications, while the decoy PSMs are solely random matches (**Figure 1.9**). It is possible for non-random target spectra to be incorrectly identified as random occurrences.

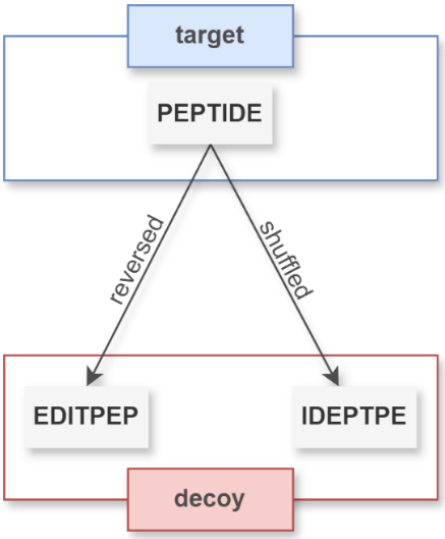


Figure 1.7 Target-decoy approach method. An example target sequence 'PEPTIDE' is either reversed or shuffled to create a fake decoy sequence (Bern and Kil, 2011).

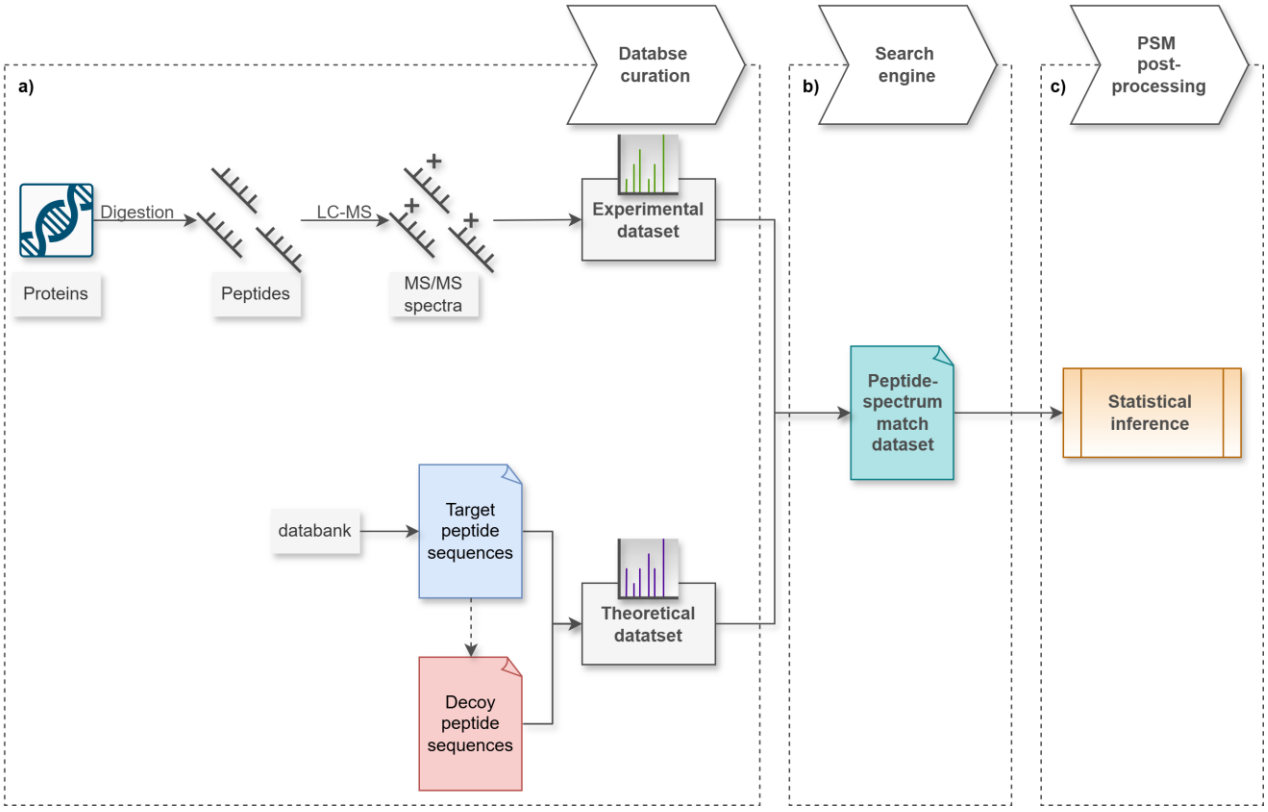


Figure 1.8 Proteogenomic pipeline with target-decoy approach. (a) Database curation: Proteins are fragmented to generate an MS/MS peptide spectrum sequence dataset. Target peptide sequences are harvested from databanks and are combined with decoy sequences to create a theoretical dataset. **(b) Search engine**: The experimental and theoretical datasets are annotated to create a peptide-spectrum match dataset. **(c) PSM post-processing**: PSMs are processed downstream for statistical inference.

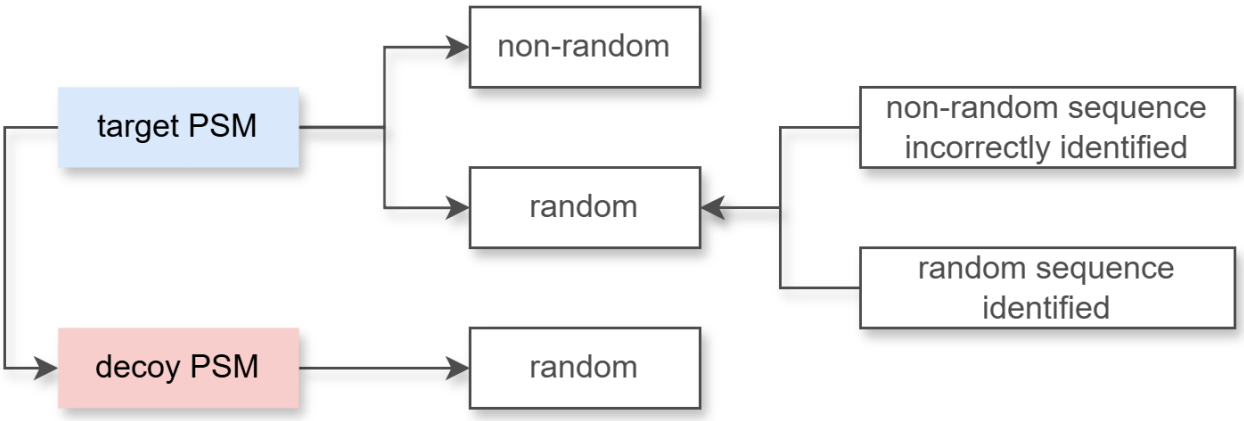


Figure 1.9 Interpretation of target and decoy PSMs. Target PSMs have two possible outcomes and decoy PSMs have one. Random target PSMs are a mixture of two outcomes.

In a separated search method, observed spectra can receive an annotation metric for both target and decoy spectra. A combined search would produce a single PSM annotation based upon whichever has the highest degree of similarity, which is determined using a threshold as observed in **Figure 1.10**. In this illustration, the distribution of a combined set of target and decoy spectra is observed under two annotation conditions representing the size of theoretical datasets. Observed spectra have a higher likelihood of being identified in an environment with a dense population of theoretical spectra, although at the expense of more incorrect matches during annotation. The frequency of false matches can be adjusted by altering the parameters of the search engine's search environment. However, as stated in the proteogenomic pipeline (see Chapter 1.4), not all observed spectra are correctly identified, and regardless PSMs require processing.

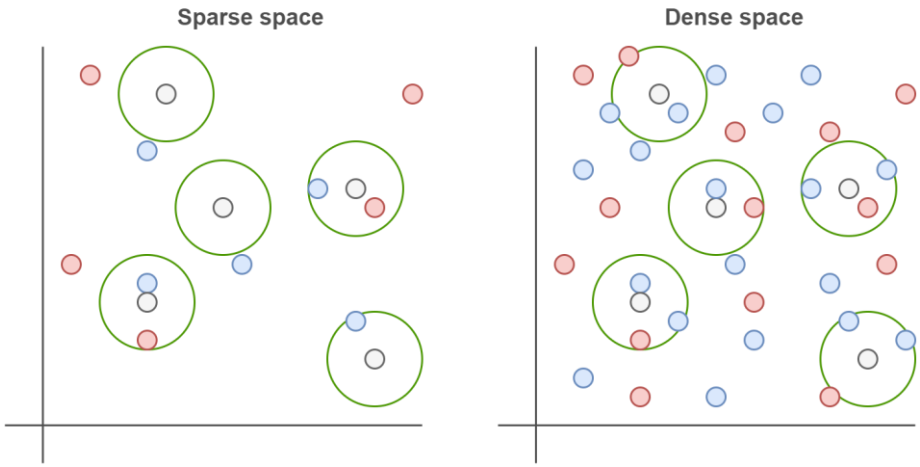


Figure 1.10 Difference in sparse and dense search space environments. The peptide sequences of the observed experimental spectra (grey circles) are identified by its similarity to either the theoretical target (blue circle) or decoy (red circle) spectra. The green circle represents the proximity threshold required for spectra to be considered close in resemblance.

It is not the purpose of PSM processing to confirm a spectrum as a correct or incorrect match. Instead, the goal is to investigate the degree of correctness or incorrectness by exploring statistical strategies using a probability metric inferred onto the PSMs. The confidence of PSMs with a high likelihood of being a true match is justified by statistical significance. Theoretically, the distribution of the processed decoy PSMs can estimate the frequency of incorrect target PSMs (Elias and Gygi, 2007). The decoy PSMs represent identifiable incorrect matches and would be issued an appropriate statistical measurement. Random target spectra (**Figure 1.9**) are expected to have similar statistical weights to random decoy spectra. Statistical inference is applied using the target and decoy PSMs to estimate error rate metrics.

1.5.2 Statistical inference: hypothesis testing

The PSMs generated from a search engine are post-processed to infer a test statistic, such as e-values (Vovk and Wang, 2021) or p-values (Fisher, 1925). The probability scores of p-values are the likelihood that the observed spectrum is a non-random prediction. These scores are irrelevant alone when reporting the quality of a PSM for identification (Anderson *et al*., 2000). To conduct statistical inference, hypothesis testing is employed.

Hypothesis (H) testing is a procedure conducted in the field of statistics, in which data is conditionally evaluated based on its ability to support a given research question (Zhang *et al*., 2017). This is generally a statement created to deem data as significant, i.e. an indication there is a strong statistical likelihood the data are not random occurrences and are likely to be genuine. The p-value is an example of a hypothesis (Biau *et al*., 2010). The null hypothesis (H_0) (Millikin *et al*., 2020) refers to a statement in which no effect is observed for a given research question or hypothesis. Consider for example; after processing, an observed target spectrum is deemed a significant non-random match (**Figure 1.9**) if it is assigned a probability score above a threshold (a), e.g. $a = 0.5$. Therefore, the target spectrum is an insignificant random match if the H_0 is accepted, which would be summarised as:

$$H_0: p \geq 0.5$$

The alternative hypothesis (H_1 or H_a) (Millikin *et al.*, 2020) is thus accepted for a non-random match if the null hypothesis is rejected; the hypothesis is observed as:

$$H_a: p < 0.5$$

Each PSM probability score represents a single hypothesis (H_1); however, in the case of multiple hypothesis testing ($H_1, H_2 \ldots H_n$) (Shaffer, 1986) (also known as multiple testing corrections), the likelihood of observing false positives or Type I errors increases. In remedy, this has been compensated for by the use of the Bonferroni correction (Dunn, 1961; Armstrong, 2014). This was developed as the Bonferroni inequalities, which in probability theory are the lower and upper bounds of a union of events found using Boole's inequality (Stone, 1936; Kneale, 1948). This was then adapted as the family-wise error rate (FWER), which is the probability of a false positive occurring among all the hypotheses (Storey, 2002). This is achieved by controlling the error rate threshold (a_{FWER}), e.g. for (n) spectra, readjusting (a) would be ($a_{FWER} = \frac{a}{n}$) globally for all PSMs (Storey, 2002). This method is known to be stringent due to a disproportionate level of low false positives occurring as (n) increases at the cost of true positives (Nakagawa, 2004; Lu and Westfall, 2009).

1.5.3 False discovery rate

An acceptable alternative proposed by Benjamin and Hochberg (1996) is to use the false discovery rate (FDR) as a less conservative method than the FWER. In the FDR approach, the error rate is a measurement of its proportion to the 'global' spectra family and is controlled by a threshold to include only significant values (Käll *et al*., 2008). An FDR threshold of $a = 0.05$ for example, denotes that spectra have a 95% chance of being non-random (Elias and Gygi, 2007); therefore, lowering the FDR implies a decreasing likelihood the PSMs are random. Spectra with an FDR below the threshold are deemed significant using hypothesis testing; following the example above, the alternative hypothesis is written as:

$$H_a: FDR \leq 0.05$$

Implementation of the FDR in proteomic studies has been adapted as a function of the TDA (Elias and Gygi, 2010). The decoy PSMs essentially act as a null model to estimate the likelihood a spectrum is a true match. In the original TDA method, it was stipulated that the number of incorrect PSMs is proportionate for both the target and decoy datasets (Elias and Gygi, 2007). The FDR was then hypothesised as an estimate doubling the frequency of decoy spectra to include the incorrect matches of itself and the ambiguous matches of the target PSMs (Aggarwal and Yadav, 2016), which appears as the formulae:

$$FDR = \frac{2 \times decoy}{target + decoy}$$

The FDR threshold additionally provides statistical inference; for example, the p-values enable the spectra to be ranked in order of significance. Then, the FDR can be estimated based on the rank for each spectrum, e.g. a target spectrum positioned after 7 target and 2 decoy PSMs would produce the following FDR using the above formulae:

$$FDR = \frac{2 \times 2}{8 + 2} = \frac{4}{10} = 0.04$$

An increase in the FDR threshold would yield more PSMs at the expense of incorrect PSMs being included; however, a stricter value of 1% increases the confidence of the spectra being a true match (Käll *et al*., 2008; Mayo and David, 2022). In the example above, an FDR threshold of 1% would reject this target spectrum as a non-random match, whereas 5% would not.

Since its inception in 2007, the above formula has been adapted to explore alternative false positive hypotheses for various research strategies, specifically adjusting the proportion of decoy spectra. The work of Levitsky *et al*. (2017) suggested increasing decoy representation by scaling the target-decoy ratio to reduce bias when estimating the FDR for the q-value test

statistic. Similarly, scaling was suggested in the work of Kim *et al*. (2019) to account for disproportionately small decoy datasets. One popular method, however, was proposed by Käll *et al*. (2008), which assumes the ratio of decoy spectra is equal to the incorrect target matches:

$$FDR = \frac{D}{T}$$

The above method has become the standard next to Elias and Gygi(2007), owing to it being user-friendly, and having been referenced in several studies concerning protein identification from large datasets (Reiter *et al*., 2009; Rosenberger *et al*., 2017; Wang *et al*., 2018). This FDR method will be used in this work.

1.5.4 Posterior error probability

1.5.4.1 Short background

In the mid-1700s, an essay published by the entrusted Richard Price on behalf of the late Reverend Thomas Bayes, conceptualised implementing past or 'prior' events to predict future or 'posterior' probabilities (Bayes and Price, 1763; Bayes, 2003). This earlier inverse probability method was later developed by the scholar Pierre-Simon Laplace into the presently used Bayes theorem (LaPlace, 1814; Grattan-Guinness, 2005). Based on conditional probability, this method is applied as the Bayesian inference, which determines the statistical likelihood of new observations within specified parameters $\{\theta\}$ of a model $\{M\}$ by updating prior probabilities $\{p(\theta|M)\}$ in what is known as posterior probabilities $\{p(M|x)\}$ (Gabbay *et al*., 2010).

1.5.4.2 Significance of posterior error probabilities

The previous chapter described the FDR method as a global error rate estimate producing a collection of significant spectra. It was also stated that test statistics, such as the p-value, can be ignored by an FDR threshold despite spectra receiving a baseline confident measurement. In proteomic experimentation, there may be a particular spectrum of interest, e.g. a MODY peptide, and it is of interest to report the likelihood of it being a non-random match. Complementary to the FDR method that estimates an error rate for multiple hypotheses, posterior probabilities, initially termed 'local FDR' (Efron *et al*., 2001), employ a negative or incorrect data type to create the posterior error probability (PEP) method.

The application for PSM validation using PEP values finds its roots with PeptideProphet (Keller *et al*., 2002), the first tool to investigate the confidence of mass spectrometry spectra annotated by search engines. Its method later implemented the target-decoy strategy (Choi and Nesvizhskii, 2008). Later, Percolator (Käll *et al*., 2007) was introduced as a standard method to

process target and decoy spectra and estimate the FDR and PEP values. These tools estimate the FDR and PEP using a null hypothesis $\{H_0\}$ modelled by decoy spectra. In **Figure 1.11**, the interchangeable relationship between the FDR and PEP is illustrated for spectra ranked in order by a test statistic. Then for a spectrum-match, the metrics are estimated according to its corresponding probability scores' rank position. The FDR concerns the number of incorrect and correct PSMs regional to the specified spectrum. The method to estimate the PEP scores measures the error rate of an individual spectrum to the sum of decoy and target spectra with the same test statistic score.

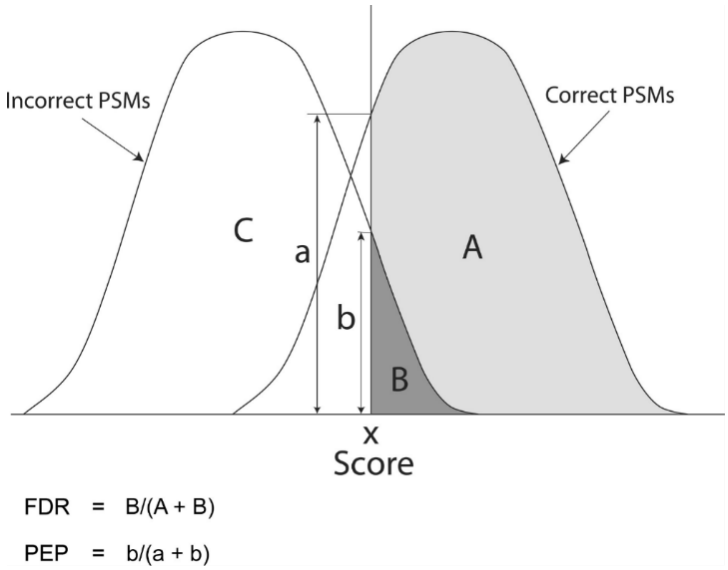


Figure 1.11 Relationship between FDR and PEP. Spectra are ranked by a test statistic. The FDR is the ratio of the number of incorrect and correct spectra ranked above a specific score (x) of a spectrum. The PEP is the ratio of all the incorrect and correct spectra with the same score (x) of a spectrum. The formulae to estimate the FDR and PEP are described. This unmodified figure is from Käll *et al*. (2008).

Using a hypothesis $\{H\}$ statement, the PEP is the ratio of prior probability densities if incorrect $\{f_0 x\}$ and correct $\{f_1 x\}$ spectra with a score $\{x\}$ in relation to all the same scoring spectra $\{X\}$, generating the null hypothesis $\{H_0\}$ as (Käll *et al*., 2008):

$$P(H_0|X = x) = \frac{f_0 x}{f_1 x}$$

Similar to the FDR, a threshold of significance is applied to the spectra; however, the criteria for the PEP estimates are more stringent. While an FDR threshold range of 1 - 5% is generally acceptable, the PEP metric is limited to 1% for reporting peptide identifications (Käll *et al*., 2008).

Decoy spectra emulate incorrect matches by reversing and shuffling the target spectra sequences. Therefore, variant peptides bearing similarity to decoy spectra may be unaccounted for. More concerningly, variant PSMs are compounded with the canonical PSMs as target spectra and are likely to obtain similar test statistics, further making them difficult to identify. To overcome this challenge, existing PSM processing methods can potentially be adapted, which is the nature of this work. Modern PSM processing methods adopt sophisticated computational techniques to obtain probability test statistics.

1.6 Machine learning for processing PSMs

Machine learning (ML), a branch of artificial intelligence, leverages computational technologies to implement statistical algorithms designed for large-scale data analysis (Yoo *et al*., 2014). Its applications span a broad range of research fields, such as robotics (Soori *et al*., 2023) and plant biology (Soltis *et al*., 2020). Here, the attributes of ML will be described in the context of PSM post-processing. Search engines annotate peptides by assigning multiple numerical scores that quantify the similarity between observed and theoretical spectra; for instance, X!Tandem applies Hyperscore, E-value and Delta score (Craig and Beavis, 2004). Conventionally these scores are evaluated independently during PSM assessment. ML provides a framework for collectively processing these scores to generate a single test statistic that infers an overall measure of significance. This approach facilitates a clearer interpretation of the PSMs, which is particularly advantageous when combining outputs from multiple search engines to enhance identification resolution (Kwon *et al*., 2011; Shteynberg *et al*., 2013).

The process of ML involves a learning algorithm or base learner (also referred to as a model) to be *trained,* i.e. identify patterns and trends within a dataset in order to infer predictions on unseen *testing* data. The scoring functions of PSMs are defined as feature vectors (Yoo *et al.*, 2014) or variables (x), and they are assigned labels (y) representing either a target or decoy class. A standard ML model receives this data (D) as a set of coordinates, e.g. $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ (Dietterich, 2000). The numerical measurements of the feature variables characterising each PSM class label are used to train the algorithm formalised as a function, e.g. $f: y = f(x)$. The trained model is then used to classify unlabelled PSMs as either target or decoy class types based solely on the feature variables of the unseen PSM testing data.

The above function is an example of a linear learning algorithm designed to perform a classification task, i.e. categorising a data type. Regression tasks, by contrast, involve predicting continuous values to support decision-making processes (Timm *et al*., 2008). A wide variety of algorithm architectures exist, each employing different functional strategies that may be more

suitable for processing specific tasks, challenges and datasets. There are three principal learning styles (Oladipupo, 2010) and their characteristics and functionalities are briefly described in **Table 1.2**, along with examples of ML architectures appropriate for each style. Supervised and semi-supervised learning styles can perform classification and regression tasks using deep learning models (e.g. Neural Networks) or classical architectures, such as support vector machines (SVM), gradient boosting and decision trees (Al-Azzam and Shatnawi, 2021).

The semi-supervised style includes a self-training model, which involves classifying unlabelled data points by iteratively retraining a model using the dataset's labelled data points (Tanha *et al*., 2015). The concept of self-training is analogous to post-processing target and decoy PSMs, due to the nature of target spectra comprising both non-random and random matches (see Chapter 1.5 and **Figure 1.9**). Random target PSMs are technically unlabelled as they are initially unknown, and through processing they are identified through statistical inference. PSM datasets can range from hundreds to millions of peptide entries that require classification, and certain architectures are more optimal for efficiently processing these large datasets.

Table 1.2 Different machine learning styles.

	Supervised	**Unsupervised** (Lin *et al*., 2012)	**Semi-supervised**
Dataset characteristics	Data points are labelled, and all feature variables are provided	Feature variables only	Some data points are labelled and unlabelled
Learning behaviour and expected outcomes	Model learns patterns of feature variables for each class label and makes predictions on unlabelled data	Model learns and explores patterns of feature variables to create labels or groups with similar characteristics	Model learns from a small amount of labelled data combined with more unlabelled data, usually to save time and to prevent data loss when labels are unavailable
Common algorithms	Linear Regression, Random Forest, Neural Networks	Clustering, K-means	Linear Regression, Random Forest, Neural Networks

The linear function $y = f(x)$ can operate as either supervised or semi-supervised learning and is designed for a single feature variable (x), thus creating a single-dimensional search space environment. Incorporating additional feature variables, i.e. multiple scoring functions, creates a higher-dimensional environment (Clarke *et al.*, 2008). With more dimensions, the model can more effectively separate data points based on the measurement weights of the feature variables. The concept illustrated in **Figure 1.12** can be applied to PSM processing and is used to visualise a learning algorithm performing binary classification of target and decoy spectra. The model can subsequently distinguish between class types and infer label identification onto unseen PSMs.

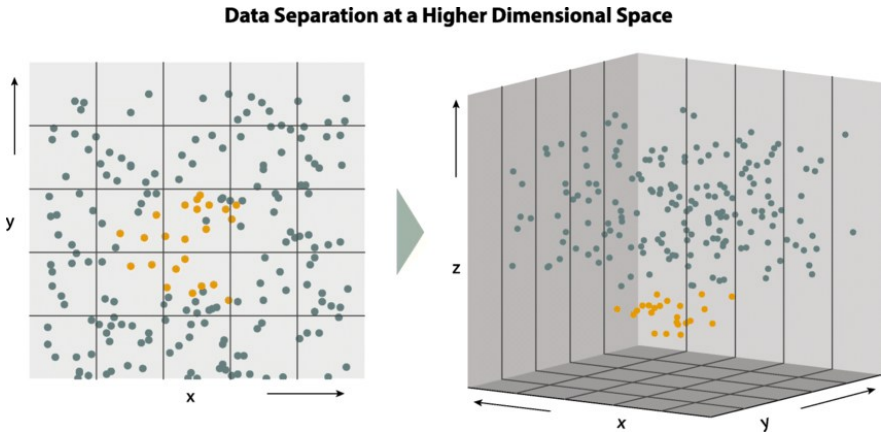


Figure 1.12 Data point separation in different search space dimensions. Data points identified as yellow and olive dots are separated into two-dimensional spaces. The lower dimension (**left**), consisting of the x and y scoring functions, shows a simple separation. The higher dimension (**right**) consists of the z scoring function in addition to x and y, which increases the complexity of the search space environment. This unmodified figure is from Juarez-Orozco *et al*. (2018).

Non-linear models are capable of separating data points with greater complexity in high-dimensional environments. Although there is no theoretical limit to the number of dimensions, an overly complex model can reduce the similarity between data points of the same class, while insufficient model complexity can lead to biased discrimination. Thus, controlling dimensionality is critical for optimising the performance of a learning algorithm. In the next chapter, the issue of dimensionality is further explored, along with methods for improving the performance of classification algorithms.

1.6.1 Overfitting and underfitting

In supervised or semi-supervised classification tasks, a learning algorithm is expected to learn the trends of feature variables that separate data points by their labelled class types. Then the patterns are inferred onto unlabelled data points, categorised by only the feature vectors. The performance of the model's ability to discriminate between class types is determined by the frequency of errors when separating data points (Anderson *et al*., 2003; Sampson *et al*., 2011). While it may appear that a good model is one that can confidently identify each class type, excessively high accuracy can indicate overoptimistic classifications. To illustrate this, **Figure 1.13** visually represents binary classification performance as three possible outcomes: underfitting, overfitting, or a good fit.

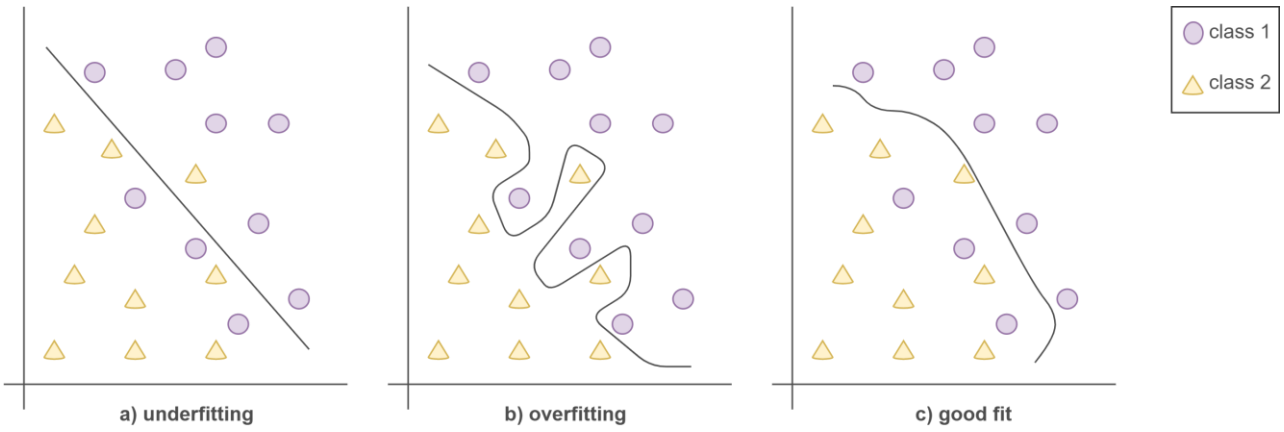


Figure 1.13 Three types of classification performance outcomes. Two class labels are separated by a learning algorithm performing binary classification. The line drawn through the data points in each illustration represents the model's decision-making process during binary classification. **(a) Underfitting:** The model oversimplifies. **(b) Overfitting:** Too many redundancies are influencing the classification. **(c) Good fit:** The model generalises well.

An algorithm may fail to capture the detailed patterns of a dataset, oversimplifying the learning process. This typically occurs when the dataset provides a poor representation of the data points (Dincer *et al*., 2022), leading the model to underfit (**Figure 1.13[a]**) and create an overly simplistic distinction between class types. A biased classification is observed due to the algorithm's inability to learn the variance present within the dataset's patterns. The trade-off between bias and variance during the learning (training) and classification (testing) processes is visualised in **Figure 1.14**. High-bias classification is associated with unskilled or low-complex models, producing high volumes of errors during both training and testing (Pothuganti, 2018).

On the contrary, an algorithm that overly learns the complexities of a dataset results in overfitting, as illustrated in **Figure 1.13[b]**. This can occur in PSM datasets with high levels of contaminants or poorly fragmented peptides, leading to noisy random spectra (Anderson *et al*., 2003), and in large-scale datasets, such as MS/MS proteomic data (Frank, 2009). An overly complex or highly skilled model (**Figure 1.14**) is sensitive to these fluctuations, causing high variance unrelated to the true patterns of the data points. Note that as model complexity increases in **Figure 1.14**, the training performance appears to generalise well, yet the testing process produces little to no errors. A PSM dataset classified under such conditions would be unreliable, as the expectation of no errors is unrealistic given the irregularities present in the data.

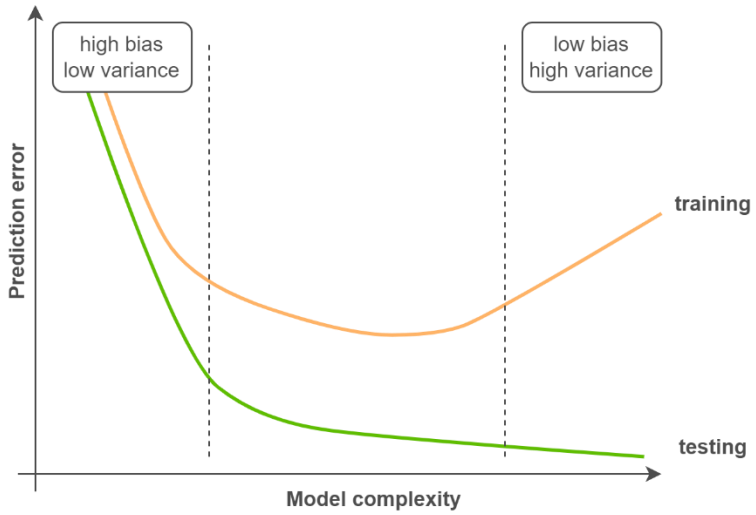


Figure 1.14 Prediction error vs. model complexity. The degree of bias and variance is demonstrated for a classification model's training and testing process. A low-complex model is visualised to the left of the first dashed line. A high-complex model is visualised to the right of the second dashed line.

Classified data points are more reliable when produced by an algorithm displaying appropriate discernment (**Figure 1.13[c]**) (Anderson *et al*., 2003). A model that is efficiently trained can generalise well with a moderate level of complexity (**Figure 1.14**) and produce a low volume of errors during the testing process.

1.6.2 Creating a balanced learning model

1.6.2.1 Hyperparameter tuning

A model's classification ability is influenced by various factors occurring during the training process. Configuring certain attributes of an ML architecture can potentially resolve the issue of overfitting and underfitting. The similarities between data points are measured by their distance from one another within the search space. Unsimilar data points, such as two class labels (**Figure 1.12**) with distinct identifying patterns, will be positioned further apart, whereas those sharing the same label will cluster together (Ghaddar and Naoum-Sawaya, 2017). A high-dimensional search space can negatively impact the classification of data points that are likely similar yet are distanced by the environment. A wider search space further increases the occurrence of variance (Bai and Saranadasa, 1996) and leads a model to overfit.

Not all features significantly impact prediction values, and their importance is graded by the ML architecture. To mitigate overfitting, feature variables that contribute poorly to the model's learning algorithm can either be removed or have their influence limited in the classification process by penalising their significance (Deng and Runger, 2012; Blanco *et al*., 2018). This is achieved through regularisation, a technique used to control a model's complexity. Configuring regularisation is not straightforward, as minor influencing feature variables can in fact improve the classification separation of data points. Several metrics, such as Lasso and Ridge (Tibshirani, 1996; Muthukrishnan and Rohini, 2016), are available to control this; however, the statistical details covering these methods are beyond the scope of this work.

Alternatively, model complexity can be controlled with hyperparameter tuning. ML architectures are designed using internal values, known as parameters, which define how the algorithm processes information. Parameters are unalterable functions that decipher the patterns of feature variables characterising class labels and update the algorithm's learning function automatically during the training process (Breiman, 2001). Hyperparameters are external values configured prior to training to control the robustness of the learning algorithm (Sipper, 2022). There are numerous types of hyperparameters, and each architecture consists of a unique set related to its learning algorithm (Yang and Shami, 2020). Hyperparameter tuning is the process of finding the best value for a combination of hyperparameters.

In the example diagram **Figure 1.15**, four arbitrary hyperparameters are listed alongside a set of values specific to their modes of operation. During hyperparameter tuning, the training process repeats for every combination of variable settings. This process is computationally expensive, and more so when large-scale databases require processing (Ali *et al*., 2023).

Parameter '*C*' for example, has a range of 10 – 500; therefore, every integer within this interval will be tested in combination with the other variables of the other hyperparameters. The combination producing the highest classification accuracy can optimally balance bias and variance, thereby preventing overfitting and underfitting. However, tuning too many hyperparameters with broad variable ranges can limit the learning potential of the algorithm and inadvertently induce overfitting and underfitting (Zimmermann *et al.*, 2023). A practical solution is to configure the model initially with a smaller selection of hyperparameters and reduced integer ranges, expanding the search space later if necessary.

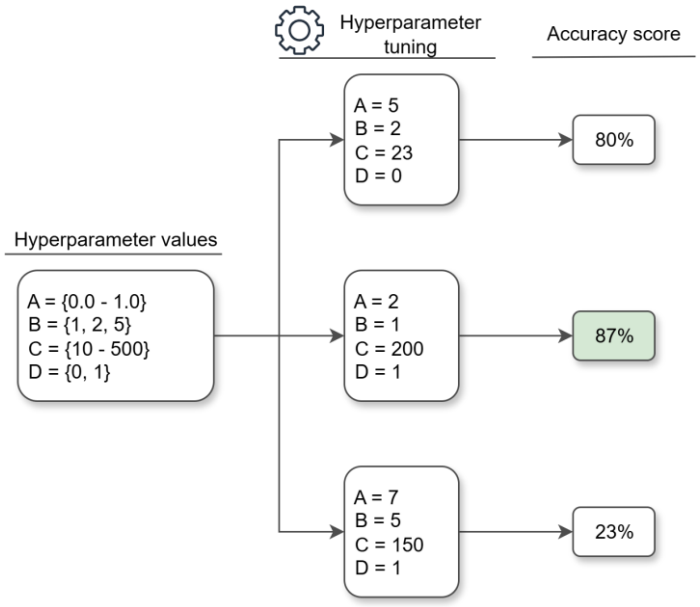


Figure 1.15 Example of a hyperparameter tuning method. Hyperparameters A, B, C and D are listed with several variables that adjust the robustness of a learning algorithm. All possible combinations of variables are screened for accuracy. The best-performing combination of variables for each hyperparameter is determined by the highest accuracy (green box).

1.6.2.2 Stratified nested k-fold cross-validation

A typical supervised learning algorithm requires one dataset for training the learning algorithm and another for classification. However, an issue with using different PSM datasets for training and classification is that they are not necessarily representative of each other. Experimental sample preparation varies across laboratories, and certain variables, such as enzyme type, modified peptides, and mass spectrometry calibration, are difficult to account for during downstream processing (Lam *et al.*, 2007). PSM identification is also dependent on the experimental procedure conducted to harvest specific peptide sequences (Spahr *et al.*, 2001; Lycette *et al.*, 2016). Decoy spectra propagation and search engine scoring functions measuring

annotations are therefore not uniform, making the qualities of one PSM dataset untransferable when classifying another. This disparity is also highly likely to cause overfitting. An alternative solution (as proposed in the work of Percolator (Granholm *et al*., 2012)) is to use *cross-validation*.

The term cross-validation was coined by Stone (1974), who identified alternative forms of Monte Carlo predictive subsampling provides a means to estimate error rates by omitting a sequence of single subsamples from contextual data (Geisser, 1975). An example of single random sub-sampling is the traditional ML method of splitting data into training, validation and testing sets (Bai *et al*., 2021). A learning model is trained to learn the underlying patterns of the dataset using the training sample; the validation portion then assesses the quality of its classification performance before the testing sample is finally classified. Typically, the testing subset comprises 10 – 30% of the dataset (Berrar, 2018). However, this approach is not feasible when processing PSMs as it would result in the direct loss of unclassified spectra held out for the training and validation samples. A simple solution is to apply the modern form of cross-validation, which has been redeveloped for regression analysis models (Picard and Cook, 1984).

Classification tasks primarily adopt the *k-fold cross-validation* method (Xu and Goodacre, 2018); **Figure 1.16** details the method. The principle of the technique is to split the dataset into subsets or *folds*. The number of folds is determined by the k value, which in this example $k = 5$. A single fold $\{k - (k - 1)\}$ is held out for testing, and the remaining folds $\{k - 1\}$ are used to train the learning algorithm. This process is repeated for k iterations, ensuring that each fold is used exactly once for classification.

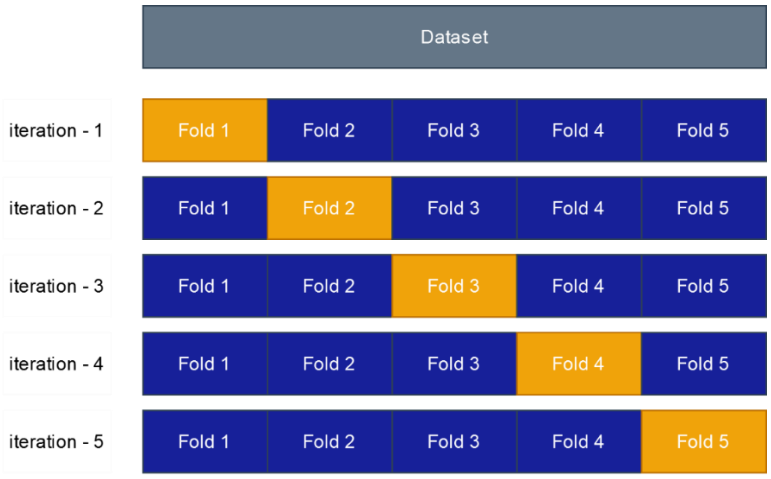


Figure 1.16 Cross-validation method. A dataset undergoes 5-fold cross-validation and is divided into five folds. The process repeats for five iterations. In each iteration, the blue folds are used for training a learning algorithm, and the orange folds are held out for classification.

With this approach, the learning model functions as a self-trainer (see Chapter 1.5, **Table 1.2**) and the dataset is classified using its own family of data points. The number of k-folds can range from 2 to n, where n is the total number of data points in a dataset (Marcot and Hanea, 2021). It is recommended to use a higher number of folds for smaller datasets, e.g. less than 100 data points (Isaksson *et al.*, 2008). In such cases, 10-fold cross-validation (Kohavi, 1995) or leave-one-out ($n - 1$) (Gronau and Wagenmakers, 2019) is commonly suggested. For larger datasets, such as PSM datasets containing over a million spectra, a smaller number of folds (typically 3 – 5) is sufficient. This is owing to there being enough data points to efficiently represent the training data for the learning model (Yadav and Shukla, 2016).

The dataset is split into subsets with an equal distribution of data points, e.g. a dataset consisting of 10 data points divided into 5 folds would allocate 2 data points per fold. In the case of PSM datasets composed of target and decoy spectra with uneven class frequencies, it is necessary to ensure that each fold receives a proportionate distribution to avoid biased classification. To achieve this, data points undergo *stratification*. This process is illustrated in **Figure 1.17** in which nine data points are split into three folds, each containing three data points in a 3-fold cross-validation setup. Despite varying class frequencies, stratification ensures balanced representation within each fold.

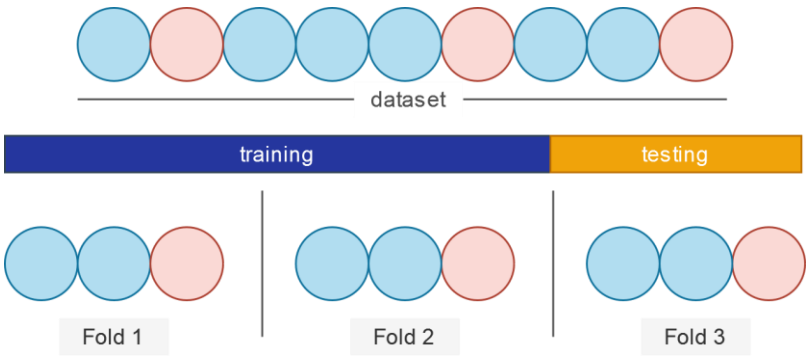


Figure 1.17 Stratification in k-fold cross-validation. An imbalanced dataset consisting of nine data point class types is stratified in a 3-fold cross-validation setup. Each fold receives one of class type 1 (red) and two of class type 2 (blue) to create an equal divide.

The best combination of hyperparameters (see Chapter 1.6.2.1) can improve a model's learning accuracy; however, it is acknowledged that this is an overly optimistic estimation. Error rates tend to be underestimated, leading to an issue known as *multiple comparisons* in statistical hypotheses testing (see Chapter 1.5.2). To navigate around this concern, it is recommended to perform hyperparameter tuning on a smaller sample of the dataset using a method known as *nested k-fold cross-validation* (Rakhshani *et al.*, 2015). This process is illustrated in **Figure 1.18**,

which is an extension of the *5-fold cross-validation* example. In this approach, a nested second-level cross-validation is applied to the training folds $\{k - 1\}$ of the first-level cross-validation. The number of folds used in the outer and nested cross-validation process does not need to be the same; however, in this example, a *nested 5-fold cross-validation* is used for uniformity.

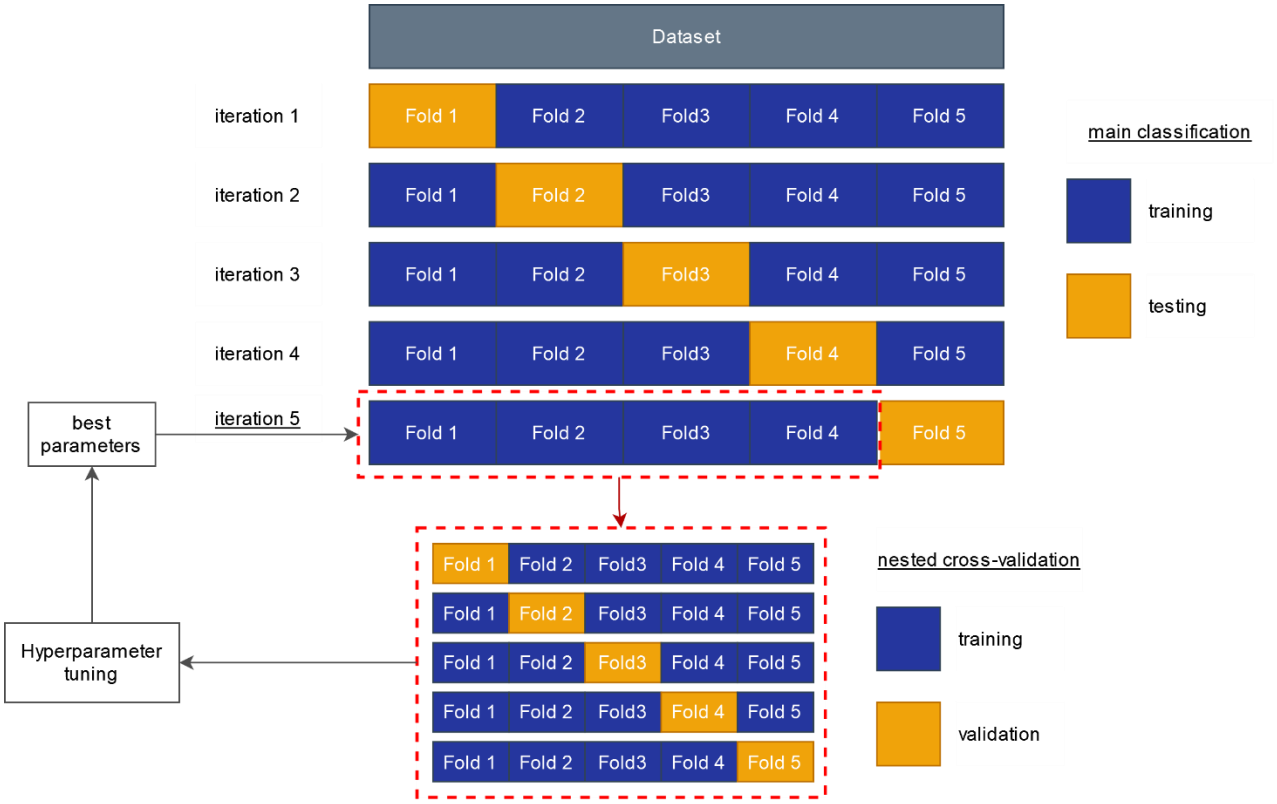


Figure 1.18 Nested k-fold cross-validation method. Two layers of cross-validation is created. The second layer is created using the training folds of the first layer. Hyperparameter tuning is performed within the second layer of cross-validation. The best parameters are used to train the learning model in the first layer for the main classification.

Hyperparameter tuning is conducted for each iteration of the nested cross-validation. The training folds are further divided into new training folds $\{k - 1\}$ and the remaining fold $\{k - (k - 1)\}$ is for validation. The best hyperparameters are then passed back to the learning algorithm of the first level, where the main classification and performance evaluation take place. This nested cross-validation process is repeated for each iteration of the first-level cross-validation. Consequently, the selected hyperparameters correspond only to the training folds of each specific iteration, i.e. the hyperparameters tuned during iteration 1 are not carried over to iteration 2.

Although hyperparameter tuning can help a model achieve a good fit, accuracy scores alone are insufficient for determining model performance. ML algorithms performing binary classification infer a test statistic, i.e. a probability score to indicate the degree of confidence each data point belongs to a particular class. For instance, if 8 out of 10 data points are correctly predicted, the overall accuracy score would be 80%. Another model classifying the same data points could achieve an accuracy of 85%, and so forth. Regardless of whether an individual data point receives a 51% or 95% probability of being a certain class type, both would be assigned the same predicted class label if the decision threshold (commonly 50%) is exceeded. Therefore, accuracy alone is uninformative, as it does not reflect the degree of certainty, nor provides statistical context regarding the correctness of individual predictions.

Furthermore, a global accuracy score cannot justify, nor confidently ensure which classifier produces more reliable predictions (Ling *et al.*, 2003). Owing to this, the performance of a learning model is more appropriately determined by the frequency of errors (see Chapter 1.6.1) made during classification tasks. The TDA (see Chapter 1.5.1) can be effectively employed to infer error rate metrics for PSMs and in so doing, describe the performance of learning algorithms, and more importantly, characterise spectra.

1.6.3 Determining model performance

1.6.3.1 Confusion matrix

To examine the behaviour of binary-classified data points, a 2 × 2 confusion matrix (or error matrix) can be used (Starovoitov and Golub, 2020). This matrix measures the distribution of class labels after prediction by reporting the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) as illustrated in **Figure 1.19**. Ordinarily, the matrix is a comparison of the actual class labels originating from the dataset against the predicted labels generated by the learning model. However, this framework can be specifically tailored for PSM processing. Since the goal of PSM processing is to identify random spectra within the target PSMs (see Chapter 1.5), the predicted labels themselves are less relevant. Therefore, hereon the process will specifically describe target and decoy spectra.

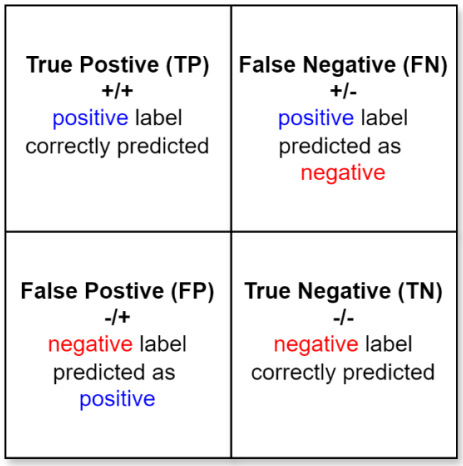


Figure 1.19 Confusion matrix. The binary class labels, positive and negative, are distributed into four statistical outcomes following classification.

Similar to the FDR and PEP, the confusion matrix factors in the benefit of ranking the PSMs (**Figure 1.11**) by a probability of fitness. In the context of **Figure 1.19**, the confusion matrix definitions are adapted as follows: target spectra are considered the *positive* class type, and decoy spectra the *negative* class type. The test statistic assigned to the classified spectra by the learning algorithm is used to rank the PSMs, as demonstrated with probability scores in **Figure 1.20**. Then, instead of using predicted class labels the frequencies of target and decoy spectra occurring at the rank of each spectrum are tallied. The TP, FP, TN and FN rates are calculated using the appropriate estimation strategies. The confusion matrix rates are typically estimated empirically or by using similar tactics altered by various statistical methods to compare effectiveness (Luque *et al*., 2019). An alternative approach will be explored in this work, and it should be noted that the calculations visualised in **Figure 1.20** are designed for the current project and will be referenced later in Chapter 2.8.2. After the spectra have been assigned the four confusion matrix rates, additional metrics prescribed for model performance evaluation can be computed.

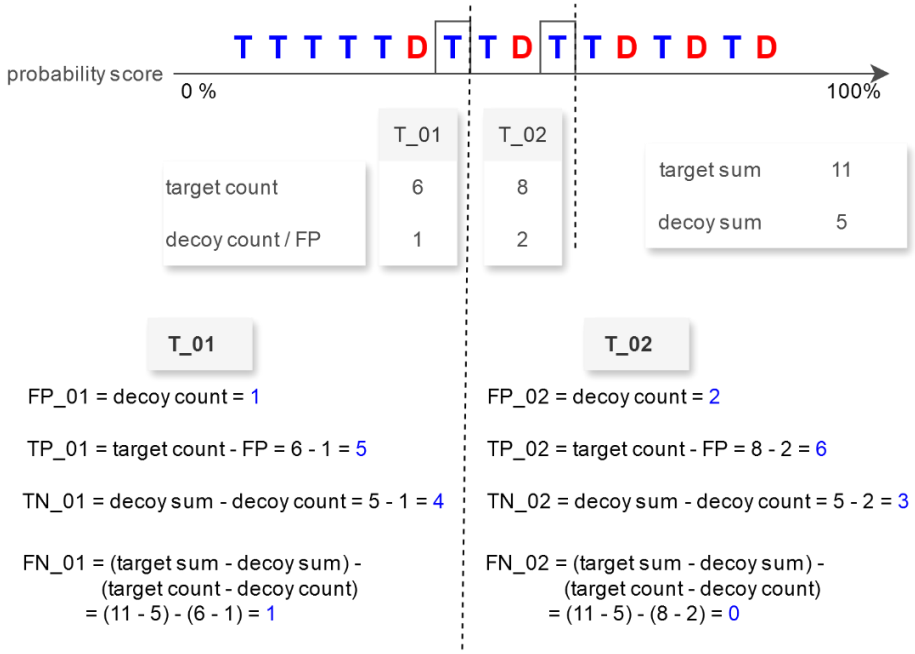


Figure 1.20 Confusion matrix concept simplified. A classified PSM dataset is ranked by probability scores. The confusion matrix rates are computed for two target spectra (blue 'T's in the outlined boxes) at different rank locations. The *target count* and *decoy count* are the number of spectra occurring up to the spectrum's rank indicated by the dashed line. The *target sum* and *decoy sum* are the total number of each class type in the dataset. The false positive, true positive, true negative and false negative rates are estimated by inserting these counts into the corresponding equations.

1.6.3.2 Model performance metrics

The error rates and distribution trends of classified data points can be visualised graphically to empirically assess a learning algorithm's performance. Two common methods for this are the area under the receiver operating characteristic (AUROC) curve and the precision-recall (PR) curve. The receiver operating characteristic (ROC) curve is a standard graphical illustration applied in statistics to analytically evaluate the performance of binary classification. Originally developed for warfare to discern real enemy fire from false alarms (D. M. Green, 1970; P. E. Green, 1970), this method quickly found its way into medical diagnostic research in the 1970s (Lusted, 1971). The ROC curve compares the trade-off between the true positive rate (TPR) and false positive rate (FPR) (Huang and Ling, 2005). The TPR (referred to as *sensitivity* and *recall*) measures the proportion of correctly identified positive instances (TPs) relative to all actual positives, including incorrectly classified positive instances (TPs and FNs). The FPR (referred to as 1 – $specificity$) measures the proportion of negative instances incorrectly classified as a positive class (FP) against all negatives, including correct estimations of the negative instances

(FPs and TNs) (Van Stralen *et al*., 2009). In the context of PSM processing, positive and negative instances correspond to target and decoy PSMs, respectively.

The classification performance for each spectrum can be comprehensively visualised using a ROC curve, as illustrated in the example plot of **Figure 1.21**. A curve that lies close to the diagonal line (representing a random classifier) indicates the model has poor ability to discriminate between class types. The area under the curve (AUC) quantifies the ROC curve's performance: the closer the curve approaches the upper-left coordinate $(0, 1)$, the higher the AUC, which reflects strong model performance (Nahm, 2022). The AUC score is generally considered more informative than the accuracy score issued by a learning algorithm (**Figure 1.15**), as it incorporates the ranking of spectra based on the probability of being a true match (Ling *et al*., 2003). The reliability of true matches estimated by a model can be ensured by a well-performing model measuring the error rates in the same manner. The combined use of AUC and ROC (referred to as the AUROC curve) enables effective comparison of multiple model performances.

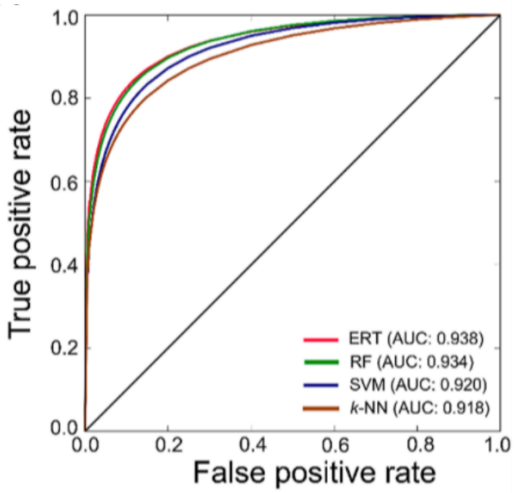


Figure 1.21 Comparison of architectures using the AUROC curve. The ROC curve is the trade-off between the true positive rate and the false positive rate. Four ROC curves, each representing a different machine learning architecture, are distinguished by colours. Model performance is compared using their respective AUC scores. The black diagonal line represents the performance of a random classifier. This unmodified figure is from Manavalan *et al*. (2018).

The AUROC curve is prone to overlooking class imbalances as it focuses on the learning model's ability to distinguish between positive and negative class types, without accounting for their relative frequencies (Saito and Rehmsmeier, 2015). Consider for example, an FP target spectrum ranked at position 8, preceded by 7 TN decoy spectra. Although this reflects poor

classification performance it would still result in a low FPR, which the AUROC metric may not penalise adequately. Such scenarios are common when processing a large-scale PSM dataset, where the prevalence of negative cases is typically high (Saito and Rehmsmeier, 2015). To address this limitation the AUROC is often complemented by the PR curve. Similar to the ROC curve, the PR curve incorporates *recall* (which is the same as *sensitivity*) against *precision*, which is a measurement of the positive instances (TPs) among all predicted positives, including FPs (Liu and Bondell, 2019).

The performance of two learning algorithms compared using the ROC coupled with the PR curve is demonstrated in **Figure 1.22**. The PR curve mirrors the ROC, meaning the higher the curve is to the top right towards the (x, y) coordinate $(1, 1)$ (top-left for an ROC plot), the better the performance of the model. In this example, the two algorithms appear to perform well in the ROC plot yet exhibit poor precision in the PR plot. As both metrics are computed using ranked classified PSMs, the correlation between precision and accuracy can be directly inferred. Generally, an accuracy score similar to the AUC accompanies the PR curve (Sofaer *et al*., 2019); however, the observation of trends is sufficient to establish model performance.

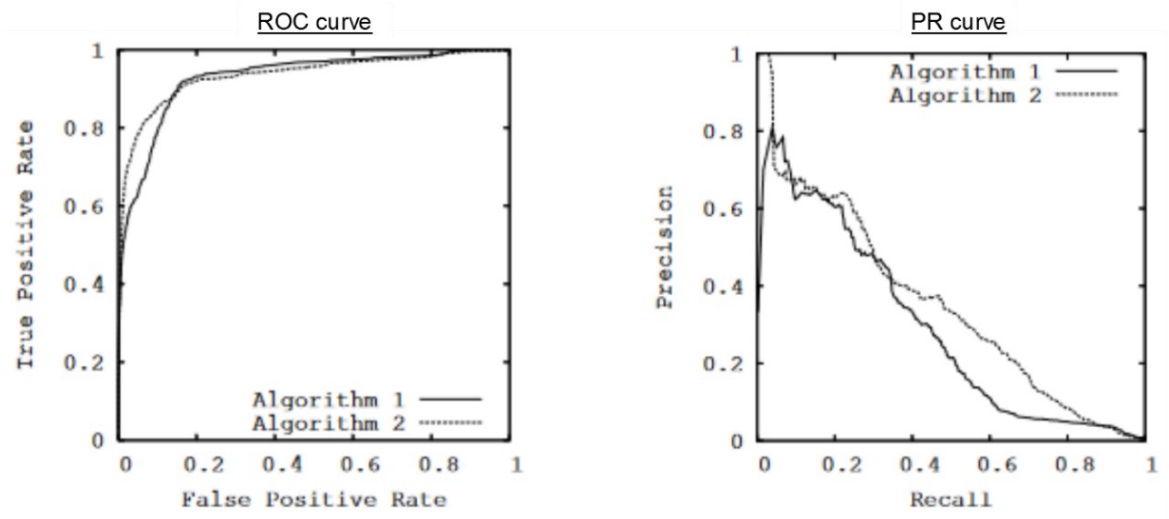


Figure 1.22 Relationship between a ROC and a PR curve. The performance of two algorithms is observed in each graph. The PR curve displays poor precision as a means to compare to their well-performing ROC curves. This unmodified figure is from Davis and Goadrich (2006).

The AUROC and PR curves have been applied widely in biological studies whose methodologies include statistical inference. A study by De Smet *et al*. (2004) for example, identified an increase in FNs in large-scale leukaemia biopsy research and explored balancing error rates using ROC curves. Pang *et al*. (2021) applied both AUROC and PR metrics to evaluate classification performance in the computational identification of rare anti-coronavirus peptides from

imbalanced datasets. Regarding PSMs ranked by a probability metric indicating fitness, the ROC and PR curves are indicative of spectra behaviour. It is to be expected that a higher frequency of non-random target PSMs would incur strong probability scores reflecting a high TPR and low FPR. Conversely, an increasing FPR would be synonymous with target PSMs having a strong likelihood of being an incorrect match. Therefore, the curve of the ROC should resemble a well-performing model as visualised in **Figure 1.21**, which would be validated by the PR curve. In this chapter, the method of processing PSMs using ML was introduced. The next chapter will discuss existing tools and methods.

1.7 Machine learning in practice as a post-processing tool

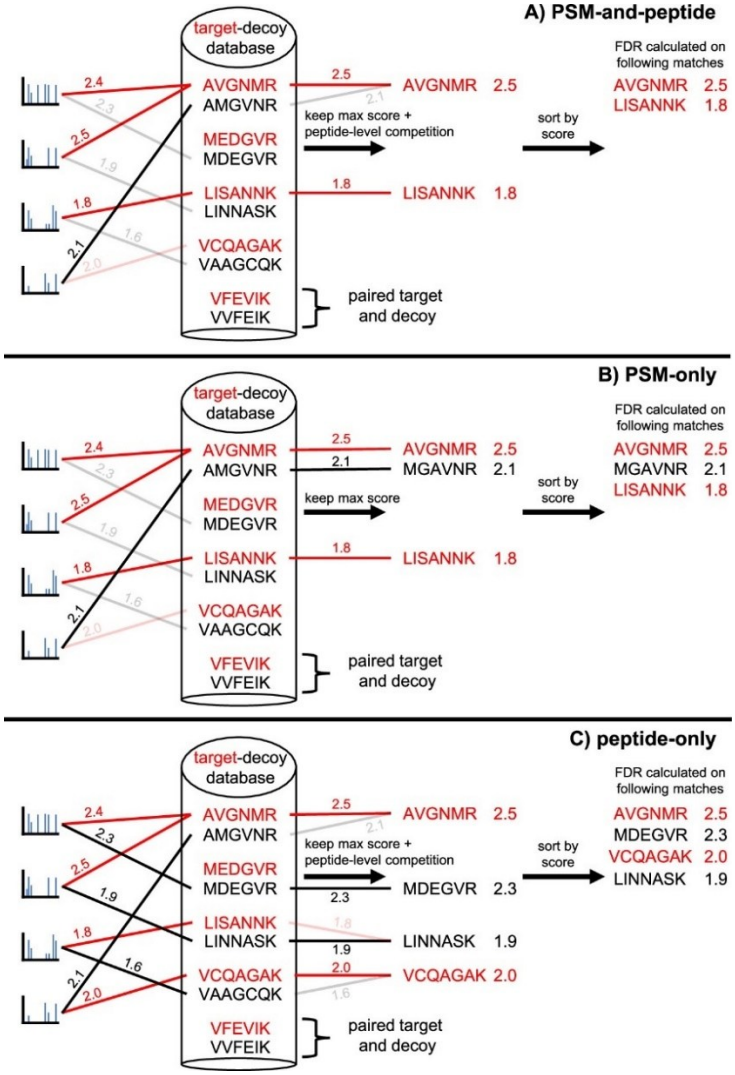


Figure 1.23 Difference between PSM-level and peptide-level processing. Mass spectrometry spectra can undergo error rate performance evaluation at three stages. **(A) PSM-and-peptide**, **(B) PSM-only**, and **(C) peptide-only**. This unmodified figure is from Lin *et al.* (2022).

Processing PSMs in proteomics addresses the challenge of distinguishing true peptide identifications from false positives. Traditional search engines, such as SEQUEST (Eng *et al*., 1994) and X!Tandem (Craig and Beavis, 2004), generate vast amounts of PSM data, often leading to high rates of false matches due to noise, ambiguous spectra, or extensive database size (see Chapter 1.4). Modern methods have been created to filter out low-quality spectra either at the PSM- or peptide-level. In **Figure 1.23** a PSM dataset consisting of target and decoy spectra is used to annotate MS/MS peptide sequences.

At the PSM-level (**Figure 1.23[B]**), the overall highest-scoring alignment to either a target or decoy spectrum is retained during search engine processing. A widely used PSM-level tool is DTASelect (Tabb *et al*., 2002), and there are more recent ventures, such as Crema (Lin *et al*., 2024) that measures the FDR using target-decoy tactics. Such methods are suspected to be indirectly biased (He *et al*., 2015). Peptide-level TDA methods (**Figure 1.23[C]**) are less commonly applied and select the highest-scoring target and decoy alignment for each observed spectrum, from which the top-scoring alignments are retained. Peptide-level filtering ignores the relationship between target and decoy spectra during the annotation process when computing error metrics, such as the FDR (Li *et al*., 2017).

These approaches are reliant on statistical measures to be accurately defined to manage the nuance of inferring significance in peptide identification tasks. Alternatively, it has been suggested that combining PSM-level and peptide-level processing (**Figure 1.23[A]**) is the most effective approach for producing substantial amounts of identified peptides. Research on improving FDR application on *E.coli* peptides from the work of Lin *et al*. (2022) reported a performance improvement of over 40% for the combined approach compared to the PSM-only method. Existing PSM processing tools, such as Scaffold (Searle, 2010) and IDPicker (Ma *et al*., 2009), explore the potential of extending analysis from the PSM-level to protein-level validation.

Although these tools are well-established, ML offers an alternative tactic by combining the multiple scoring functions from search engines into a single interpretable test statistic. One of the earliest ML-based tools, PeptideProphet (Keller *et al*., 2002), employed Bayesian inference to construct a statistical model for ranking annotated peptides and assessing their validity. Subsequently, Anderson *et al*. (2002) applied a SVM architecture to discriminate between correctly and incorrectly identified PSMs derived from SEQUEST. In their approach, incorrect peptide sequences were created experimentally and selected from low-scoring annotations. Later, decoy PSMs were introduced to enact as incorrect sequences, and the TDA was implemented in newer tools, such as Percolator (Käll *et al*., 2007). The workflow of Percolator is of particular interest as its design inspires the methods of this work.

1.7.1 Percolator

Percolator (Käll *et al.*, 2007) is a semi-supervised (see Chapter 1.6, **Table 1.2**) SVM classifier system designed to discriminate between target and decoy spectra. In its methodology, the annotated target and decoy spectra are first refined using 20 scoring functions (**Figure 1.24**) derived from SEQUEST (Anderson *et al.*, 2002) and other statistical measures. The workflow of Percolator (**Figure 1.25**) begins with ranking the combined target and decoy PSMs using a single scoring function. Target spectra with a 1% FDR (see Chapter 1.5.3) are then collected from the ranked spectra. The SVM learning algorithm operates as a self-trainer by implementing a 3-fold cross-validation scheme to distinguish the selected target spectra from the full-set of decoy PSMs. Nested 3-fold cross-validation hyperparameter tuning (see Chapter 1.6.2.2) is conducted during the training process. The testing folds (**Figure 1.16**) are normalised and merged using the SVM's probability scoring method, which is used to re-rank the classified PSMs.

1	XCorr	Cross correlation between calculated and observed spectra
2	ΔC_n	Fractional difference between current and second best XCorr
3	ΔC_n^L	Fractional difference between current and fifth best XCorr
4	Sp	Preliminary score for peptide versus predicted fragment ion values
5	ln(rSp)	The natural logarithm of the rank of the match based on the Sp score
8	Mass	The observed mass $[M+H]^+$
6	ΔM	The difference in calculated and observed mass
7	abs(ΔM)	The absolute value of the difference in calculated and observed mass
9	ionFrac	The fraction of matched b and y ions
10	ln(NumSp)	The natural logarithm of the number of database peptides within the specified m/z range
11	enzN	Boolean: Is the peptide preceded by an enzymatic (tryptic) site?
12	enzC	Boolean: Does the peptide have an enzymatic (tryptic) C-terminus?
13	enzInt	Number of missed internal enzymatic (tryptic) sites
14	pepLen	The length of the matched peptide, in residues
15–17	charge1–3	Three Boolean features indicating the charge state
18	ln(numPep)	Number of PSMs for which this is the best scoring peptide.
19	ln(numProt)	Number of times the matched protein matches other PSMs.
20	ln(pepSite)	Number of different peptides that match this protein.

Figure 1.24 Scoring functions used in Percolator. This unmodified figure of the twenty scoring functions is in the supplementary documents of Percolator (Käll *et al.*, 2007).

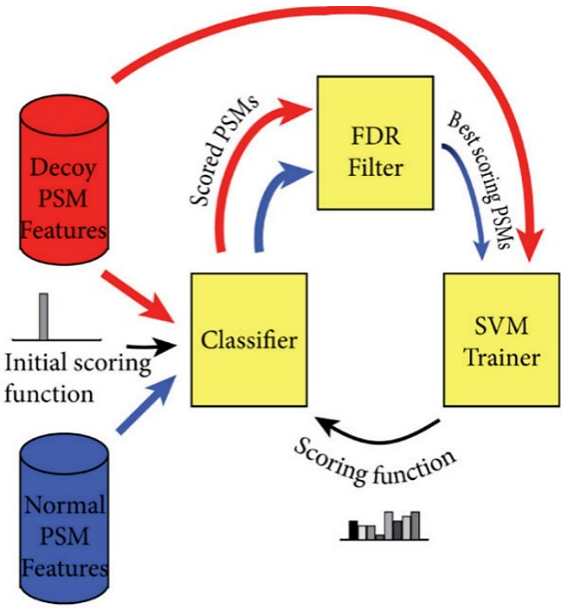


Figure 1.25 Learning algorithm of Percolator. The workflow of Percolator operates through an iterative loop involving three main components. Its process starts with ranking a PSM dataset of target and decoy spectra using an initial scoring function within the *Classifier*. An *FDR filter* is then used to extract target spectra for the *SVM trainer*. This unmodified figure is from Halloran *et al*. (2019).

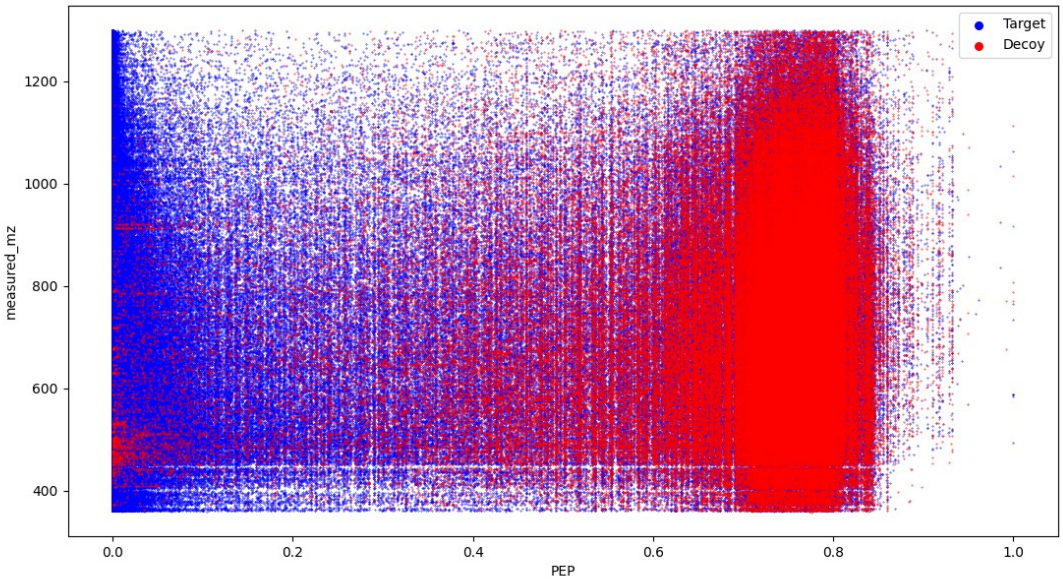


Figure 1.26 Performance evaluation of Percolator. Target and decoy spectra, colour coded according to the legend in the upper-right corner, were classified by Percolator. Two scoring functions, *measure_mz* and *PEP*, were used to observe the model's ability to distinguish between the spectra.

This process repeats until there is no further change in the ranking. The final product is the target and decoy spectra classified with p-score estimates, and additional metrics (such as the PEP and q-values) are estimated for downstream validation. A preliminary run of Percolator was conducted during this research to demonstrate its classification performance, which is visualised in **Figure 1.26**. This figure observes an accumulation of target spectra with significant PEP scores, and a low frequency of decoy spectra amongst its population. While the separation between target and decoy spectra is evident, it remains an open question whether there is room for improvement in classification. This work does not aim to directly compare against Percolator or other existing methods, but rather to explore the nature of PSM processing to improve variant PSM classification resolution.

1.7.2 Machine learning for variant PSMs

Although existing PSM processing tools are well-established, they are not specifically designed to identify mismatches in variant peptides. The high similarity to canonical peptides and low frequency of variant peptides make them difficult to isolate at the peptide-level. A few existing methods by Li *et al*. (2011), Tanner *et al*. (2005) and Sheynkman *et al*. (2014) focus on identifying post-translational markers of variant peptides during MS/MS research. Generally, these methods specifically generate variant peptide sequences through experimentation, which are then annotated with variant peptide genome databases to improve identification.

Here in this work, alternative machine learning architectures will be explored for their ability to discriminate between target and decoy PSMs of variant peptides at the PSM-level. Specifically, the performance of modern decision tree ensemble machine learning architectures is investigated to determine their capability in processing PSMs. The model that performs better will be used to evaluate *decoy variant* PSMs.

1.8 Decision tree ensemble models

1.8.1 Ensemble models

A single learning architecture function ($y = f(x)$ from Chapter 1.6) is an example of an individual base learner. Ensemble learning is a term to describe the method of combining multiple base learners. The underlying principle of ensemble models is the 'wisdom-of-the-crowd', the notion that a group of independently trained learning algorithms is more likely to produce accurate predictions (Alizadeh *et al*., 2015; Elliott and Anderson, 2023). An elementary illustration of this concept is provided in **Figure 1.27**. The learning algorithms workflow (**Figure 1.27[a]**) is first trained to learn the patterns and features of labelled data points.

Unlabelled data points are then classified using the ensemble model (**Figure 1.27[b]**). In this diagram, three individual base learners produce different predictions for a given data point. The ensemble model aggregates these predictions and selects the majority vote to produce the final classification, completing the workflow of **Figure 1.27[a]**.

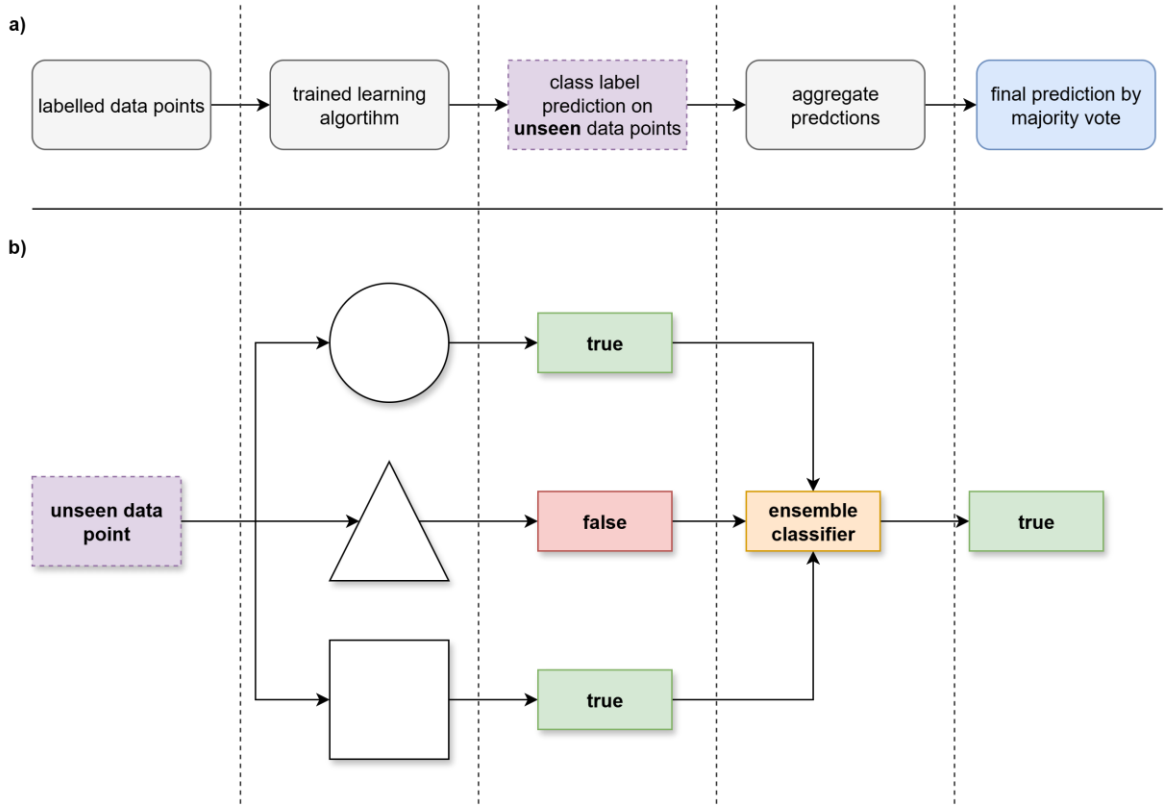


Figure 1.27 Ensemble learning algorithm methodology. (a) The workflow of ensemble architectures. The dotted lines align the stages of the workflow to the classification process. **(b) The classification process of ensemble architectures.** An unlabelled data point is classified by three individual base learners represented by different shapes. Each learner produces a prediction. The ensemble classifier aggregates the predictions. The final prediction is determined by majority vote.

1.8.1.1 Improving the performance of ensemble algorithms

In ensemble modelling, the performance accuracy of the individual base-learners is measured on a scale of 0.0 – 1.0, indicating poor to perfect accuracy. In standard binary classification tasks, an unlabelled data point is assigned a positive class label if it receives an accuracy score above 0.5. Therefore, whether a data point receives a score of 0.51 or 0.85 for example, it would be predicted as the positive class. A *weak learner* achieves an accuracy just above a random chance, i.e. slightly over 0.5 (Joshi *et al.*, 2002). In contrast, *strong learners* produce near-accurate

predictions, though they are more difficult to obtain and have a likelihood of being incorrect (Galton, 1907). To improve ensemble model performance, the *Boosting* method (introduced by Kearns (1994)), is employed. This tactic transforms learners from weak to strong by increasing their influence on the overall prediction accuracy of the ensemble base learners.

Another method to improve performance is the Bootstrapping and AGGregratING method (introduced by Breiman (1996)) abbreviated as *Bagging*. The goal of *Bagging* is to increase the independence of the base learners within an ensemble model as prediction accuracy tends to improve when individual learners produce diverse, uncorrelated predictions (Zhou, 2012). This is achieved by introducing randomness during training through bootstrapping (Efron and Tibshirani, 1993) or out-of-bag sampling. In this approach, data points are selected at random (with replacement) to train the individual base learners, allowing the same data points to be resampled across different learners.

The ensemble technique can be applied to modern ML architectures, such as neural networks (Rosen, 1996), SVMs (Shen and Chou, 2006) and decision trees (Geurts *et al*., 2005). Ensemble models can be created using a single type of architecture (homogeneity) or as a mixture of different types (heterogeneity). Heterogeneous ensembles are a form of combination or 'stacking' learning style in which base learners from different architectures are arranged in layers. The strongest learners from each layer are then combined to make a final prediction (Zhou, 2012). In this work, homogeneous ensemble learning models based on decision trees will be investigated for processing PSMs.

1.8.2 Decision trees

The decision tree (Quinlan, 1987) learning method is structured as a conditional flow diagram or *tree*, composed of a series of *nodes*, *branches* and *leaves*. The tree structure (**Figure 1.28[a]**) starts from a *root node*, which *branches* into *internal nodes* that *branch* and end in *leaf* nodes. The *root* and *internal nodes* pose conditional questions based on feature variables, while the *leaves* represent final predictions. The diagram in **Figure 1.28[b]** illustrates how a decision tree classifier system applies to PSM datasets consisting of class labels and feature variables. Each data point (spectrum) follows a *decision path*, i.e. the flow from the *root* through *internal nodes* to a *leaf node*, guided by a series of choices until it is predicted as either a target or decoy spectrum. In this example, two question *nodes* are created; however, the *tree* depth can be extended to increase the verbosity of the decision-making process. Attributes, such as the *nodes*, *leaves* and number of *trees*, are a few of the adjustable hyperparameters (see Chapter 1.6.2.1) controlling model complexity.

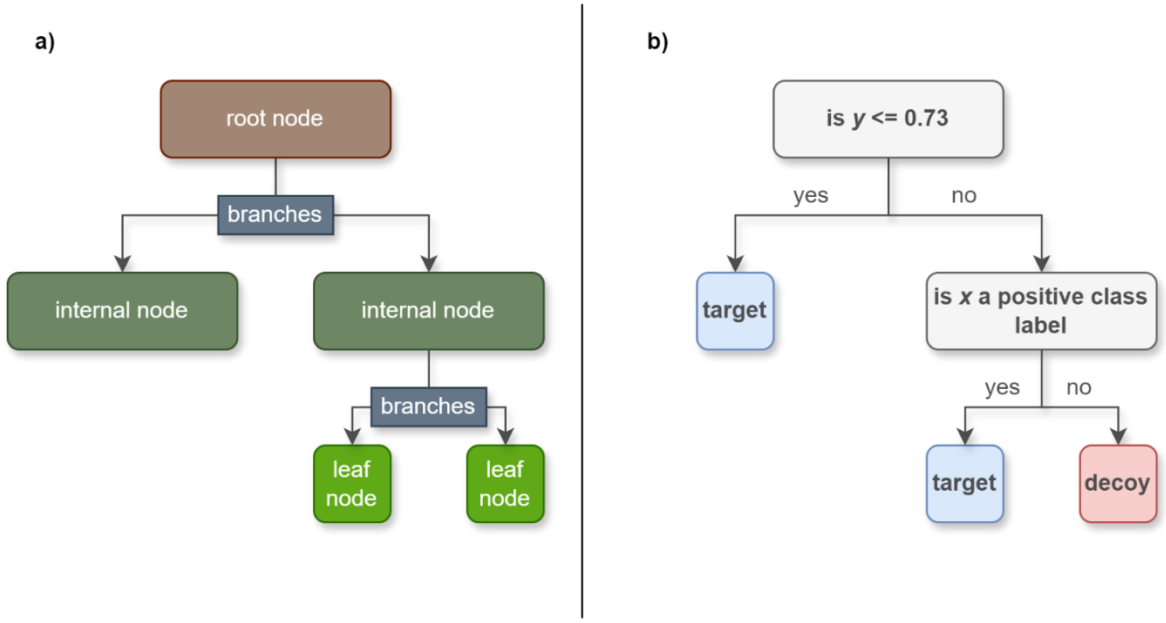


Figure 1.28 Decision tree classification method. (a) Structure of a decision tree. A decision path starts from the root node and follows a route of nodes until a prediction is made in a leaf node (Zhou, 2012). **(b) A decision tree processing PSMs.** Spectra (x) are classified as either a target or decoy class type following a decision path of nodes that describe scoring functions (y) using conditional statements.

There are several decision tree-based ensemble architectures, such as Random Forest (Kam Ho, 1995), Gradient Boosting (Friedman, 2001), Histogram Gradient Boosting (Guolin *et al*., 2017), Extra Trees (Guerts *et al*., 2006) and XGBoost (Chen and Guestrin, 2016). The ability to process PSMs will be compared for each of these architectures, which support both supervised and semi-supervised binary classification. Previous research, including studies by Sankari and Manimegalai (2017), Ahmad *et al*. (2021) and Krawczyk *et al*. (2014), have demonstrated that decision tree ensemble models are effective in handling imbalanced datasets. This characteristic may prove to be beneficial for PSM datasets with rare variant peptides occurring at a lesser frequency than their canonical counterparts. A brief description of each architecture will be introduced next.

1.8.2.1 Random Forest and Extra Trees

Feature importance is a metric measured by a learning algorithm that evaluates a dataset's feature variables' contribution to the prediction process. In Random Forest, feature variables are randomly selected to construct decision trees (Ho, 1995). Feature importance is then calculated based on the frequency and effectiveness each feature creates *branches* to determine the *best-split* (i.e. decision path (Breiman, 2001)), whilst reducing prediction errors. In **Figure 1.29** an example of three decision trees is displayed with varying levels of *tree* growth (depth). The decision trees of Random Forest are constructed independently from each other to ensure they are unrelated or *decorrelated* (Hastie *et al*., 2008), which in effect reduces the likelihood of overfitting. Extra Trees (shortened from 'extremely randomised trees') is a newer ML addition that is an aggressive form of Random Forest (Guerts *et al.*, 2006). In addition to Random Forest randomly selecting feature variables to compute the *best-split*, Extra Trees incorporates the entire dataset and sets feature thresholds to create the *branches* of the decision trees (Efron and Tibshirani, 1993).

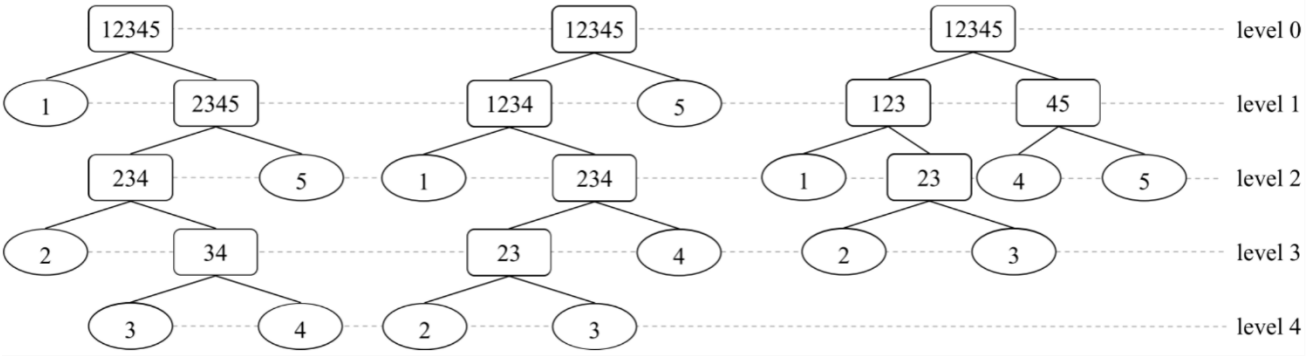


Figure 1.29 Random Forest algorithm method. Five data points are classified in three individual random decision trees. Each tree has a root and internal nodes (rounded squares) that create layers of branches. The data points are classified when they reach a leaf node (oval). This unmodified figure is from Zhou (2012).

Random Forest and Extra Trees are *Bagging* (Breiman, 1996) ensemble models that create fully-grown decision trees. This means the decision trees are permitted to develop without depth restrictions, imposing as many question nodes and decision paths as needed to produce the best prediction. Additionally, due to the inherent randomness introduced by feature selection during tree construction, these models typically do not require *pruning* (Qi, 2012), i.e. *branches* that contribute minimally to predictions are retained. Therefore, randomness is intentionally preserved to induce randomness and fully capture complex patterns in the training data.

1.8.2.2 Gradient Boosting, Histogram Gradient Boosting and XGBoost

The Gradient Boosting architecture creates an ensemble of decision trees by sequentially aggregating *weak learners* and iteratively correcting their prediction errors (Friedman, 2001). To achieve this, the Gradient Boosting method starts with a simple *weak learner* model low in prediction accuracy, as illustrated in **Figure 1.30**. The initial learning model does not efficiently capture the patterns of the training data. The algorithm then estimates *residuals* (Natekin and Knoll, 2013), which are error metrics measuring the differences between the actual and predicted class labels. A new learner is created with these *residuals* to reduce the errors from the previous learner. Each subsequent model is added to the ensemble to gradually improve the overall prediction performance. This iterative refinement process consecutively builds decision trees until the model converges or reaches a predefined stopping criterion. Histogram Gradient Boosting and XGBoost are advanced variants of the Gradient Boosting methodology, offering optimised implementations and improved scalability.

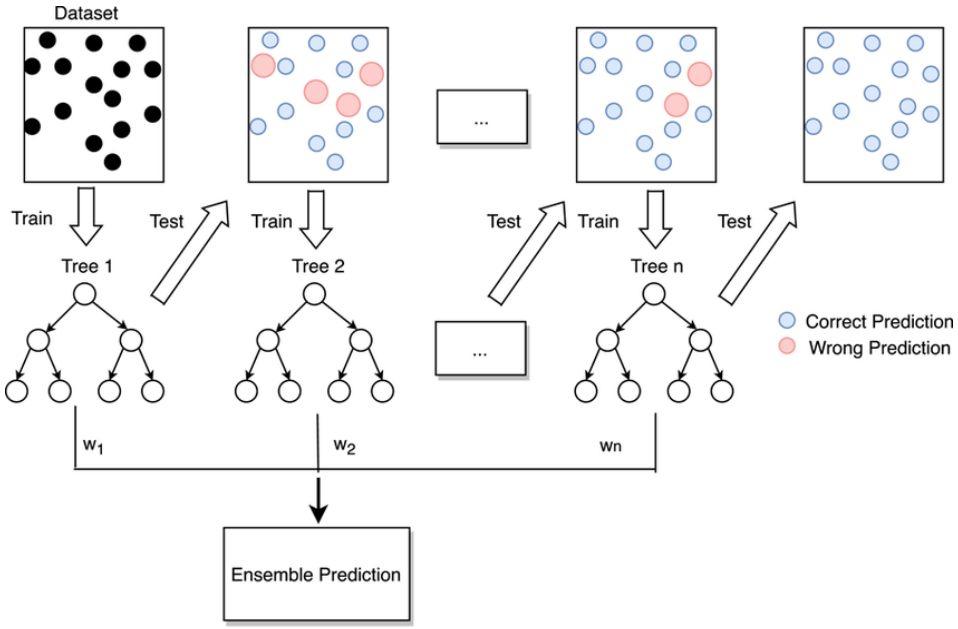


Figure 1.30 Gradient Boosting algorithm method. The process begins with a single decision tree producing a high frequency of incorrect predictions (upper left box with black circles). New decision trees are created until incorrect predictions (red circles) are reduced and correct predictions (blue circles) increase. The final prediction is made by aggregating all the decision trees (Ensemble prediction). This unmodified figure is from Zhang *et al.* (2021).

The Histogram Gradient Boosting architecture (Guolin *et al.*, 2017) compartmentalises feature variables into fixed intervals (known as *bins*) before training the learning algorithm. Data points are then assigned to a *bin* based on their feature values (Hossain and Deb, 2021). Decision tree *branches* are created using the *bins* to determine the *best-split* and estimate the *residuals* (errors). Developed by Microsoft as an element of their Light Gradient Boosting Machine library (Guolin *et al.*, 2017), this architecture gained prominence due to its ability to reduce computation time while maintaining high classification performance on large-scale datasets.

One of the key advancements of Extreme Gradient Boosting (XGBoost) (Chen and Guestrin, 2016) is the incorporation of regularisation to prevent overfitting, in which overly complex models are *pruned*, thereby enhancing generalisation performance. XGBoost applies two forms of regularisation: L1 (Lasso) and L2 (Ridge) (Chen and Guestrin, 2016; Cortes *et al*., 2019; Ahmad *et al*., 2021). The Lasso metric prunes *leaf nodes* by penalising feature weights with low contribution to the model, effectively removing irrelevant or redundant variables. The Ridge method discourages the learning algorithm from assigning large weights to any single feature variable, essentially promoting *weak learners* and preventing any individual tree from disproportionately influencing the ensemble's overall prediction outcome.

1.9 Project scope

Detecting variant and canonical proteins can be an effective strategy to determine the aetiology of monogenic paediatric diabetes; moreover, diagnose undetermined MODY cases. Here, it is proposed that investigating the performance of modern ML architectures to confidently predict canonical peptides can be further applied to predicting mismatched-variant peptides. To this end, a PSM-post-processing pipeline fondly named Nagilums Tree (NT) is created in this work, which follows the ML tactics explained in Chapter 1.6. The pipeline of NT is not a single tool, but a series of scripting files created to classify a dataset composed of target and decoy PSMs, conduct statistical inference and generate graphical visualisations of performance metrics.

The core structure of the classification system is described in **Figure 1.31**, which is purposefully designed to identify high amounts of canonical and variant PSMs. The *predictor function* and *counting function* are the main functional features serving to accomplish this objective. The classifier accepts a PSM dataset with associated scoring functions and ranks the spectra using a single function to estimate the FDR. The *predictor function* houses the main ML classification system. Its learning model is trained using target spectra filtered at a 1% FDR threshold and decoy spectra from the dataset, acting as a self-trainer system by utilising *k*-fold cross-validation.

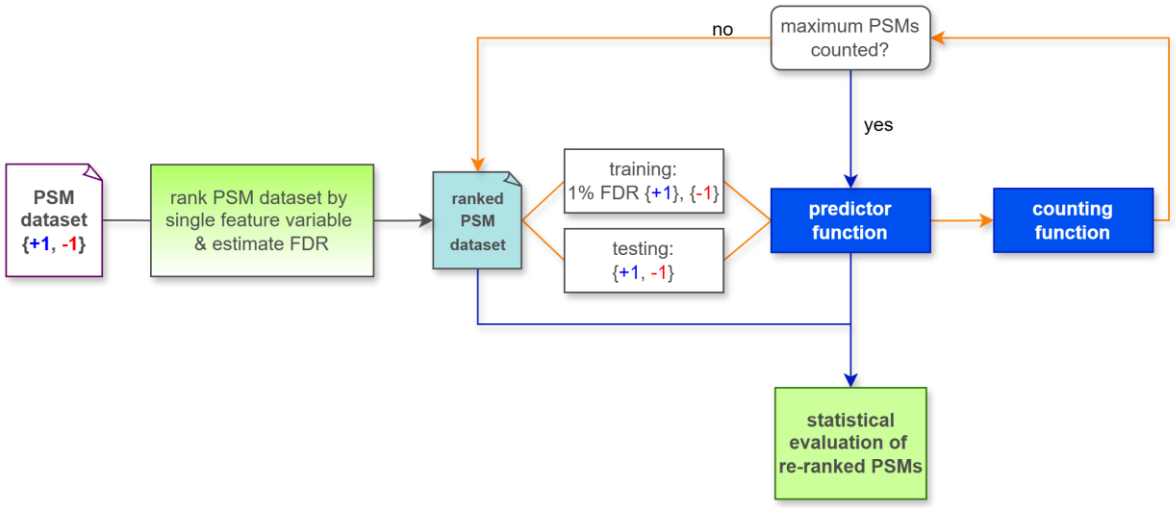


Figure 1.31 General schematic of the Nagilums Tree pipeline. A PSM dataset consisting of target {+1} and decoy {-1} labelled spectra, with feature variables, is processed. The spectra are ranked in ascending order based on a selected feature and classified within the *predictor function*. The orange path indicates a conditional loop controlled by the *counting function*. When the specified condition is met, the blue path is followed, and the model's performance is evaluated on the classified dataset.

The *counting function* monitors the number of confidently predicted significant target PSMs and initiates a conditional loop between itself and the *predictor function* until the maximum number of classified spectra is observed. After each iteration, the PSM dataset is re-ranked, and a new FDR is estimated to improve classification. The final ranking is then used for statistical evaluation. This setup is inspired by the Percolator pipeline (see Chapter 1.7.1) although differing in that its iteration aspect repeats until PSM ranking does not change, whereas in NT the loop serves to collect a high amount of significant target PSMs.

The NT pipeline is used to implement a workflow consisting of three tasks (**Figure 1.32**) with statistical inference and graphical visualisation specifically set up to each objective. The purpose of Task 1 is to investigate and compare the prediction performance of five decision tree ensemble architectures: Random Forest, Gradient Boosting, Histogram Gradient Boosting, Extra Trees and XGBoost. This task follows a standard PSM processing approach using the TDA, where the target PSMs include both canonical and variant peptide spectra. The best-performing architecture is then selected for Task 2.

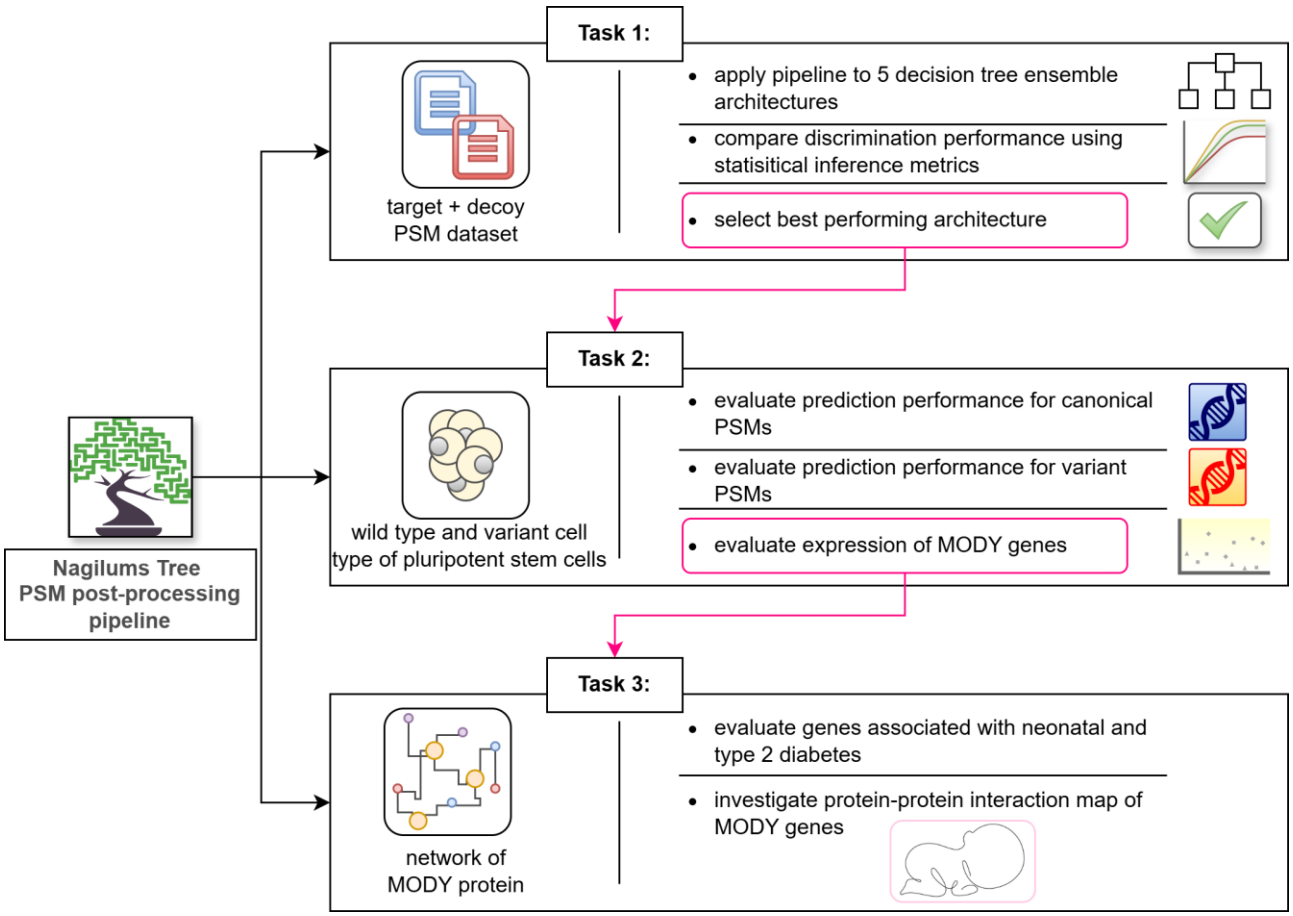


Figure 1.32 Project pipeline. The Nagilums Tree pipeline executes three consecutive tasks. Task 1 compares the discrimination performance of machine learning architectures using the target-decoy approach. Task 2 explores two search strategies to improve variant and canonical PSM classification of MODY-related proteins. Task 3 conducts a protein-protein interaction exploration for other diabetes subtypes as related to the classified MODY proteins.

Task 2 (**Figure 1.32**) is an exploratory analysis conducted using separate class types for canonical and variant spectra, which are used independently to create *decoy canonical* PSMs and *decoy variant* PSMs. Two search strategies are investigated using each decoy PSM type against the target class consisting of canonical and variant PSMs. The objective of this task is to investigate whether decoy variants can serve as a null model, potentially improving the statistical significance and resolution of confidently predicted variant PSMs. This analysis is performed on experimental data from pluripotent stem cells induced with insulin resistance targeting the HNF1A (MODY3) gene. The NT pipeline is adapted accordingly to support these modifications and conduct a comparative analysis of protein expression across both decoy search strategies.

Lastly, MODY gene expression is analysed using error metrics from the two search strategies to evaluate their effectiveness. Task 3 aims to evaluate the protein-protein interaction network of significant peptides to infer associations with MODY and other diabetic conditions. However, due to the extent of the current study, Task 3 is designated as future work and will unfortunately not be fully explored here although the step is developed. Instead, a brief evaluation of the MODY expression is included to support the discussion on the potential application of the concepts presented in this work.

The NT pipeline was written using Python, and all files are available on GitHub (link: Nagilums Tree pipeline). GitHub links are provided throughout the next chapters for direct reference to the appropriate Python files. The next chapter outlines the tools and methods implemented in the NT pipeline, including detailed descriptions of the PSM datasets, workflows, statistical tactics and performance evaluation techniques.

2: Materials and Methods

2.1 Flowchart of classification system

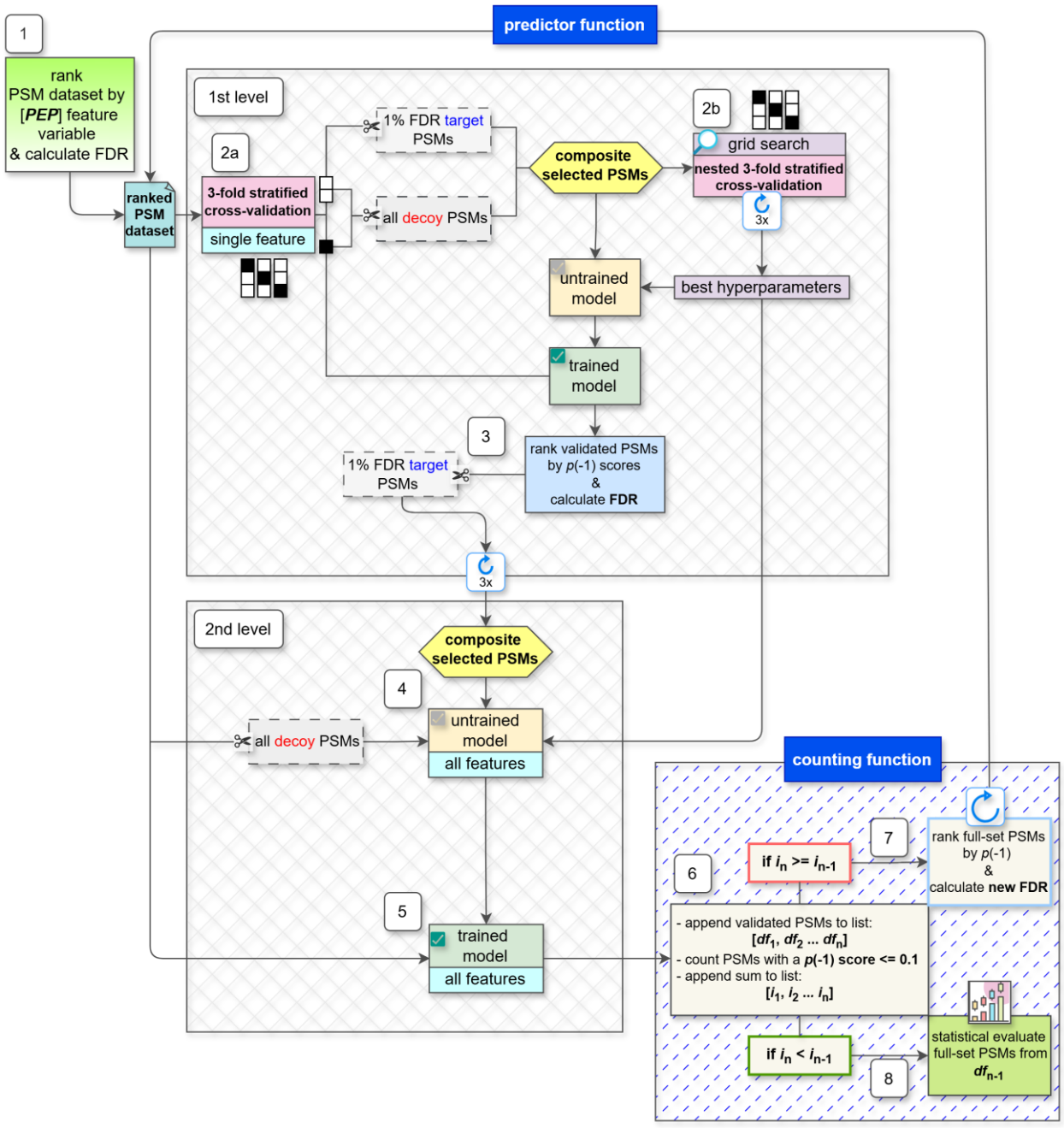


Figure 2.1 Schematic of the Nagilums Tree pipeline. The workflow is divided into 8 steps to classify a PSM dataset. The *predictor function* is separated into 2 levels. The 1st level hosts the nested *k*-fold cross-validation process. The white blocks are the training folds, and the black block is the testing fold (see Chapter 1.6.2.2, **Figure 1.16**). The full-set PSM dataset is classified at the 2nd level. The *counting function* loops with the *prediction function* until conditions are met.

The workflow of the NT pipelines' core learning structure is introduced in this chapter. In the workflow of Percolator, the testing folds of k-fold cross-validation (see Chapter 1.6.2.2) are merged using its normalisation process (Granholm *et al*., 2012). In this work an alternative approach is suggested, which is to employ cross-validation as a means to threshold and compile significant target spectra. Then, a learning algorithm is trained with this selection, and the full-set PSM dataset is classified. Therefore, no merging would be required, and the spectra can be evaluated as a complete set using the same test statistic. To accomplish this, the learning structure of the predictor function consists of 2 levels. The 1st level houses the cross-validation method and hyperparameter tuning. The 2nd level enacts the full-set classification. To understand the details of the workflow, a comprehensive view of the mechanics of NT is illustrated in **Figure 2.1**. The pipeline is divided into eight steps:

Step 1: The PSM dataset, consisting of target and decoy spectra matches, is first ranked by a single feature variable to estimate the FDR. In this work, the PEP scoring function is selected owing to its significance as a local error rate for peptide identification tasks. The ranked PSM dataset enters the predictor function at the 1st level.

Step 2a: The ranked PSM dataset undergoes 3-fold cross-validation using a single scoring function for the training and testing process. This is to avoid overestimation in the 2nd level classification process. Target spectra within a 1% FDR threshold are selected from the training folds and are combined with the full-set of decoy PSMs from the dataset.

Step 2b: These composite PSMs then undergo hyperparameter tuning through nested 3-fold cross-validation. An untrained learning algorithm is trained with the composite PSMs, and the best combination of hyperparameters is selected for classification optimisation. The trained algorithm then classifies the testing folds of cross-validation from Step 2a.

Step 3: The classified PSMs from each testing fold are ranked by a test statistic issued by the learning model, which in this work are the *prediction probability* scores and will be mentioned interchangeably with p(-1) scores hereon. The FDR is then estimated, and target spectra that meet the 1% FDR threshold are retained. The 1st level process repeats three times, once for each fold in the 3-fold cross-validation, during which a new learning model is trained for each iteration. After every iteration, the 1% FDR target spectra are compiled for the 2nd level.

Step 4: The composite set of 1% FDR target spectra and the full-set of decoy PSMs are used to train a new learning algorithm in the 2nd level. All available feature variables from the PSM dataset are included for this training step. The best combination of hyperparameters from Step 2b is passed to the untrained algorithm.

Step 5: The full-set PSM dataset is classified with all available feature variables using the trained learning algorithm and is then passed to the counting function.

Step 6: Two archives are created: one to store the classified PSM dataset and another to record the number of its target PSMs with a score $p(-1) \leq 0.1$. The predictor function repeats its process in this pipeline, and its results are appended to the respective archives. The counting function imposes a conditional check to track changes in the target PSM count archive.

Step 7: If the count of target PSMs increases compared to the previous iteration, the classified PSM dataset is re-ranked using the updated p(-1) test statistic, and a new FDR is estimated. This newly ranked dataset is passed back to the predictor function, and Steps 2 – 6 are repeated. With each iteration, the learning algorithm generates a new p(-1) test statistic, and the updated FDR is expected to further improve classification accuracy.

Step 8: If there is a decrease count of target PSMs from the previous iteration, the PSM dataset from the previous loop is retrieved from the archive and undergoes statistical evaluation.

This pipeline is designed to produce a PSM dataset ranked by the learning algorithm, enriched with a high frequency of significant target PSMs. The base learner created for this pipeline is applied in both Task 1 and Task 2 (see Chapter 1.9, **Figure 1.32**). The structure and logic of the predictor function and counting function will be explained using pseudo-code in the following chapter.

2.2 Python scripting language

The entirety of this project was written in the Python scripting language using the PyCharm IDE (version 3.10). Python is an attractive programming language for biological computing due to its ease-of-use, the versatility of its vast ecosystem of third-party library packages and user-friendly nature. It is an ideal tool for data analytics and visualisation. All PyCharm libraries were used at their most current version at the time of development and are listed for reference in **Table 2.1**.

Table 2.1 Python libraries used in the Nagilums Tree pipeline.

Library	Version	Modules and Methods	Application
Scikit-learn	1.5.0	*.fit* *.predit* *.predict_proba*	Base learner
		sklearn.model_selection: *StratifiedKFold*, *GridSearchCV*	Model configuration
		sklearn.preprocessing: *LabelEncoder*	Normalise label data
		sklearn: *metrics*	Statistics Application
		sklearn.ensemble: *ExtraTreesClassifier* *GradientBoostingClassifier* *HistGradientBoostingClassifier* *RandomForestClassifier*	Machine-learning library: Ensemble models
XGBoost	2.0.3	*XGBClassifier*	Machine-learning library: Boosting models
NumPy as np	1.26.4	*np.array*	Mathematical functions, arrays and matrices
Pandas as pd	2.2.2	*pd.read_csv* *pd.read_table* *pd.Dataframe* *pd.concat*	Data frame manipulation
Matplotlib	3.9.0	*matplotlib.pyplot*	Graphical visualization

2.3 The predictor function

2.3.1 Modular design concepts with Scikit-learn

Scikit-learn is an open-source Python library that offers a wide range of classical ML architectures. Since its release in 2011, Scikit-learn has quickly become a state-of-the-art tool across disciplines and continues to be actively developed with additional updates and improvements. Python programming is popular due to its highly functional environment that creates a user-friendly interface. These qualities are effectively leveraged by Scikit-learn to avoid framework code by using established Python libraries, mainly SciPy, Cython and NumPy for statistical modelling and data management (Pedregosa *et al*., 2011). This allows users to comprehensively create models that are easily integrated with other Python libraries to creatively extend its framework use for diverse applications. Current research, such as AlphaPept (Strauss *et al*., 2024) and PeptideMind (Handler and Haynes, 2021), have explored the use of Scikit-learn in biological computing within the context of proteomics. This chapter will describe the application of the Scikit-learn methods used in this project's pipeline. Descriptions of the base estimator, cross-validation, hyperparameter tuning and the decision tree ensemble architectures are included.

2.3.2 Base estimator and key methods

The attractiveness of Scikit-learn arises from it being object-oriented by interface and not inheritance, i.e. methods and variables are housed as abstract classes. External objects are then enabled to be flexibly integrated, allowing custom workflows to be created according to the user's specification (Pedregosa *et al*., 2011). This feature is essential for Task 1, in which learning metrics of the different ML architectures can be compared under the same paradigm. Ultimately, this assures the prediction scoring methods are analogous under the Scikit-learn framework, and statistical evaluations are directly comparable. The central object of a Scikit-learn interface is the base estimator, which hosts a set of default methods required to perform predictions as described in the block code in **Figure 2.2**. Here, the base estimator is introduced in the context of PSM dataset processing. The basic configuration includes creating individual variables for the PSM dataset file and a list of its feature scoring function description titles. The Scikit-learn base estimator has three primary methods: *fit*, *predict,* and *predict_proba* (**Table 2.1**).

```
1:  input_data = PSM input file                                  : setup
2:  feature_variables = [variable_01, variable_02...variable_n]
3:
4:  training_sample = input_data[training_portion]               : create training and testing samples
5:  testing_sample = input_data[testing_portion]
6:
7:  training_features = training_sample[feature_variables]       : create label and feature samples
8:  training_labels = np.array(training_labels['Label'])
9:
10: testing_features = testing_sample[feature_variables]
11: testing_labels = np.array(testing_labels['Label'])
12:
13: classifier = ml_model(parameters)                           : base estimator
14: classifier.fit(training_features, training_labels)*
15: classifier.predict(testing_features)*
16: classifier.predict_proba(testing_features)*
```

Figure 2.2 Block code of Scikit-learn's base estimator example. A description of the script workflow for a Scikit-learn classification task (left-align) with their basic usage descriptions (right-align). The data file is set up to process a PSM dataset. The base estimator configures the architecture and includes three key methods denoted by (*).

In this work, the term *training* refers to a ML model being '*fit*', which is the process of it learning patterns from a sample of the dataset using feature variables and class labels. The term *testing* refers to evaluating the trained model's ability to predict the class labels of an unseen sample from the dataset, where only the feature weights are provided. Separate datasets can be used for the training and testing process, or as visualised here in **Figure 2.2** the single dataset can be partitioned into individual samples. The feature variables and labels are extracted from the *training_sample* and *testing_sample*. The class type labels of the target and decoy spectra are isolated using the *NumPy* library (**Table 2.1**), which allows flexible processing of multidimensional environments (see Chapter 1.6). The base learner begins with selecting an ML architecture from the Scikit-learn or XGBoost library (**Table 2.1**) and stipulating configured hyperparameters (see Chapter 1.6.2.1). The learning model is *fit* with the *training_features* and *training_labels* (**Figure 2.2,** line 14). The trained model is tested with the *testing_features* (**Figure 2.2,** line 15) by using the *.predict* function.

Beyond label prediction, the learning model's confidence regarding the classification of the testing sample can be obtained using the *.predict_proba* (**Figure 2.2,** line 16) method from Scikit-learn, which returns two probability scores following binary classification assignments: $p(1)$ and $p(-1)$. These scores represent the predicted probabilities of a given data point belonging to a positive class (target) and a negative class (decoy), respectively. An example of

this is visualised in **Table 2.2**, which demonstrates the classification distribution of two PSMs. The probabilities for each data point (spectrum) sum up to 1.0 (100%), and are a distribution scaled from 0.0 – 1.0 representing a low to high degree of confidence, respectively.

Table 2.2 An example of a prediction probability score distribution for classified data points.

PSM_identification	p**(1)**	p**(-1)**
PSM_01	0.99	0.01
PSM_02	0.02	0.98

In this work, significance is inferred onto the classified spectra using solely the p(-1) scores. The reasoning for including decoy PSMs is to infer the null model of statistical hypothesis testing (see Chapter 1.5.2), which concerns error rate estimation. It is known that the decoy spectra are false matches and are expected to obtain a 100% p(-1) probability of being an incorrect match; however, the classification process is not intended to be biased (see Chapter 1.6.1). Instead, it is expected for decoy spectra to occur throughout the probability range, although at higher frequencies toward 100% p(-1) score and lower towards a 0% p(-1) score. Target spectra are then expected to be the opposite of these trends, and the frequency of decoy spectra at probability intervals reassures the likelihood of canonical and variant PSMs being an incorrect or correct match. Furthermore, it is insincere to report PSMs as correct non-random matches with absolute certainty at a p(1) of 100%, as this grading is subjective to the classification system and metrics employed. It is more suitable to report the likelihood of a PSM not being non-random. Leveraging the p(-1) scores as a test statistic will produce informed error rate estimations and model performance metrics regarding target and decoy spectra classification.

2.4 Dataset description and ranking

The spectra are organised in rows defined by columns of the peptide sequence, label (annotated with {+1} for a target PSM and a {-1} for a decoy PSM), and multiple scoring functions. An example of the layout is demonstrated in **Table 2.3**. The PSM dataset is received as either a .csv file or a .txt file, which is processed as a dataframe object using *pd.read_csv* or *pd.read_table* (**Table 2.1**), respectively. The dataset is ranked by using a Python method called *.sort_values()*. To rank spectra by the *pep* scoring function would appear as: *df.sort_values(by=['pep'])*.

Table 2.3 Layout of a PSM data file.

index	PSM identification	Label	Variable_01	Variable_02	Variable_*n*
0	Peptide_sequence_01	+1	0.55	0.78	…
1	Peptide_sequence_02	-1	0.23	0.44	…

2.5 Development of the Nagilums Tree workflow

2.5.1 Stratified k-fold cross-validation

The base learner of the predictor function is designed as a self-training algorithm (see Chapter 1.6), by using k-fold cross-validation (see Chapter 1.6.2.2). The number of k-folds required is dependent on the dataset size, which is relative to its complexity (Yadav and Shukla, 2016; Gorriz *et al.*, 2024). Considering PSM datasets can include several thousands to millions of spectra, they can be defined as a large dataset (see Chapter 1.6.2.2). Therefore, in this work *stratified* 3-*fold cross-validation* is employed (**Figure 2.3**). This is similarly used in the SVM trainer system of Percolator (Granholm *et al.*, 2012). A lesser number of folds is also necessary for computational resource efficiency. To ensure a proportionate representation of the target and decoy class types in each fold, *stratification* is included.

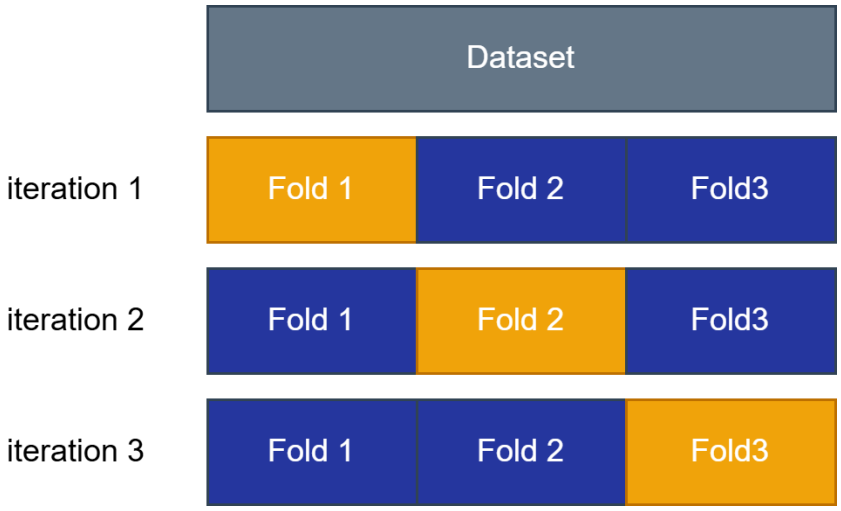


Figure 2.3 Method of 3-fold cross-validation. A dataset is divided into three equal subsets known as folds. The two blue folds are used for training a learning model. The single orange fold is classified. This process iterates three times to classify the complete dataset.

The block code of the stratified cross-validation process created for the *predictor function* is provided in **Figure 2.4[a]**, which describes the process of the 1st level of **Figure 2.1**. The *StratifiedKFold* method is derived from the Scikit-learn library (**Table 2.1**) and is the configuration setup for the number of splits (*folds*) for cross-validation. The *pep* scoring function from the PSM dataset is selected as the single feature variable (**Figure 2.1**, Step 1) for cross-validation. Individual object variables are created for the PSM dataset's scoring function, refined by the *pep* variable, and the class labels (**Table 2.3**).

To enact cross-validation, a *for loop* is created and in each iteration the dataset is separated into the training $\{k - 1\}$ and testing $\{k - (k - 1)\}$ folds, named the *training_index_01* and *validation_index_01*, respectively. The *enumerate* function (**Figure 2.4[a]**, line 8) facilitates this separation through the incorporated *StratifiedKFold* method, which monitors the three iterations, and portions the folds accordingly using the feature and label variables created earlier. The spectra are recorded by their index values (**Table 2.3**). Within the loop, the indices are used to extract the appropriate spectra directly from the dataset (**Figure 2.4[a],** lines 10 and 11) to create the *training_fold_01* and *validation_fold_01* variables. Note *validation_fold* refers to the testing process; here, it is used interchangeably to track the cross-validation levels.

In lines 13 – 16 of **Figure 2.4[a]**, the *training_folds_01* and *validation_fold_01* are prepared for a base estimator by dividing them into the appropriate feature and label object variables. Within the first layer of cross-validation (**Figure 2.1**, Step 2a), the target PSMs of the *training_folds_01* are extracted by their class label {+1} and an FDR of 0.01 (1%) using the process in lines 18 – 20 in **Figure 2.4[a]**. The full-set of decoy PSMs is selected by their class label {-1} using the process as line 21 in **Figure 2.4[a]**. These two subset dataframes are combined using *pd.concat* to then be portioned again into feature and label variables for hyperparameter tuning, which the process is explained in the next chapter. Cross-validation continues with *training_fold_01*, where, after each iteration, the classified spectra of *validation_fold_01* are ranked by the p(-1) scores to then estimate the FDR, repeating the steps of lines 18 - 23 (**Figure 2.1**, Step 3). The final composite is lastly used to train a new learning algorithm for the main classification, in which the full-set PSM dataset is classified (**Figure 2.1**, 2nd level Steps 4 and 5). The details of this process are explained later in Chapter 2.7, which includes the initiating process of NT.

a)

```
1:  StratifiedKFold(n_splits=3, random_state=1, shuffle=True)                  : setup
2:  input_file = PSM input file
3:  single feature = 'pep'
4:  dataset_feature = input_date[single_feature]                               : separate input file into feature
5:  dataset_labels = input_data['Label']                                        and labels
6:
7:  for fold, (training_index_01, validation_index_01) in \                   : stratified cross-validation loop
8:      enumerate(stratifiedKFold, split(dataset_features, dataset_labels):
9:
10:     training_folds_01 = input_data[training_index_01]                     : input file separate into training
11:     validation_fold_01 = input_data[validation_index_01]                    and validation folds
12:
13:     training_features_01 = training_folds_01[single_feature]
14:     training_labels_01 = np.array(training_fold_01['Label'])              : base estimator setup
15:     testing_features_01 = validation_fold_01[single_feature]
16:     testing_labels_01 = np.array(validation_fold_01['Label'])
17:
18:     psms_within_fdr_threshold = training_folds_01[training_folds_01['FDR'] <= 0.01]
19:     target_psms_within_fdr_threshold =                                    : composite 1% FDR target PSMs
20:                 psms_within_fdr_threshold[psms_within_fdr_threshold['Label'] == 1]   from  training folds and
21:     decoy_psms = input_file[input_file['Label'] == -1]                      decoy PSMs from the dataset
22:     composite_target_decoy_spectra =
23:                 pd.concat([target_psms_within_fdr_threshold, decoy_psms])
24:
25:     training_features_02 = composite_target_decoy_spectra[single_feature]
26:     training_labels_02 = composite_target_decoy_spectra['Label']
27:
28:     for fold, (training_index_02, validation_index_02) in \               : nested stratified cross-validation
29:         enumerate(stratifiedKFold, split(training_features_02, training_labels_02):   loop
30:
31:         training_folds_02 = training_fold_02[training_index_02]           : training folds separated into new
32:         validation_fold_02 = validation_fold_02[validation_index_02]        training and validation
33:         training_features_03 = training_folds_02[single_feature]          : hyperparameter tuning with
34:         training_labels_03 = np.array(training_folds_02['Label'])           training  fold
35:
36:         base estimator continues ...                                      : GridSearch base estimator
```

b)

```
37: grid_parameters = {'learning_rate': =[1, 5, 10], 'n_estimators'=[100, 200], \   : dictionary of parameters of interest
38:                    'max_depth'=[3, 5]}
39: hyperparameter_tuning = GridSearchCV(estimator=ml_model(warm_start=False), \    : GridSearch setup
40:                                      param_grid=grid_parameters)
41: hyperparameter_tuning.fit(training_features_03, training_labels_03)             : base estimator
42: best_parameters = hyperparameter_tuning.best_params                             : select best parameters
```

c)

```
- [learning_rate=1, n_estimators=200, max_depth=5]
- [learning_rate=5, n_estimators=200, max_depth=3]
- [learning_rate=1, n_estimators=100, max_depth=3]

* best_parameters_overall = ['learning_rate'=1, 'n_estimators'=200, 'max_depth'=3]
```

Figure 2.4 Block code of the predictor function self-trainer structure. The layout of the scripting environment for the *k*-fold cross-validation method is described (left-align) with their basic usage descriptions (right-align). **(a) 3-fold cross-validation**. The PSM dataset is prepared for classification (lines 1 – 5). Cross-validation is set up using the *StratifiedKFold* method: *n_splits* is set to 3 to correspond with 3-fold cross-validation, *random_state* is set to 1 and *shuffle* is set to True to allow for data points to be shuffled and randomly selected into *folds*. Two levels of cross-validation are created to process a PSM dataset with hyperparameter tuning (lines 7 and 28). The process to threshold target PSMs and collect decoy PMS for training a

learning algorithm is outlined (lines 18 - 23). **(b) Hyperparameter tuning**. The *GridSearchCV* method is used to execute hyperparameter tuning and with all possible combinations of parameter variables listed in *grid_parameters*. The ML architecture is executed in the *estimator* argument and *param_grid* calls the parameter variables. *warm_start=False* enables a new model to be fit with each repeat. The best combination of parameters is returned with *best_params*. **(c) Selecting the best combination of parameters.** Multiple combinations of hyperparameters are returned after each iteration of cross-validation. The learning process proceeds with the most common occurring variables (variables in bold) for each parameter.

2.5.1.1 Hyperparameter tuning

Hyperparameter tuning configures certain characteristics attributed to decision tree structures (see Chapter 1.8.2, **Figure 1.28**), such as the number of *nodes*, *branches*, *leaves* and *trees*, to control robustness of the classification assignment. In Scikit-learn, parameters are set to default generalised variables, e.g. Histogram-based Gradient Boosting has a parameter *max_leaf_node* set at 31, meaning that each base learner would be limited to create this number of leaves. Note that as each of the ensemble architectures has a decision tree estimator, they would share similar parameters; however, the default variable can be different, e.g. this same parameter is set to 'None' (meaning limitless and not zero) for Gradient Boosting models. These default variables are set as an average variable determined by randomised datasets.

In this work, hyperparameter tuning is necessary but not a priority; therefore, only a few parameters with limited variables will be selected, as listed in **Table 2.4**. The variables for each parameter selected are a mixture of the defaults and alternatives. The descriptions of the parameters are directly from the Scikit-learn webpage, which is available online through the link: Scikit-learn or Nagilums Tree pipeline. The PSM datasets used in this study are specifically curated to be of high quality in their own degree, and to be mindful of this, a limited number of parameters are selected to reduce the chance of overfitting.

Table 2.4 Parameters of decision tree ensemble architectures**.**

Architecture	**Parameters**	**Descriptions****
Random Forest	'max_depth': [3, 10]	A
	'max_leaf_nodes': [10, 31]	B
	'min_sample_leaf': [2, 10]	C
	'max_features': [2, 5, 8]	D

Gradient Boosting	'learning_rate': [***0.1**, 1.0]	E
	'n_estimators': [***100**, 200]	F
	'min_sample_leaf': [20, 100]	C
	'max_depth': [***3**, 5]	A
Histogram-based Gradient Boosting	'learning_rate': [***0.1**, 1.0]	E
	'max_leaf_nodes': [***31**, 50]	B
	'min_sample_leaf': [***20**, 100, 150]	C
	'max_iter': [50, ***100**, 200]	G
Extra Trees	'n_estimators': [***100**, 200]	F
	'min_sample_leaf': [***2**, 20, 100]	C
	'max_depth': [***0**, 3, 5]	A
XGBoost	'gamma': [1, 5]	G
	'min_child_weight': [***1**, 5]	H
	'max_depth': [3, ***6**]	A
	'subsample': [0.5, ***1.0**]	I
	'colsample-bytree': [0.5, ***1.0**]	J

*** Default variables**

****Descriptions**

A. **Controls the growth of the tree, branching nodes. More depth can increase overestimation.**
B. **Maximum number of nodes per tree. Limits complexity and reduces overfitting.**
C. **Minimum number of data points per leaf node.**
D. **Number of feature variables considered for the best split. Controls randomness.**
E. **Controls the contribution of each tree. Lower values increase robustness, require more trees.**
F. **Number of trees in the ensemble. An increase can improve performance or overfit.**
G. **Reduce loss to increase branches/splits. Higher values increase conservativeness.**
H. **Minimum sum of instances to create a node. Higher values increase conservativeness.**
I. **Probability of each instance being selected for training; 0.5 is best.**
J. **Ratio of feature variables used to construct each tree.**

The *nested 3-fold cross-validation* method is performed using the same *StratifiedKFold* setup (**Figure 2.4**, lines 1 and 28). The composite spectra that are divided into individual objects of the single feature variable (*training_features_02*) and labels (*training_labels_02*), are stratified using the same process described earlier (**Figure 2.4[a]**, lines 25 - 29). A second level of object variables is created using the indexing method for the *training_fold_02* and *validation_fold_02* (**Figure 2.4[a]**, lines 31 and 32). Again, individual variables, *training_features_03* and *training_labels_03*, are formed (**Figure 2.4[a]**, lines 33 and 34), which will undergo hyperparameter tuning.

Hyperparameter tuning is implemented in Steb 2b of **Figure 2.1** and is continued in **Figure 2.4[b]**. An object variable is created for the dictionary list of hyperparameters and variables concerning the decision tree-based ensemble architecture (**Figure 2.4[b]**, line 37). Then the *GridSearchCV* model is executed (**Figure 2.4[b]**, line 41) by following the base estimator (**Figure 2.2**) method.

The best combination of hyperparameter variables is isolated in **Figure 2.4[b]**. There are three combinations produced for each iteration of cross-validation with *training_fold_02*. Altogether, nine combinations are produced for each *training fold_01*. Although each hyperparameter set is specific to its corresponding training subset, the NT pipeline adopts a generalised approach by selecting the most frequently occurring variables across all combinations. Therefore, one out of the three combinations generated for *testing_fold_02* will be selected to complete the training process of cross-validation (**Figure 2.4[c]**, **Figure 2.1**, Step 2a), and one out of the nine combinations derived for *testing_fold_01* is selected for the main classification (**Figure 2.1**, Steps 4 and 5).

2.6 FDR implementation method

To estimate the FDR for a classified testing dataset, this work adopts the method originally introduced in Percolator (see Chapter 1.5.3), which is well-suited to the TDA. Although this FDR equation was intentionally created for p-score evaluation, **Figure 2.5** demonstrates how its application can be suited for *prediction probability* p(-1) scores.

To ensure the accuracy of the FDR, the p(-1) scores are rounded to six decimal places. In the argument of the importance of significant digits, Cole (2015) stipulates that test statistics require no more than two; however, percentage estimates lower than 1% can use more as necessary. The *prediction probability* p(-1) estimates are a probability test statistic, and during the course of developing NT, it was found that fewer significant digits produced biased performance due to the strictness of the 1% FDR threshold. Therefore, to retain variability, six decimal places were the lowest number of digits observed to successfully complete classification for each ML architecture.

The FDR script for NT is set up as a function called *FDR_calculation* and is available in the *Calculations* Python files of Tasks 1 and 2. The GitHub links are provided in each task's implementation chapters. The function can be imported into their PSM classification Python files to be readily implemented during the pipeline (**Figure 2.1**, Steps 3 and 7).

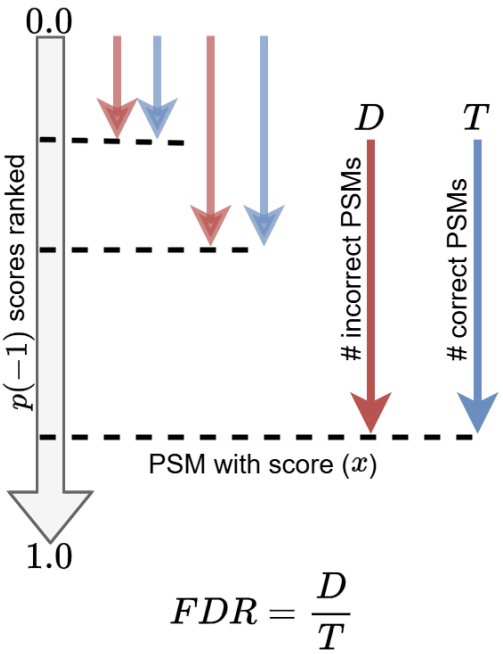


Figure 2.5 FDR estimation method. Classified spectra are sorted by the *prediction probability* p(-1) scores in ascending order. Going down the list, the number of correct PSMs (T for target PSMs) and incorrect PSMs (D for decoy PSMs) up to each spectrum threshold (x) are counted. These accumulated counts are used to calculate the FDR using the prescribed formula.

2.7 The counting function

The main objective of NT is to classify a PSM dataset while maximising the number of significant peptide sequences. The block code of the *counting function* is provided in **Figure 2.6** to explain the process. The NT workflow begins and the predictor and counting functions are executed in this script. Outside the function the main setup for the NT pipeline is configured. Archives (**Figure 2.1**, Step 6) for the significant target spectra count and dataset are created. The PSM dataset is initially ranked by the single feature variable '*pep'*. Note that line 5 is scripted differently in the Python files and is the correct format of this function.

```
1:  def counting_function(PSM_dataset, target_spectra_count_archive, dataset_archive):
2:      classified_PSM_dataset = predictor_function(PSM_dataset)                    : classify PSM dataset
3:      dataset_archive.append(classified_PSM_dataset)                              : store classified dataset
4:      confidently_predicted_target_PSMs = \                                       : collect target PSM count
5:                          classified_PSM_dataset.query('Probability p(-1) <= 0.1 and label == 1')
6:      number_of_target_PSMs = len(confidently_predicted_PSMs)
7:      target_spectra_count_archive.append(number_of_target_PSMs)                  : store target PSM count
8:      current_target_count_archive_entry =target_spectra_count_archive[-1]
9:      previous_target_count_archive_entry = target_spectra_count_archive[-2]
10:
11:     if current_target_count_archive_entry >= previous_target_count_archive_entry:   : conditional statement
12:         classified_dataset_index = classified_PSM_dataset.index.to_list()          : revert ranked classified spectra into
13:         ranked_PSM_dataset = PSM_dataset.iloc[classified_dataset_index]              PSM dataset
14:         ranked_PSM_dataset['FDR'] = classified_PSM_dataset.values                  : add FDR from ranked spectra
15:         counting_function(ranked_PSM_dataset, target_spectra_count_archive, dataset_archive)   : classify ranked PSM dataset
16:     else:
17:         previous_dataset_archive_entry = dataset_archive[-2]                       : collect previous archived dataset
18:         ranked_dataset_prepared = \                                                : remove redundant columns
19:                     previous_dataset_archive_entry.drop(columns=[target counter, decoy counter,
20:                                                                  target, decoy, FDR])
21:         ranked_dataset_confusion_matrix = confusion_matrix(ranked_dataset_prepared)   : confusion matrix
22:         ranked_dataset_confusion_matrix.to_csv('ensemble architecture confusion matrix.csv')   : save locally as csv file
23:
24: target_spectra_count_archive = [0]                                              : setup
25: dataset_archive = []
26: PSM_input_file = PSM input file
27: pep_ranked_dataset = PSM.input_file.sort_values(by=['pep'])                     : rank and apply FDR
28: pep_ranked_dataset_with_FDR = fdr_calculation(ranked_PSM_input_file)
29: counting_function(pep_ranked_dataset_with_FDR , target_spectra_count_archive, dataset_archive)   : start model pipeline
```

Figure 2.6 Block code of the counting function. The workflow of the scripting environment for the counting function is described (left-align) with their basic usage descriptions (right-align). The two archives: *target_spectra_count_archive* (line 24) and *dataset_archive* (line 25), are created. The PSM dataset is initiated for processing by ranking by the *pep* variable using *.sort_values(by=['pep'])* and the FDR is calculated (lines 26 - 28). The NT pipeline begins by running the counting function (line 29). The predictor function is executed (line 2) with the setup parameters passed from the counting function. Significant spectra counts are isolated in *confidently_predicted_target_psms*. The conditional loop (line 11) is set up to assess the classified PSM dataset for the number of significant target spectra and run the counting function again as required (line 15). The final dataset with the most significant spectra is retrieved in line 17 and the confusion matrix is executed (line 21).

The ranked PSM dataset and the significant target spectra count with a $p(-1) \leq 0.1$ are archived (**Figure 2.1**, Step 6). An example of the first entry into these archives would appear as follows:

- Significant target spectra count = [0, 235 872]
- Datasets = [df_01]

The counting function enacts a conditional assessment to ascertain if the total count has increased or decreased from that of the previous classification. Archives for the significant target spectra count are created with an initial entry value of zero to initiate the first repeat of the predictor function. Upon observing an increase in the target count from 0, the FDR is estimated on the newly ranked PSM dataset, and the predictor function is repeated by running the counting function again with the newly ranked dataset. With each repeat, the spectra are ranked by the newly issued $p(-1)$ scores and the FDR is estimated. Then, the dataset is again assessed within the counting function. If there is an increase again, this loop process (**Figure 2.1**, Step 7) continues as such:

- Significant target spectra count = [0, 235 872, 324 586, 435 671…]
- Datasets = [df_1, df_2, df_3…]

The loop between the predictor and counting function ends when there is a drop in the next target spectra count. The ranked dataset producing the most significant target spectra from the previous entry (indicated by the * in the example below) is retrieved for statistical evaluation (**Figure 2.1**, Step 8):

- Significant target spectra count = [0, 235 872, 324 586, ***[435 671]**, 218 723]
- Datasets = [df_1, df_2, ***[df_3]**, df_4]

The dataset to be evaluated is prepared by removing redundant columns that would interfere with metric estimation for downstream analysis (**Figure 2.6**, lines 18 - 20). In this script, the *confusion_matrix* is executed for the final ranked dataset. This step is only necessary for implementation in Task 1, which concerns evaluating ML model performance and is removed for Task 2 as only the PSM ranking is required; however, it can easily be added if necessary for future works.

2.8 Implementation tasks of Nagilums Tree

2.8.1 Task 1: Comparison of decision tree ensemble architectures

The nature of this Task is to evaluate the prediction performance of five decision tree architectures processing a combined PSM dataset of target and decoy spectra. The architectures compared are: Random Forest, Extra Trees, Gradient Boosting, Histogram Gradient Boosting and XGBoost. The method of Task 1 follows as a final statistical inference onto the proteogenomic pipeline, which is detailed as a flow diagram in **Figure 2.7**. In this chapter, details of this process are explained with reference to this figure. All Python files for Task 1 are available on GitHub at: Task 1: Comparison of ML architectures.

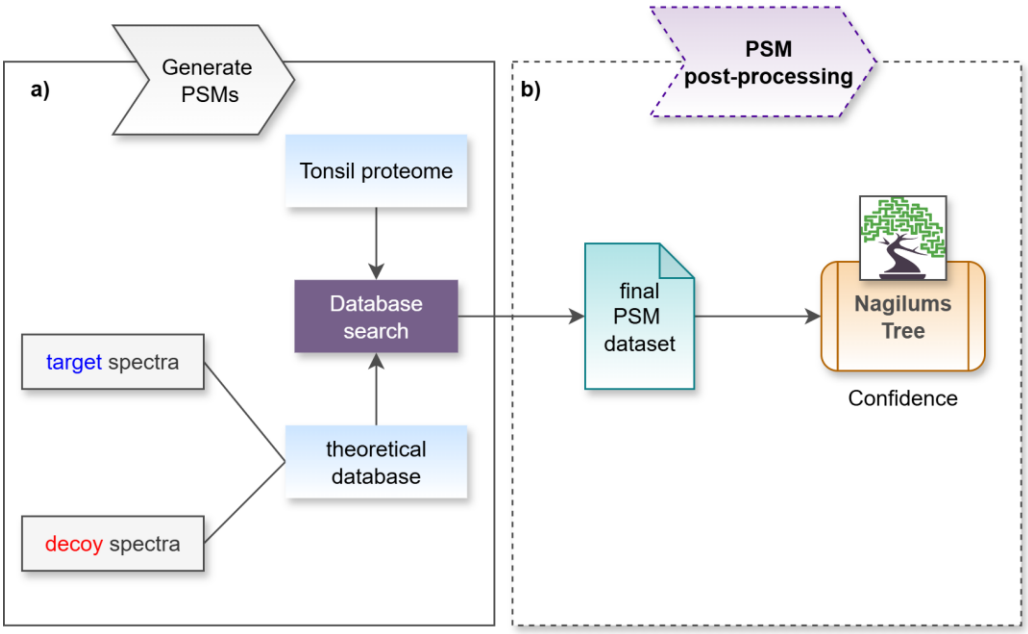


Figure 2.7 Proteogenomic pipeline of Task 1. (a) Generate PSMs: The tonsil proteome dataset is annotated with a theoretical dataset consisting of target and decoy spectra. PSMs are produced after a database search. **(b) PSM post-processing:** The final PSM dataset is processed using Nagilums Tree.

2.8.1.1 Creating and processing the PSM dataset

The process of creating the PSM dataset is illustrated in **Figure 2.7[a]**. Publicly available mass spectrometry proteomic data of healthy tonsil tissue by Wang *et al*. (2019) is collected as the experimental dataset. These observed spectra underwent sequence alignment using a theoretical dataset composed of target and decoy spectra. The target spectra, comprised of the canonical and variant proteome, are obtained from Ensembl (Harrison *et al*., 2024). Their respective decoy sequences are created using DecoyPYrat (Wright and Choudhary, 2016).

The observed spectra are matched to this theoretical dataset using X!Tandem (Craig and Beavis, 2004) and refined using PeptideShaker (Vaudel *et al*., 2015). The PSMs created from this database search are returned with the raw scoring functions (**Table 2.5**) of the search engine. Peptide and spectrum features, such as *charge*, *enzymatic, isotopes*, etc., and engine scores, such as the *pep* and *delta_pep*, are included. The method of **Figure 2.7[a]** was conducted by Jakub Vašíček and Dafni Skiadopoulou. Owing to this, the physical file is not included in the GitHub folder.

Table 2.5 List of the scoring features of the PSM dataset.

measured_mz	*mz_error*	*pep*	*delta_pep*	*enzymatic_C*
ion_fraction	*peptide_length*	*charge_2*	*charge_3*	*enzymatic*
charge_4	*isotope_0*	*isotope_1*	*isotope_2*	
isotope_3	*isotope_4*	*unspecific*	*enzymatic_N*	

The final PSM dataset produced from this is intended to undergo a final post-processing using the NT pipeline (**Figure 2.7[b]**). Each architecture individually processes the PSM dataset using the NT pipeline (**Figure 2.1**) and produces a *.csv* file of the re-ranked spectra. The Python files for each architecture are available within the GitHub folder link: Decision tree ensemble models.

2.8.1.2 Confusion matrix method

The final step of NT is statistical inference on the processed PSM dataset ranked by its *prediction probability p*(-1) scores. In Task 1, these ranked spectra are investigated using the confusion matrix; the method was introduced in Chapter 1.6.3, **Figure 1.19** and **Figure 1.20**. The computational process to create a confusion matrix is visualised in **Figure 2.8**, which processes three example spectra (two target PSM and one decoy PSM). The classified spectra are sorted in ascending order by their *prediction probability p*(-1) scores in **Figure 2.8[a]**. The target and decoy spectra are then evaluated by their rank position by tabulating the frequency of each class label. The confusion matrix can then be applied as depicted in **Figure 2.8[b]**. The true positive, false positive, true negative and false negative rates are lastly computed for each spectrum. Note that the FPs are the same as the decoy count.

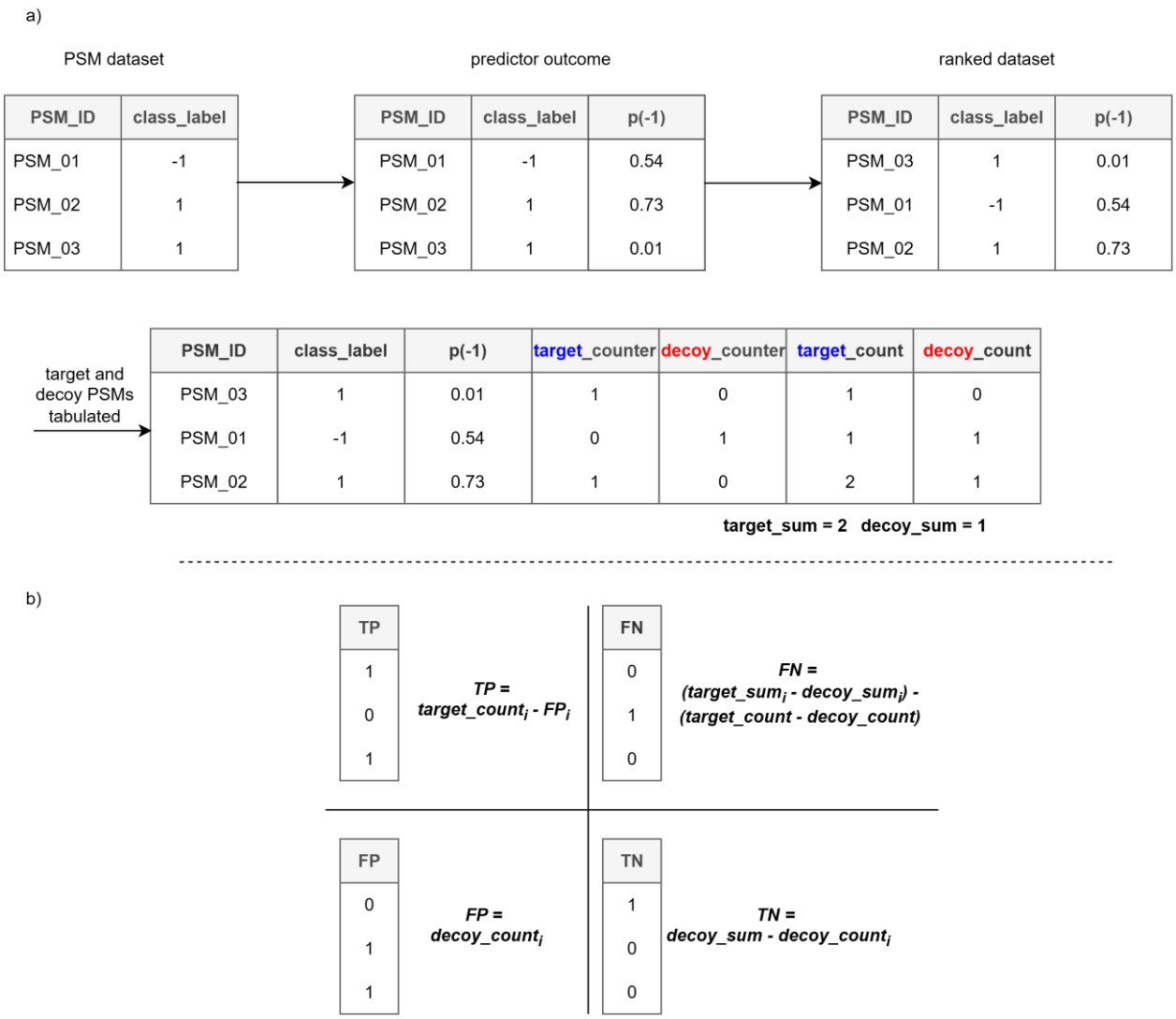


Figure 2.8 Creating the confusion matrix. An example PSM dataset consisting of two target PSMs and one decoy PSM is processed. **(a) The steps to setup a classified PSM dataset for the confusion matrix.** The *predictor outcome* step displays the *prediction probability* $p(-1)$ scores following classification, which is then used to sort the PSMs in the *ranked dataset* step. In the *target and decoy PSMs tabulated* step, the target_counter and decoy_counter record the class type of each spectrum. The target_count and decoy_count tally the frequency of each class type. The target_sum and decoy_sum are the number of target and decoy spectra within the dataset. **(b) Confusion matrix formulas.** The subscript (i) represents the frequency of target and decoy spectra to the rank of the PSM evaluated. The true positives (TP), false negatives (FN), false positives (FP) and true negatives (TN) are computed manually using these formulas for each spectrum.

After the confusion matrix is computed, further statistical metrics are estimated for each spectrum. An overview of the statistical measurements, as well as their respective formulas, is included in **Table 2.6**. The script is set up as a function called *confusion_matrix* and is available on GitHub at: Calculations. The function is imported into the Python file of each architecture's classifier and is implemented as the final step of the NT pipeline (**Figure 2.6**, line 21). The GitHub link: Plots_and_figures is the Python file containing all the necessary graphical functions to visualise the performance metrics. The GitHub link: Graphical_visualisation is a Python file set up to import the final *.csv* files of the classified PSM dataset of each decision tree ensemble model and enact the *plots_and_figures* script.

The plots created follow the model performance techniques of Chapter 1.6.3.2, with additional figures to display spectrum distribution using histograms and total count plots to evaluate the FDR. When computing the FDR, the *prediction probability* $p(-1)$ scores are adjusted to six decimal places (see Chapter 2.6). Although it is unnecessary to similarly adjust the calculated metrics due to the employed graphical displays, the *prediction probability* $p(-1)$ and FDR estimates are rounded down to three significant digits to improve visualisation resolution. All figures are included in the results chapter of this work.

Table 2.6 Learning model performance metrics using confusion matrix estimates.

Metric	Formulae	Description	Interpretation
Sensitivity	$\frac{TP}{TP + FN}$	True positive rate Proportion of true positives correctly identified Measurement of the ability or inability to detect a positive instance	A high sensitivity indicates a low rate of false negatives
1-Specificity	$\frac{FP}{FP + TN}$	False positive rate Proportion of actual negatives correctly identified Measurement of the ability to avoid false positives	A high specificity indicates a low rate of false positives
Precision	$\frac{TP}{TP + FP}$	Evaluates the accuracy of positive predictions Proportion of predicted positives that are true positives Measures trade-off between accurate positive predictions with a high occurrence of false positives	A high precision indicates that when a positive result is given, it is likely to be correct
Recall	$\frac{TP}{TP + FN}$	Same as sensitivity Measures the proportion of actual positives that are correctly identified	A high recall indicates that most positive cases are detected

2.8.2 Task 2: Decoy variant concept

This task is an exciting endeavour to investigate the possibility of using decoy variants as a null model that improves the prediction confidence of variant PSMs. The basis for this task is made possible by the setup of the PSM dataset, which is processed differently from the standard TDA, as in the methods of Task 1 from the previous chapter. Instead of two PSM class types, here four are created: *canonical*, *decoy sequence*, *variant* and *decoy variant*. Processing of these class types requires the NT classification process to be altered slightly so to manually label the PSMs appropriately for this task. Statistical inference will not include the confusion matrix at this point. Instead, PEP analysis will be conducted to infer confidence of protein expression as the PSM datasets concern MODY genes. The method of these tactics is described in the next subchapters with reference to the proteogenomic flow diagram in **Figure 2.9**. All Python files for Task 2 are on GitHub at: Task 2: Decoy variant concept. Corresponding links will be provided throughout this chapter to the appropriate file location of its methods.

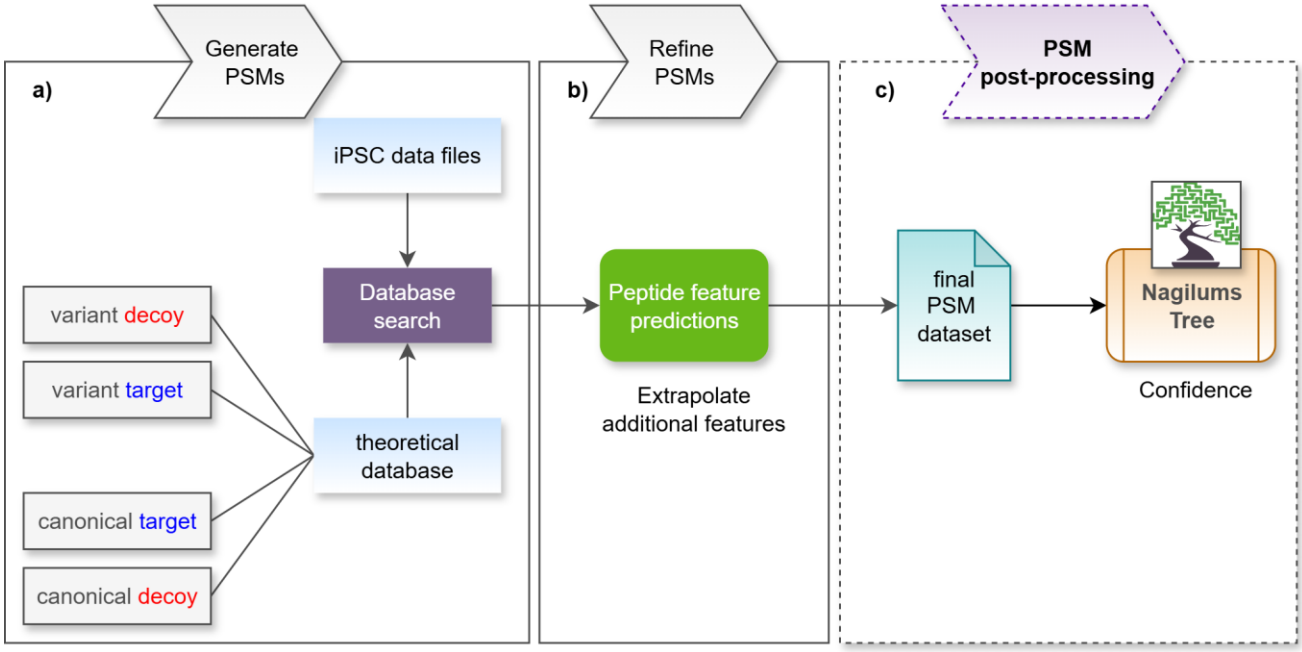


Figure 2.9 Proteogenomic pipeline of Task 2. (a) Generate PSMs: PSM dataset files are created from induced pluripotent stem cell (iPSC) experimental proteome sequences and a theoretical database setup with four PSM class types. **(b) Refine PSMs:** Additional scoring functions are extracted and are included in the annotated iPSC PSM files. **(c) PSM post-processing:** The refined PSM dataset is classified for statistical inference using the processing pipeline Nagilums Tree.

2.8.2.1 Theoretical dataset: creating decoy variants

Previously in Task 1, the target spectra consisted of the canonical and variant sequences. Here, they are separated into individual peptide species and following the same method in Task 1 (see Chapter 2.8.1.1), decoy spectra are created for both class types. The theoretical dataset is now comprised of four types of sequences. Decoy spectra were created for the canonical spectra using the standard method (see Chapter 1.5.1); however, instead of reversing and shuffling the sequences, decoy variants were designed using a novel approach. The specifics of the method are proprietary and unreportable at this time, although it can briefly be described that fake variants were crafted into the sequences of variant peptide genomic regions. It is not the purpose of this task to assess if a spectrum is a true variant; rather, its nature questions the likelihood of a point mutation (**Figure 1.3**) occurring on a known variant peptide (**Figure 2.10**).

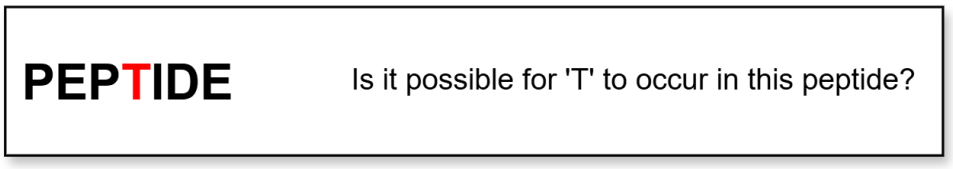


Figure 2.10 Explanation of decoy variant search strategy. The word 'PEPTIDE' is used as an example of a peptide sequence in which the individual letters represent amino acids. The red 'T' serves to indicate a variant in the sequence.

2.8.2.2 Experimental dataset: induced pluripotent stem cells

Pluripotent stem cells (PSCs) are undifferentiated animalia eukaryotic cells with the ability to specialise into functional molecular cell types diverging from the three primary germ layers: ectoderm, mesoderm and endoderm (Romito and Cobellis, 2015). Differentiation of PSCs is subjective according to experimental studies, and this quality has made them valuable to human health clinical research fields, including disease modelling, organ transplant, regenerative medicine and drug discovery (Shi *et al.*, 2017).

There are two types of human PSCs: those derived from embryonic cells and those that are induced from somatic cells (iPSCs) (Shi *et al.*, 2017). For this work, iPSCs engineered as pancreatic β-cells (see Chapter 1.2, **Figure 1.1**) were induced to develop insulin resistance. More specifically, a common variant was crafted into the HNF1A (MODY3) gene (see Chapter 1.3.2, **Table 1.1**) to suppress insulin production (Cujba *et al.*, 2022). The *mutant* cells and *wild type* cells (unmutated) were developed in hyperglycaemic conditions.

Then, proteomic expression was extracted at two key developmental stages: Stage 0 (early development) and Stage 4 (pancreatic progenitor). The pancreatic progenitor stage is not a fully developed pancreatic β-cell; however, most proteomic expression can be observed at this stage (van de Bunt *et al*., 2016). Three cell lines were generated for both development stages and cell types. In total, 12 iPSC proteome extractions were obtained that underwent LC/MS-MS to generate peptide sequences. This experimental research was conducted by Dr. M. Wierer from the Proteomic Research Infrastructure of the University of Copenhagen and procured by Dr Ksenia Kuznetsova.

2.8.2.3 Creating the iPSC PSM datasets

The observed iPSC spectra were matched to the theoretical dataset, and a PSM dataset comprised of the four spectra class types was generated. Annotation was conducted with the X!Tandem search engine and PeptideShaker (see Chapter 2.8.1.1). In this task, one other step is included in the proteogenomic pipeline, which is the PSM refinement stage (**Figure 2.9[b]**). This step is the experimental research of Dafni Skiadopoulou, whose work aimed to improve PSM classification resolution by extrapolating scoring features based on peptide feature predictors. These additional features, along with the 18 scoring features from X!Tandem, represent the 41 scoring features used in this task and are listed in **Table 2.7**. The features with *rt_* are peptide retention time predictions based on comparisons between *measured_rt* from the mass spectrometer using DeepLC (Bouwmeester *et al*., 2021). The remaining features are comparisons between *measured* spectra and peptide fragmentation pattern predictions using the model MS2PIP (Degroeve *et al*., 2013). The 12 iPSC PSM datasets are not included in the GitHub link at this point due to their use in external projects.

Table 2.7 List of the scoring features from the iPSC PSM datasets.

measured_rtpredicted_rt	*isotope_0*	*spectra_log*
engine_score	*isotope_1*	*spectra_cos_similarity*
measured_mz	*isotope_1*	*spectra_angular_similarity*
mz_error	*isotope_3*	*spectra_cross_entropy*
pep	*isotope_4*	*b_ion_coverage*
delta_pep	*unspecific*	*b_ion_matched_peaks*
ion_fraction	*enzymatic_N*	*b_ion_spectra_log*
peptide_length	*enzymatic_C*	*b_ion_spectra_cos_similarity*
charge_2	*enzymatic*	*b_ion_spectra_angular_similarity*
charge_3	*rt_Abs_error*	*b_ion_spectra_cross_entropy*
charge_4	*rt_Square_error*	*y_ion_spectra_cos_similarity*
y_ion_matched_peaks	*matched_peaks*	*y_ion_spectra_angular_similarity*
y_ion_spectra_log	*y_ion_coverage*	*y_ion_spectra_cross_entropy*

2.8.2.4 Processing the iPSC PSM datasets

The final iPSC PSM datasets are processed using NT, which is adapted to include the relevant scoring features and conduct the two search strategies, as visualised in **Figure 2.11**. The PSM class types are described as follows: *canonical* (*canonical* PSM), *target_seq_target_var* (*variant* PSM), *target_seq_decoy_var* (*decoy variant* PSM) and *decoy_seq* (*decoy sequence* PSMs, named in reference to decoy canonical PSMs). In both strategies, the *canonical* and *target_seq_target_var* are assigned a positive class label to represent the target PSMs, and both of the decoy PSM class types are labelled as the negative class. The method of the first strategy omits the *target_seq_decoy_var*, and the learning model discriminates between the target PSMs and the *decoy_seq* PSMs. The Python file is available on GitHub at: Decoy seq strategy. Then, in the second strategy the *decoy_seq* PSMs are removed, and the learning model discriminates between the target PSMs and the *target_seq_decoy_var* PSMs. The Python file is available on GitHub: Decoy variant strategy.

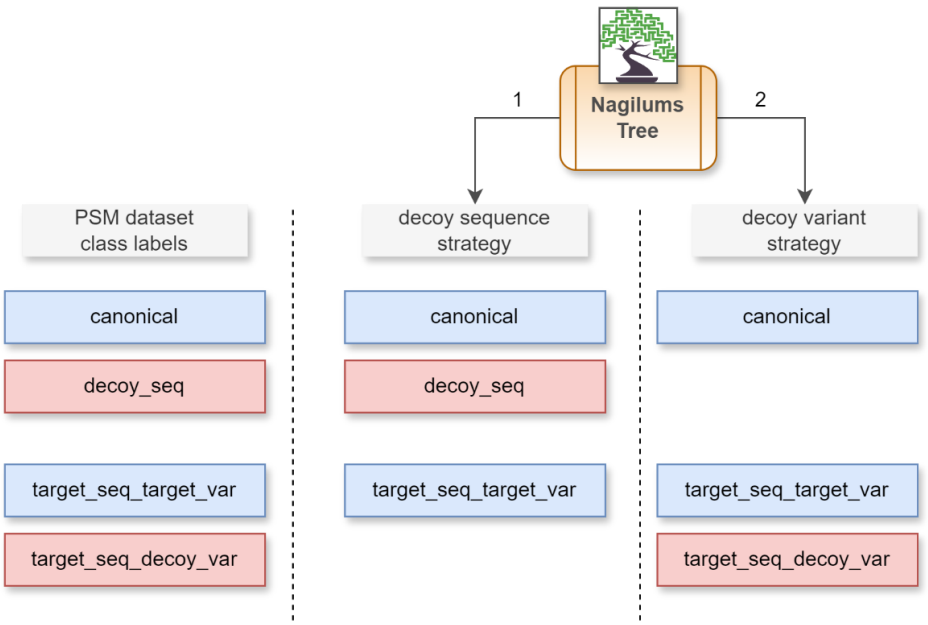


Figure 2.11 Description of the two search strategies of Task 2. The blue boxes indicate the PSM type labelled as a target PSM. The red boxes indicate the PSM type labelled as a decoy PSM. The decoy sequence strategy removes the *target_seq_decoy_var* class type. The decoy variant strategy removes the *decoy_seq*.

The performance of the *decoy sequences* PSMs, and *decoy variant* PSMs strategies is then evaluated. The Calculations Python file contains the FDR function and the confusion matrix, which is included as a formality, but will not be used for evaluation. Instead, histograms are created to compare the probability distribution of both canonical and variant PSMs of each search strategy. The Python file on GitHub: Plots and figures contains the necessary functions to obtain all figures.

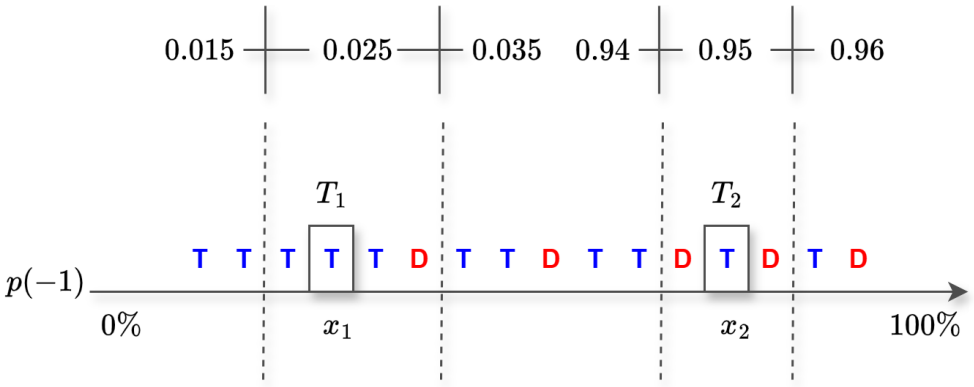
2.8.2.5 Posterior error probability estimation

The PEP metric is inferred (see Chapter 1.5.4) to evaluate the distribution of significant PSMs following classification. PEP values are typically estimated using ML techniques specific to the learning model's parameters and training data. However, in this study, the PEP values are not derived using Scikit-learn, and instead, a novel technique is explored as visualised in **Figure 2.12**. The formula recommended to estimate the PEP is adapted from the method originally proposed in Percolator (see Chapter 1.5.4.2, **Figure 1.11**). Here, the Percolator method is repurposed to accommodate the custom 'container' strategy.

Identification of mismatched variants is challenging due to their similarity to canonical counterparts and low abundance. This difficulty is further compounded by the stringent requirement that a PSM must meet a PEP threshold of 1% to be considered significant. Although

generally multiple PSMs would share a similar *prediction probability* p(-1) score, the 'container' PEP method is expected to capture spectra that would otherwise be marginally excluded. To further enhance their significance, the p(-1) scores serve as a secondary threshold measure, and its impact will be explored in the results chapter of this work.

The PEP script function in NT is named *pep_calculation* and is available in a Python file of the GitHub link: Calculations. The function is imported into the NT search strategies and is included in the final *.csv* output files. The GitHub link: Graphical visualisation provides the Python file setup to receive the processed iPSC PSM *.csv* datasets, which are 24 in total (12 for each decoy search strategy). The appropriate iPSC cell type and development stage are annotated for each file. The figures created are to display the PEP against the p(-1) distribution and compare the collective count of spectra for each cell type of both search strategies.



$$Range\ of\ T_n = [x_m - 0.01, x_m + 0.01]$$

$$PEP = \frac{D}{T}$$

$$T_1 : PEP = \frac{1}{3} = 0.66 \qquad T_2 : PEP = \frac{2}{1} = 2$$

Figure 2.12 PEP calculation method. Classified spectra are ranked by the *prediction probability* p(-1) test statistic in ascending order. Then for the p(-1) score (x_m) of each target spectrum (T_n), the number of target (T) and decoy (D) spectra with a score (x_m) ranging above (x_m + 0.01) and below (x_m − 0.01) are tallied and inserted into the PEP formulae prescribed from the works of Percolator (Käll *et al.*, 2008). An example of the method is observed for two target PSMs (T_1 and T_2) at different p(-1) positions, x_1 and x_2, respectively.

2.8.2.6 PEP score distribution of MODY genes

Lastly, the MODY protein-protein interaction network of Task 3 was omitted due to the extent of the current work. Nevertheless, it remains the final proposed step of the NT pipeline, with the Python scripts available on GitHub at: Task 3: MODY expression analysis for future works. As an alternative, a MODY protein expression plot is created by observing the PEP score distribution of the 14 MODY genes (see Chapter 1.3.2, **Table 1.1**).

The iPSC PSM datasets are structured to include inferred protein sequences and corresponding gene names for each spectrum. After classification using each decoy strategy, the datasets are merged with its original iPSC file to link estimated metrics to the PSM information. A Python script is written to extract the PEP scores of the 14 MODY genes from the classified iPSC PSM datasets. Peptide sequences generally map to multiple genes, and for this, gene names are separated into an individual PSM entry with the same metrics. To account for the case of multiple MODY genes being identified from each file, an average PEP score is estimated. This analysis is performed for both decoy strategies. The Python script for the *decoy sequence* and *decoy variant* strategies is available on GitHub at: (Decoy seq strategy) and (Decoy variant strategy), respectively. The .txt file containing the list of the 14 MODY genes is available on GitHub at: MODY gene files. This folder also includes an additional file with gene name identifiers curated from experimental research articles mentioning their possible association. This comprehensive list, compiled by Dr. Ksenia Kustenova, may provide interesting insight for future studies on protein-protein interactions concerning MODY and other forms of diabetes.

This setup is exploratory and not typical for protein expression profiling. The analysis aims to provide a general overview of how the PEP expression of the *decoy variant* PSM strategy differs from the *decoy sequence* strategy. Protein identification from peptide-spectrum matches is a complex and extensive topic in itself, which falls outside the scope of this work. A protein is made up of multiple peptide sequences, and the MODY genes evaluated here are identified by a single associated PSM. For this reason, protein expression is not directly inferred, and analysis will focus on PSMs. The PEP scores are not an indicator of protein expression level or abundance; rather, they indicate how the prediction resolution of the MODY genes is influenced by the decoy search strategies, which produced comparable PEP scores.

3: Results

3.1 Task 1: Comparison of decision tree ensemble architectures

In Task 1, five decision tree ensemble architectures: Histogram-based Gradient Boosting, Gradient Boosting, Random Forest, Extra Trees and XGBoost, individually processed a PSM dataset composed of proteomic data derived from tonsil tissues searched against a theoretical peptide sequence database. These target spectra were annotated with their corresponding decoy sequences. Each ensemble model performed binary classification using the target-decoy approach within the NT pipeline, a custom-built classifier system written in Python using the Scikit-learn library. The learning algorithms were trained to discriminate between target spectra with a 1% FDR and decoy spectra to improve true match prediction accuracy. Here, each architecture's classification performance is evaluated. The assessments will mainly concern the architecture's ability to discriminate between the target and decoy PSMs using histograms. Firstly, an overall performance review using AUROC and PR curves created from confusion matrix estimations is conducted. To conclude, the TDA is explored, as well as the proficiency of the FDR application concept. All graphical plots and supplementary items are available on GitHub at: Thesis and supplementary figures.

3.2 Model performance evaluation

The basis of the AUROC and PR curves was introduced in Chapter 1.6.3.2. The ROC graph is an (x, y) plot of the TPR (*sensitivity*) on the y-axis, against the FPR (*1 - specificity*) on the x-axis. Together, these metrics are used to assess a learning model's ability to discriminate dichotomous datasets, which in this case are the spectra with target and decoy class labels. Unlike in disease diagnostics, which requires a threshold to determine binary class types (Bainbridge, 2020), here the class labels are assigned prior to classification and are used in downstream analysis.

The AUROC curve created for Task 1 is viewed in **Figure 3.1**. The *sensitivity* metric ranges from 0.0 to 1.0 and reflects the model's ability to predict the positive class, or target spectra. Increasing *sensitivity* indicates a greater likelihood of a TP or a non-random target. Its companion metric, *1 – specificity*, similarly ranges from 0.0 to 1.0 and reflects the likelihood of spectra belonging to the negative class type, or decoy. A decreasing *1 – specificity* reflects a low chance of spectra being a FP or incorrect match. Therefore, a target spectrum with a high *sensitivity* and low *1 – specificity* is likely to be a correct non-random true match (see Chapter 1.5.1, **Figure 1.9**), and an increasing *1 – specificity* with high *sensitivity* indicates it is predicted

as random. Furthermore, decoy spectra are expected to have low *sensitivity* and high *1 – specificity*, and target spectra with similar metrics are likely to be non-random incorrect matches.

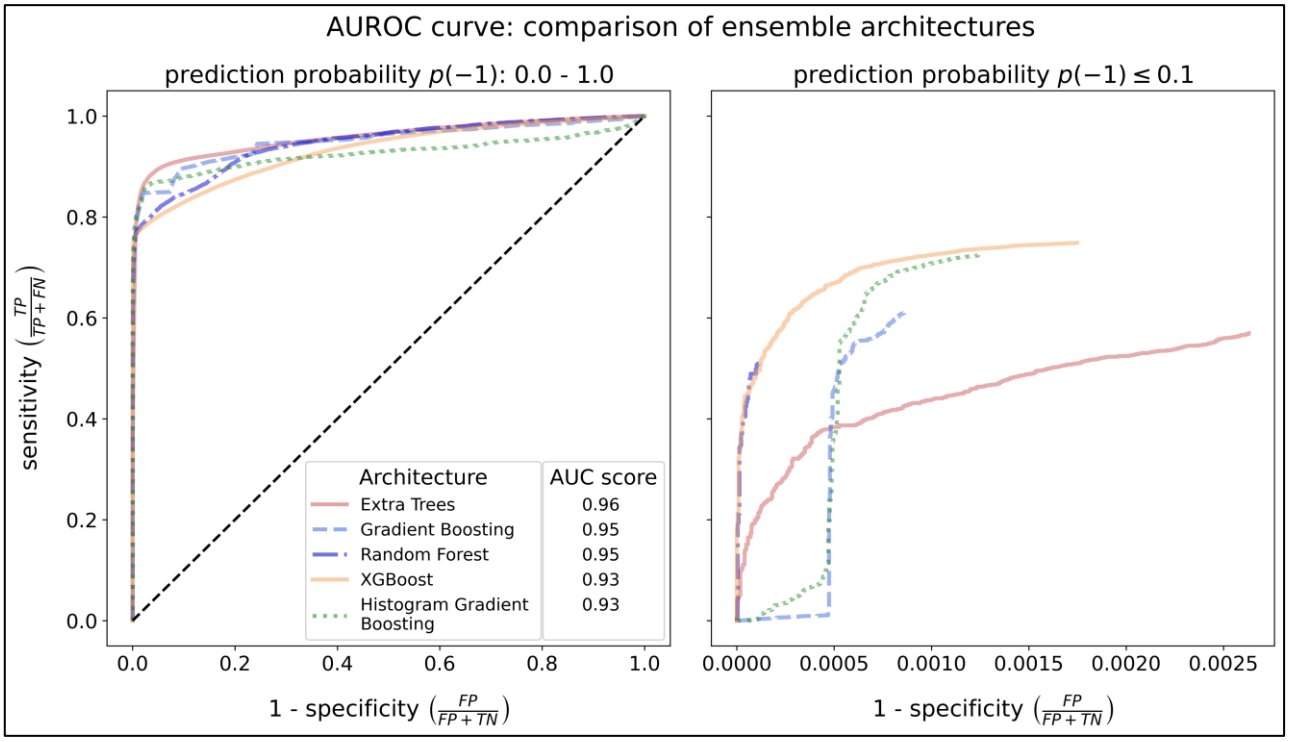


Figure 3.1 AUROC graph. The classification performance curves of five decision tree ensemble architectures are compared. The rate of *sensitivity* and *1 – specificity* is measured for each curve. **Left plot**: The full *prediction probability* p(-1) distribution of classified spectra. The black dotted line indicates a random learning model. **Right plot**: The performance of spectra with a p(-1) less than 0.1 for each architecture is compared. The legend in the lower right corner of the left plot colour codes the curves for each architecture (for both plots) and provides the AUC scores.

The decision tree architectures individually processed the PSM dataset, producing spectra with unique p(-1) scores and distinct ranking positions. The combined AUROC prediction performance of the architectures is viewed under two conditions in **Figure 3.1**: the first (left) displays the complete p(-1) range of 0.0 – 1.0, while the second (right) focuses on spectra with a p(-1) less than 0.1. In this work, a threshold of p(-1) $\leq$ 0.1 is applied to identify spectra as highly significant, indicating a high chance of being true matches, as outlined in Chapter 2.3.2. Together, these plots provide a comprehensive evaluation of prediction accuracy both at full inspection as well as a close-up view of the performance of the significant spectra.

The plots are accompanied by an AUC score, which is a summary of the learning algorithm's overall ability to discriminate between the class types. The AUC scores are a range of 0.0 – 1.0, where a value of 1.0 represents perfect classification. **Table 3.1** provides a guide for interpreting different AUC intervals. An AUC score of 0.5 indicates ambiguous classification, i.e. the learning model estimates there is a 50% chance of a spectrum being either a positive (target) or negative (decoy) class. This is depicted by the black dotted line in the left plot of **Figure 3.1**. The conditions of underfitting and overfitting can further be inferred by the shape of the curve and its distance to this random line, as discussed in Chapter 1.6.1. A curve near the random line would indicate underfitting, while an irregular curve may reflect overfitting (**Figure 1.13**).

Table 3.1 AUC value interval interpretation.

AUC value	**Interpretation suggestion**
$\mathbf{0.9} \leq \boldsymbol{AUC}$	Excellent
$\mathbf{0.8} \leq \boldsymbol{AUC} < \mathbf{0.9}$	Acceptable
$\mathbf{0.7} \leq \boldsymbol{AUC} < \mathbf{0.8}$	Fair
$\mathbf{0.6} \leq \boldsymbol{AUC} < \mathbf{0.7}$	Poor
$\mathbf{0.5} \leq \boldsymbol{AUC} < \mathbf{0.6}$	Ambiguous/fail

The first observation from **Figure 3.1(left)** is that all architecture curves appear in the upper-left corner of the plot far from the random line, and achieved AUC scores above 0.9, indicating excellent classification performance. However, on closer inspection of **Figure 3.1(right)**, the confidently predicted PSMs do not exhibit sufficiently high *sensitivity* scores. Therefore, significant spectra may not be amongst the spectra with high *sensitivity* scores of **Figure 3.1(left)**. XGBoost and Histogram Gradient Boosting had the highest *sensitivity* of approximately 0.7 yet recorded the lowest AUC of 0.93. They are followed by Gradient Boosting and Extra Trees with AUC scores of 0.95 and 0.96 respectively, and *sensitivity* scores of approximately 0.6. Lastly, Random Forest performed the weakest with a *sensitivity* of 0.5, although it still reached an AUC of 0.95. Despite these differences, all architectures exhibited considerably low 1 – *specificity* values well below 0.1. This suggests that while the target spectra of this range may be poor quality non-random true matches, they still maintain relatively reliable low error rates.

The PR plot in **Figure 3.2** is presented under the same two settings as the ROC curve in **Figure 3.1**. In this graph, the *precision* metric is plotted on the y-axis against the *recall* on the x-axis. The *recall* metric is equivalent to the *sensitivity* measurement of the ROC plot. The *precision* is a measurement scaled at 0.0 – 1.0, where an increasing score indicates spectra are highly likely to be true positive matches with a low chance of being false positives. Target spectra with high *precision* and *recall* scores are interpreted as having a high probability of being a TP non-random with a low likelihood of being a FP or incorrect random match.

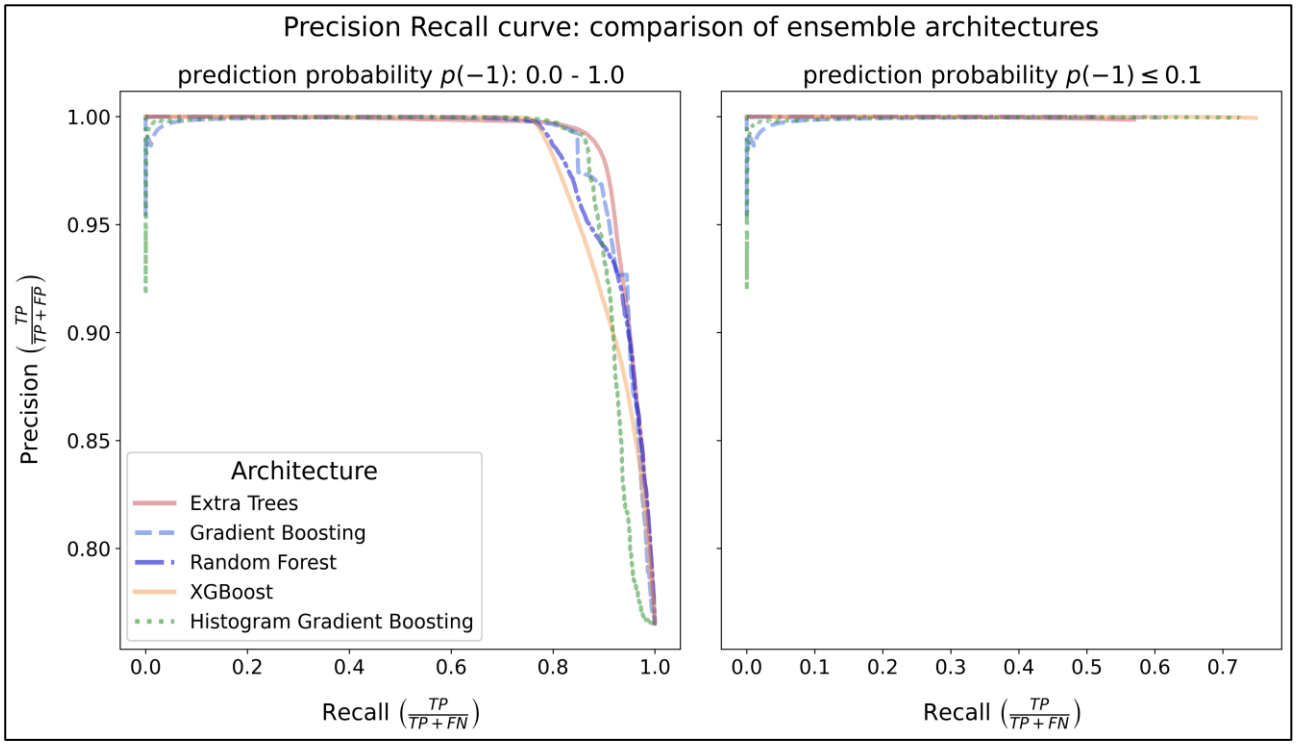


Figure 3.2 Precision-recall graph. The degree of classification accuracy of five decision tree ensemble architectures is compared. The rate of *precision* and *recall* is measured for each curve. **Left plot**: The curves are created using the full *prediction probability* p(-1) distribution of classified spectra. **Right plot**: The performance of spectra with p(-1) scores less than 0.1 for each architecture are compared. The legend in the lower left corner of the left plot colour codes the curves for each architecture.

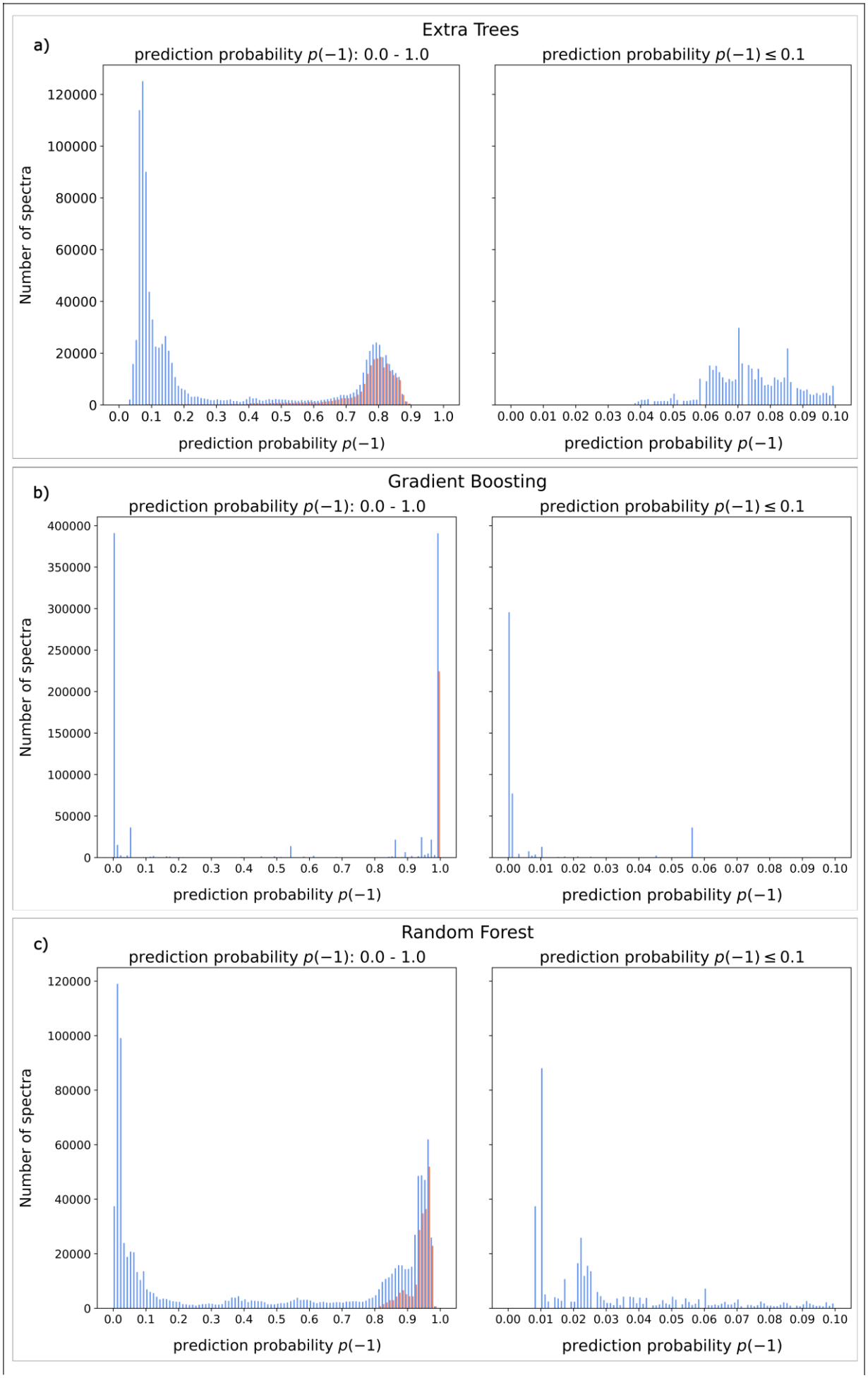
The AUROC and PR curves are directly related, as both are estimated from the same classified, ranked PSM dataset. In **Figure 3.2(left)**, the full p(-1) range reflects excellent precision scores, especially given that the y-axis begins at 0.7. Histogram Gradient Boosting and Gradient Boosting display slight variance, with a dip in *precision* at low *recall*/*sensitivity* values near 0.0. Interestingly, the confidently predicted spectra in **Figure 3.2(right)** maintain excellent *precision* scores despite having low *recall*/*sensitivity* and 1 – *specificity* rates. Therefore,

although these spectra display an unconvincing likelihood of being non-random true matches, they still hold some significance, as reinforced by their *precision* scores. Overall, a preliminary assessment indicates no clear signs of underfitting or overfitting at this juncture.

At this point, it may be questioned why the curves in the AUROC or PR plots display such wide distributions across each metric. The PR curve for example, has spectra yielding a high *precision* but low *recall* likely due to an increase in false negatives. An FN in this instance, may represent a poorly classified decoy or a random target spectrum. The PSM dataset used in this task contains 1,187,495 sequences, making it difficult to discern the trends of individual spectra using these plots at this scale. Since both target and decoy spectra are included in the AUROC and PR graphs, it is further challenging to identify which class type is causing certain trends, such as an increase in the FN rate. It is common practice to include additional metrics, such as confidence intervals, to support the validity of ROC graphs; this will be elaborated on in the discussion chapter of this work. To better visualise the relationship between the classified target and decoy spectra, a secondary evaluation using histograms is explored next.

3.3 Histograms

Generally, learning models built to conduct binary classification are evaluated for overfitting and underfitting as discussed in Chapter 1.6.1. To recap, overfitting arises from a model learning irregular patterns from the training sample, making it unable to generalise when classifying unseen data. Underfitting occurs due to a model over-simplifying patterns and failing to objectively classify data, resulting in biased learning. A reliable model should be able to generalise fairly while making confident predictions. This can be observed by sorting the classified spectra by a test score (i.e. *prediction probability* $p(-1)$) and observing the distribution behaviour of the target and decoy PSMs across the score range of 0.0 – 1.0. To visualise this, a rank histogram plot was created for the classified PSM dataset from each decision tree ensemble model, which can be viewed as a collection in **Figure 3.3**. The non-random and random PSMs correspond to the target and decoy class types, respectively. Just as with the AUROC and PR graphs in the previous subchapter, the classified PSM dataset for each architecture is viewed under two conditions: **Figure 3.3(left)** views the complete $p(-1)$ distribution range, while **Figure 3.3(right)** focuses on spectra with a $p(-1)$ range less than 0.1.


Extra Trees
a)
prediction probability $p(-1)$: 0.0 - 1.0
prediction probability $p(-1) \leq 0.1$
Number of spectra
prediction probability $p(-1)$
Gradient Boosting
b)
prediction probability $p(-1)$: 0.0 - 1.0
prediction probability $p(-1) \leq 0.1$
Number of spectra
prediction probability $p(-1)$
Random Forest
c)
prediction probability $p(-1)$: 0.0 - 1.0
prediction probability $p(-1) \leq 0.1$
Number of spectra
prediction probability $p(-1)$

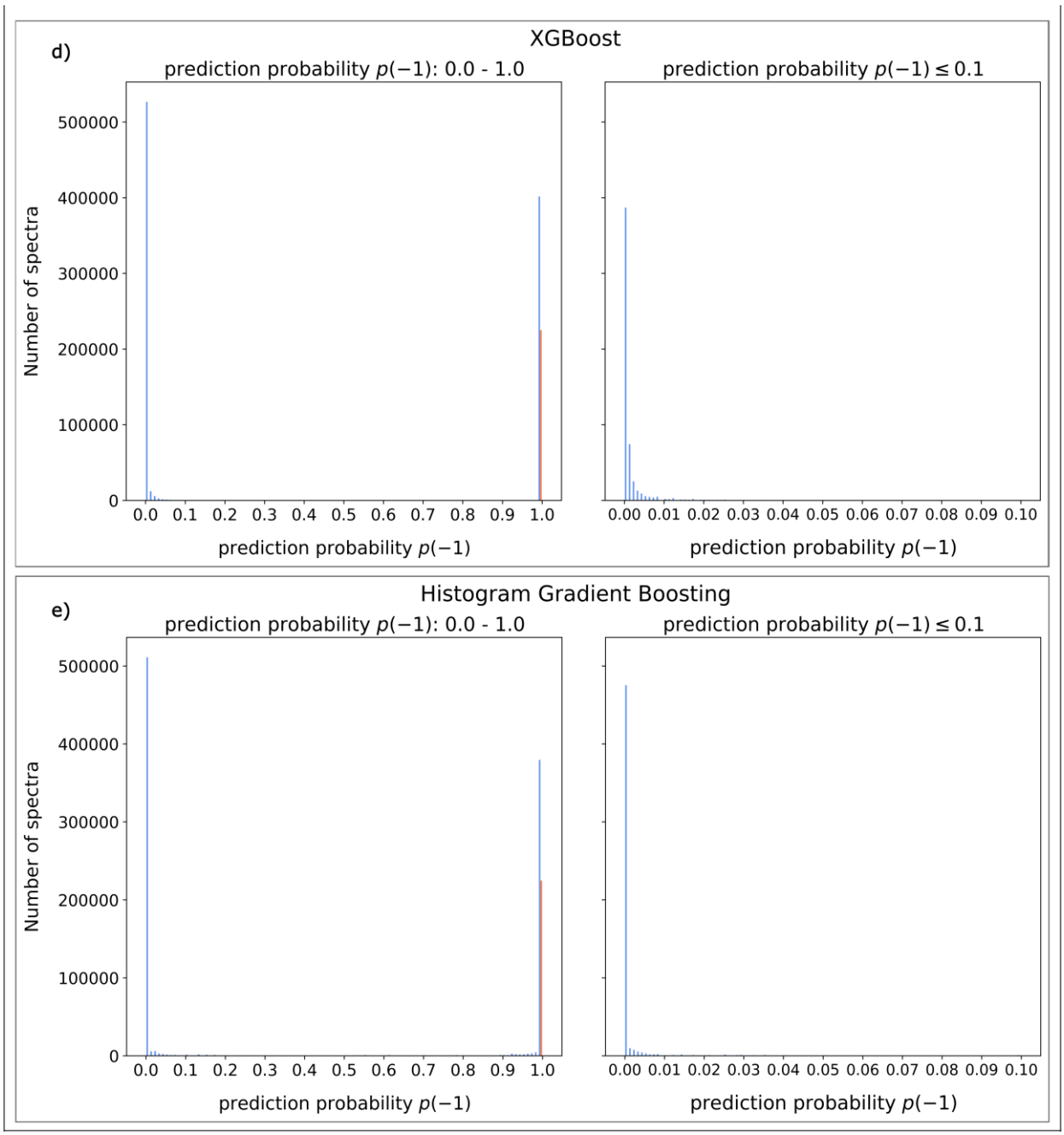


Figure 3.3 Histograms of the PSM dataset classified by decision tree architectures. Classified non-random (blue peaks) and random (red peaks) spectra are ranked in ascending order by the *prediction probability* p(-1) test statistic. There are two histograms for each architecture. The histograms (**left**) are the spectra scored within the p(-1) range 0.0 – 1.0. The histograms (**right**) are the spectra with p(-1) score less than 0.1. **(a) Extra Trees**. **(b) Gradient Boosting**. **(c) Random Forest**. **(d) Gradient Boosting**. **(e) Histogram Gradient Boosting**. The spectra are dispersed into 100 bins for each histogram.

The histograms of **Figure 3.3** are used to evaluate the behaviour of classified spectra by observing the distribution trends of non-random and random PSMs, and to evaluate model performance. A biased model would be observed under two conditions, the first being that all non-random spectra are located far left at interval 0.0 – 0.1, while random spectra are clustered far right at interval 0.9 – 1.0. This pattern is interpreted as the model being overly confident, having failed to objectively discriminate between class types. An initial observation reveals that none of the architectures displays bias conditions. However, the *Boosting* models: Gradient Boosting, XGBoost, and Histogram Gradient Boosting (**Figure 3.3[b, d, e]**) appear to impart a skewed bias or unbalanced classification, observed by the lack of spectra across the full *prediction probability* p(-1) range. The learning models of the Nagilums Tree pipeline were trained to discriminate between target PSMs at a 1% FDR and decoy PSMs, and consequently, some degree of class separation is expected. *Boosting* models are designed to increase the significance of *weak-learners* (see Chapter 1.8.2.2), which corresponds to the decrease in spectra between the 0.1 - 0.9 interval observed in their histograms. The *Bagging* models (see Chapter 1.8.2.1): Extra Trees and Random Forest (**Figure 3.3[a, c]**), display a well-balanced distribution of target and decoy spectra. These methods aim to maximise their learning strategies by learning the intricacies of the dataset. This robust approach is reflected in the broader distribution of spectra class types in their histograms.

With regards to the p(-1) distribution range, spectra with scores above 0.5 are deemed insignificant. This is due to the learning models classifying these spectra with a high probability of them being a negative class type. This aligns with an increase in decoy PSMs, interpreted as true negatives, observed beyond this interval. Target spectra at this interval are observed as false positives, i.e. the random matches occurring within the canonical proteome (see Chapter 1.5, **Figure 1.9**).

A balanced model would display both non-random and random PSMs at each interval, indicating the degree of similarity or likeness between the class types. Here, an interval is in steps of 0.1 along the p(-1) score range, as depicted on the x-axis of the histograms. The second condition for evaluating performance bias is if there are more random than non-random PSMs occurring at each p(-1) probability interval. This would be observed, for each interval, as a longer red peak than blue peak within either of the histograms. In each of the architecture's histograms, this pattern is not observed beyond the p(-1) 0.5 probability distribution; however, it is difficult to determine the frequency of both class types below this range. A solution to this issue is described next.

3.4 Counting function performance evaluation

The purpose of the *counting function* was to collect a high frequency of significant true match non-random spectra, which is observed by the high blue peaks of true positive non-random (target) spectra in each architecture's histograms (**Figure 3.3(right)**). The Extra Trees and Random Forest architectures exhibit a wider distribution of spectra compared to the *Boosting* models (Gradient Boosting, XGBoost and Histogram Gradient Boosting). The PSM dataset used in this task consisted of 961,663 target spectra, and a substantially smaller amount of 225,832 decoy spectra that are unobservable in either of these histograms, despite the number of bins of the histograms being set to 100, i.e. peak visualisation is improved by reducing the scale of the y-axis. To address this issue, a comprehensive overview of the total number of non-random and random PSMs with p(-1) scores below 0.1 are observed in **Figure 3.4**.

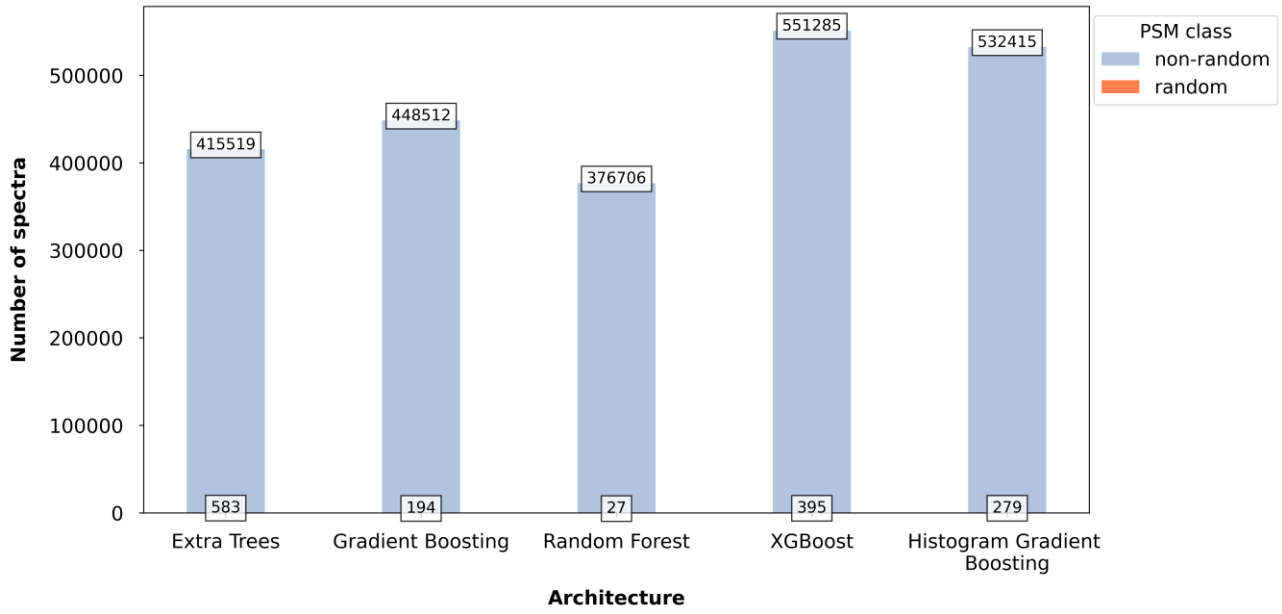


Figure 3.4 Total number of significant spectra (p(-1) < 0.1) by architecture. The PSM dataset consisting of non-random (target) and random (decoy) spectra, was processed by five decision tree ensemble architectures. The total number of classified spectra with a p(-1) score less than 0.l are compared. The legend in the upper-right corner provides a colour key for each PSM class type.

In this display, the low frequency of false negative random (decoy) PSMs is visualised, and the occurrence of both class types can be confirmed. Here, the *Boosting* models are observed to have classified more spectra than the *Bagging* models. XGBoost classified more than half of the target spectra as non-random true matches at 551,285, followed by Histogram Gradient Boosting at 532,415 and Gradient Boosting at 448,512. Extra Trees classified 415,519, and

Random Forest had the least, at 376,706. Considering spectra with p(-1) scores above 0.5 are insignificant, PSMs below this threshold are acceptable as moderately significant. Non-random target spectra scoring within the p(-1) range of 0.1 – 0.5 are still representative of possible true matches. In acknowledgement of this, **Figure 3.5** was created to observe the total number of spectra with a p(-1) below 0.5, which includes those below 0.1. A summary of the total number of spectra from **Figure 3.4** and **Figure 3.5** is provided in **Table 3.2**, which includes the percentage, rounded up, of the non-random spectra that were classified in both intervals. Here, Extra Trees surpasses the other architectures and has classified 70% of the target spectra with potential interest, aligning with its histogram (**Figure 3.3[a]**) that exhibits spectra with the widest distribution of p(-1) scores. Random Forest classified 480,742 non-random spectra, which is nearly half of the target spectra from the PSM dataset. The majority of the non-random spectra from the *Boosting* models were classified with a p(-1) score below 0.1. The difference in distribution is subjective; therefore, the overall correctness of spectra within this p(-1) range is evaluated.

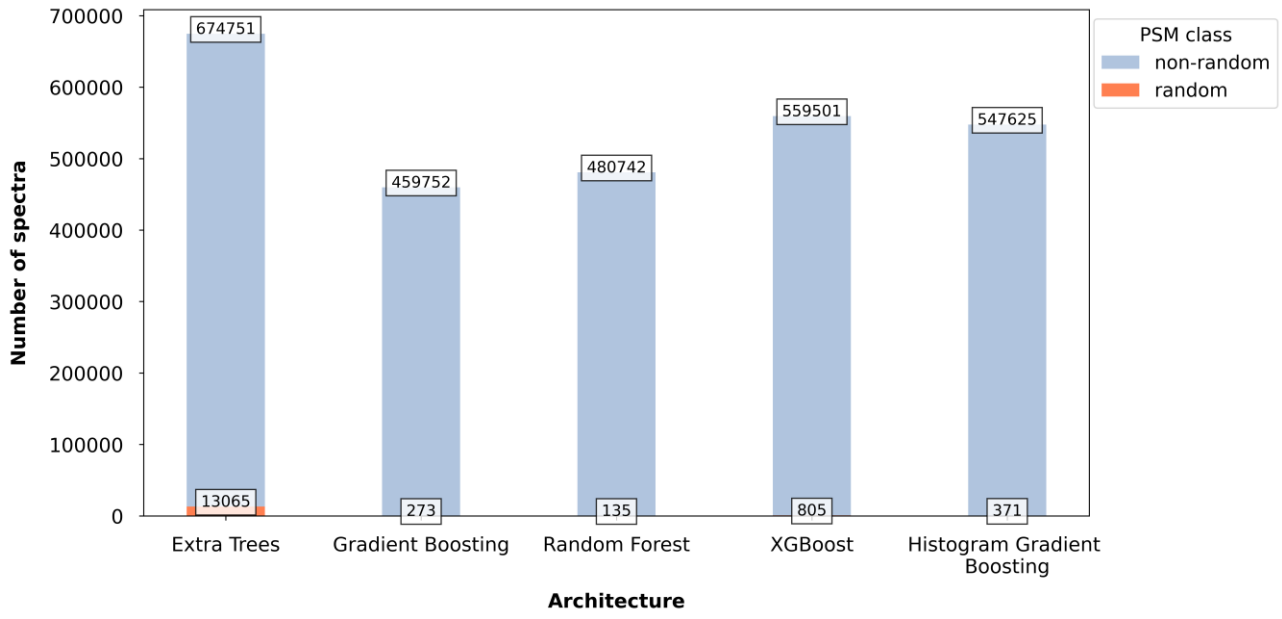


Figure 3.5 Total number of acceptable spectra (p(-1) < 0.5) by architecture. The PSM dataset consisting of non-random (target) and random (decoy) spectra, was processed by five decision tree ensemble architectures. The total number of classified spectra with p(-1) scores less than 0.5 are compared. The legend in the upper-right corner provides a colour key for each PSM class.

Table 3.2 The proportion of non-random PSMs classified for each decision tree architecture.

Number of non-random (target) PSMs	Extra Trees	Gradient Boosting	Random Forest	XGBoost	Histogram Gradient Boosting
$\boldsymbol{n = p(-1) < 0.1}$	415,519	448,512	376,706	551,285	532,415
$\% = \frac{\boldsymbol{n}}{\boldsymbol{961,663}}$	**43**	**47**	**39**	**57**	**55**
$\boldsymbol{n = p(-1) < 0.5}$	674,751	459,752	480,742	559,501	547,625
$\% = \frac{\boldsymbol{n}}{\boldsymbol{961,663}}$	**70**	**48**	**50**	**58**	**57**

3.4.1 FDR

The learning algorithms of each architecture were trained to discriminate between decoy spectra and target spectra, initially selected from the PSM dataset ranked by the *pep* scoring function and filtered at a 1% FDR threshold (**Figure 2.1**, Step 1). Thereafter, the *counting function* estimated a new FDR as an attempt to improve classification. The FDR is a global error rate metric and does not exclusively include significant data points. To understand this, the classified PSM dataset of each architecture is divided into three p(-1) intervals in **Figure 3.6**. Non-random target spectra with a p(-1) score below 0.1 are less likely to be an incorrect match, while those less than 0.5 are threshold as acceptable. The number of spectra within these two intervals is roughly similar to their total counts in the previous chapter (**Table 3.2**). Spectra with p(-1) scores above 0.5 are insignificant, and their frequency within the 1% FDR threshold is also observed in **Figure 3.6**.

Interestingly, the *Boosting* and *Bagging* models classified false positives uniquely in contrast with their classification trends of significant spectra. Gradient Boosting produced the most insignificant spectra at 170,763, followed by Histogram Gradient Boosting with 94,198. XGBoost is seemingly the more reliable *Boosting* model, having classified 23,941 insignificant spectra. Notably, Extra Trees produced the most comprehensive p(-1) distribution (**Figure 3.3[a]**) and classified no insignificant spectra at the 1% FDR threshold, compared to Random Forest, which produced 110,479. Therefore, Extra Trees is observed to be the most reliable model out of all the architectures and is thus used for implementing Task 2.

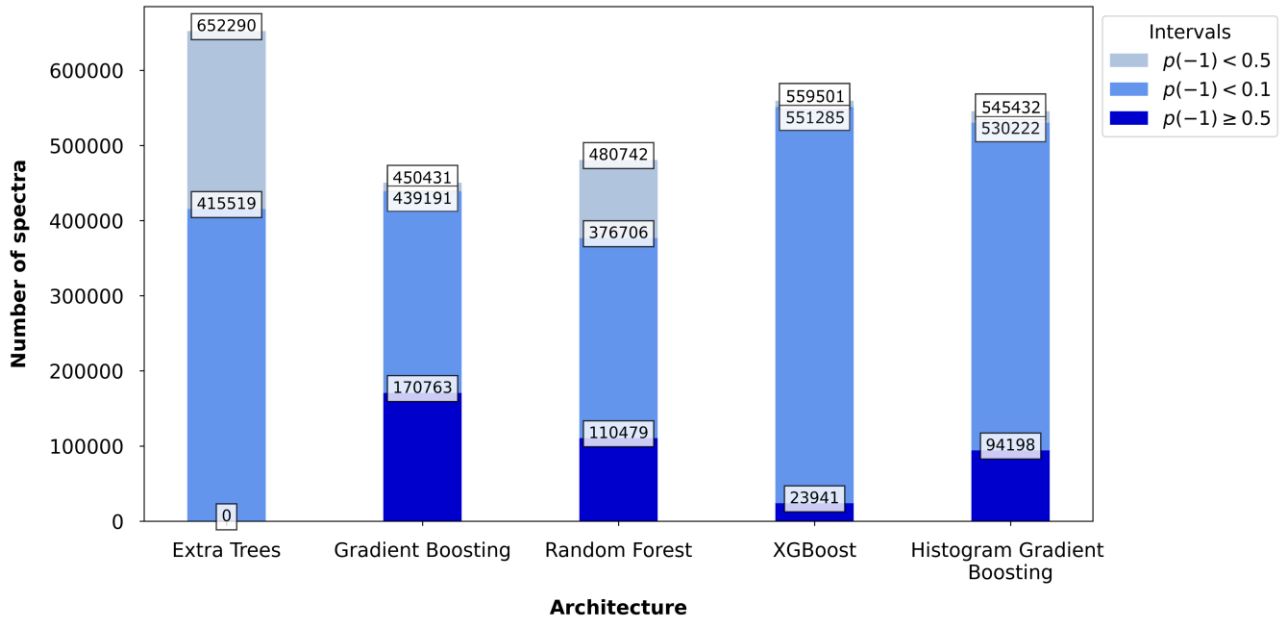


Figure 3.6 Total number of non-random PSMs within a 1% FDR of *p*(-1) intervals. Five decision tree architectures processed a PSM dataset and produced varying counts of non-random (target) spectra within a 1% FDR. The three different *p*(-1) intervals represent the significance of the spectra. The legend in the upper-right corner provides a colour key for each *p*(-1) interval.

3.5 Task 2: Decoy variant concept

Variant peptide sequences of rare human health conditions, e.g. monogenic diabetes (MODY and NDM), develop from a single variant on associated genes and occur at low frequencies despite having high penetrance in terms of impact (Bainbridge, 2020). Due to this, they are often unobservable in protein identification tasks when combined with a disproportionately larger canonical proteome during search annotation, to which they also share a high degree of similarity.

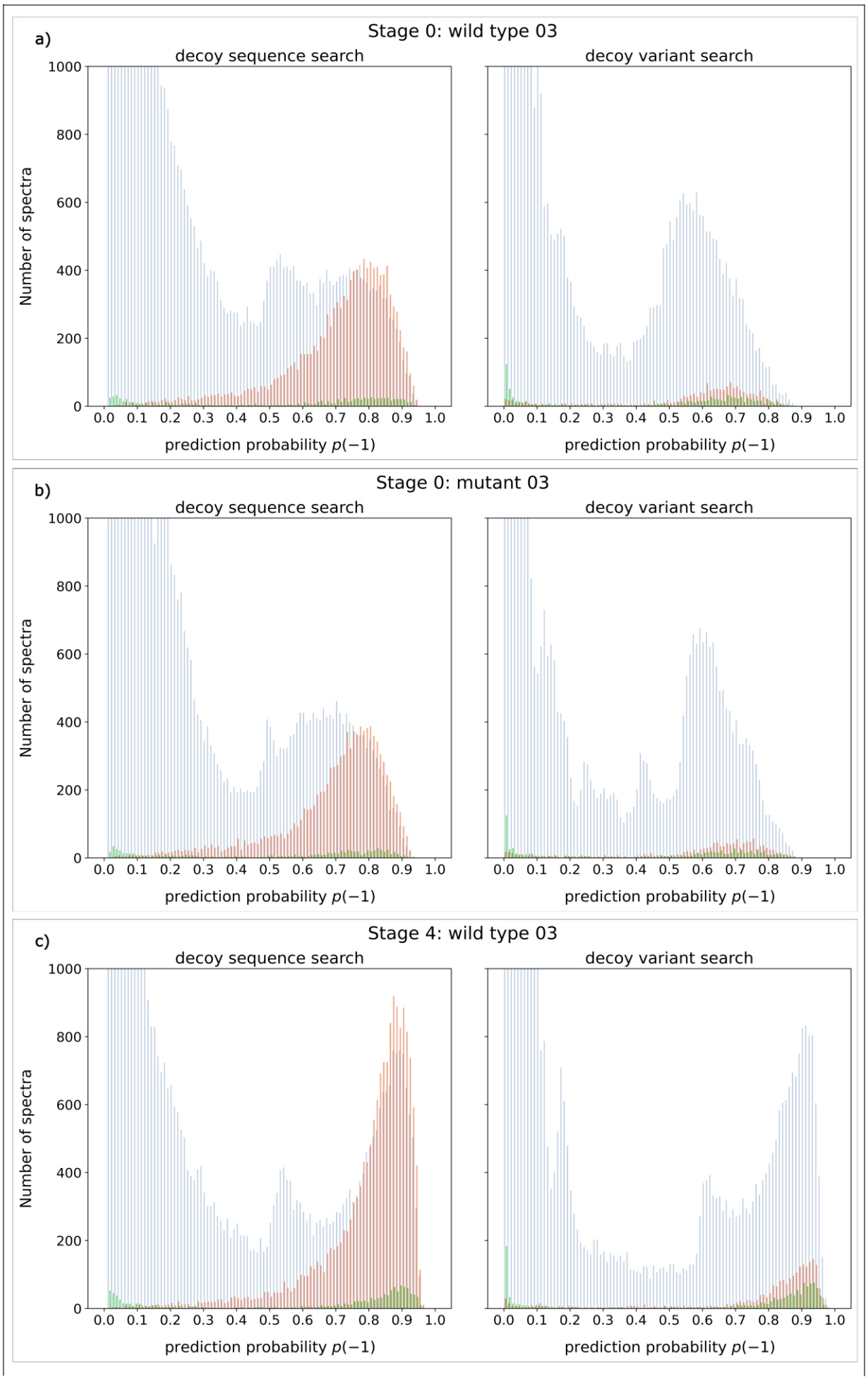
To this end, the method of Task 2 attempts to improve the statistical significance of annotated variant peptide sequences by employing the TDA. Task 1 explored the classical method of TDA created by Elias and Gylgi (2007), which concerned creating decoy peptide sequences from the combined target set of the canonical and variant proteome. Here in this task, the TDA was adapted to investigate the concept of *decoy variant* PSMs. To achieve this, the canonical and haplotype variant PSMs were instead separated into two individual target class types with identifying label names, which each generated decoy sequences. Together, these four class types were annotated to the proteomic data of iPSCs genetically modified to exhibit insulin resistance, by altering the HNF1A (MODY3) gene.

Two iPSCs development stages of pancreatic protein expression: *Stage 0* (early development) and S*tage 4* (pancreatic progenitor), were monitored for *wild type* (unmodified) and *mutant* (modified) cells. Three different cell lines were created to produce a total of 12 iPSC proteome sets that underwent mass spectrometry and annotation. The resulting iPSC PSM datasets were processed using the Nagilums Tree pipeline.

Pertaining to the performance evaluation of the five decision tree ensemble architectures in Task 1, Extra Trees was selected to perform the classification for Task 2. This investigation is a novel pursuit in which the potential of using *decoy variant* PSMs as a null model is evaluated. To the knowledge of this study, this has not been previously reported at this current time. In the next subchapters, the classification performance of the two decoy PSM search strategies (*decoy variant* and *decoy sequence*) for all 12 iPSC datasets is evaluated. First, the probability distribution trends of the different PSM class types are observed using histograms. Then, the distribution of PEP scores in relation to *prediction probability p*(-1) estimates is assessed, which is reinforced by a total count comparison of significant spectra. Lastly, the PSMs associated with the 14 MODY genes are assessed for statistical significance inferred by the PEP scores. The figures in this chapter, along with supplementary reports, are available on GitHub at: Thesis and supplementary figures.

3.5.1 Histograms iPSC files

As in Task 1, histograms are used to examine the distribution of classified PSMs. However, in this report, events of overfitting and underfitting are not investigated. Instead, the histograms provide the means to observe the behaviour of decoy PSMs. The 12 iPSCs files were processed under two search conditions, and 24 histograms were produced; therefore, for brevity, eight are displayed here, while the remainder are kept as Supplementary. The histograms are labelled according to the iPSC development stage, cell type and cell line, e.g. *Stage 0: wild type 01, Stage 4: mutant 02,* etc. The datasets displayed in **Figure 3.7** are from cell line 03: *Stage 0: wild type 03*, *Stage 0: mutant 03*, *Stage 4: wild type 03* and *Stage 4: mutant 03*. Two histograms are provided for the iPSC datasets, each representing the performance of a search strategy. The *decoy sequence* strategy is displayed in **Figure 3.7(left)**, and the *decoy variant* strategy is **Figure 3.7(right)**. Note, the terminology of 'mutant' used hereon is in the context of the altered iPSCs and is not associated with any persons. This distinction is made to avoid confusion with the term 'variant' as used in the context of peptide sequences associated with human health conditions, which is later mentioned in the discussion chapter.


a)
Stage 0: wild type 03
decoy sequence search
decoy variant search
Number of spectra
prediction probability $p(-1)$
b)
Stage 0: mutant 03
decoy sequence search
decoy variant search
Number of spectra
prediction probability $p(-1)$
c)
Stage 4: wild type 03
decoy sequence search
decoy variant search
Number of spectra
prediction probability $p(-1)$

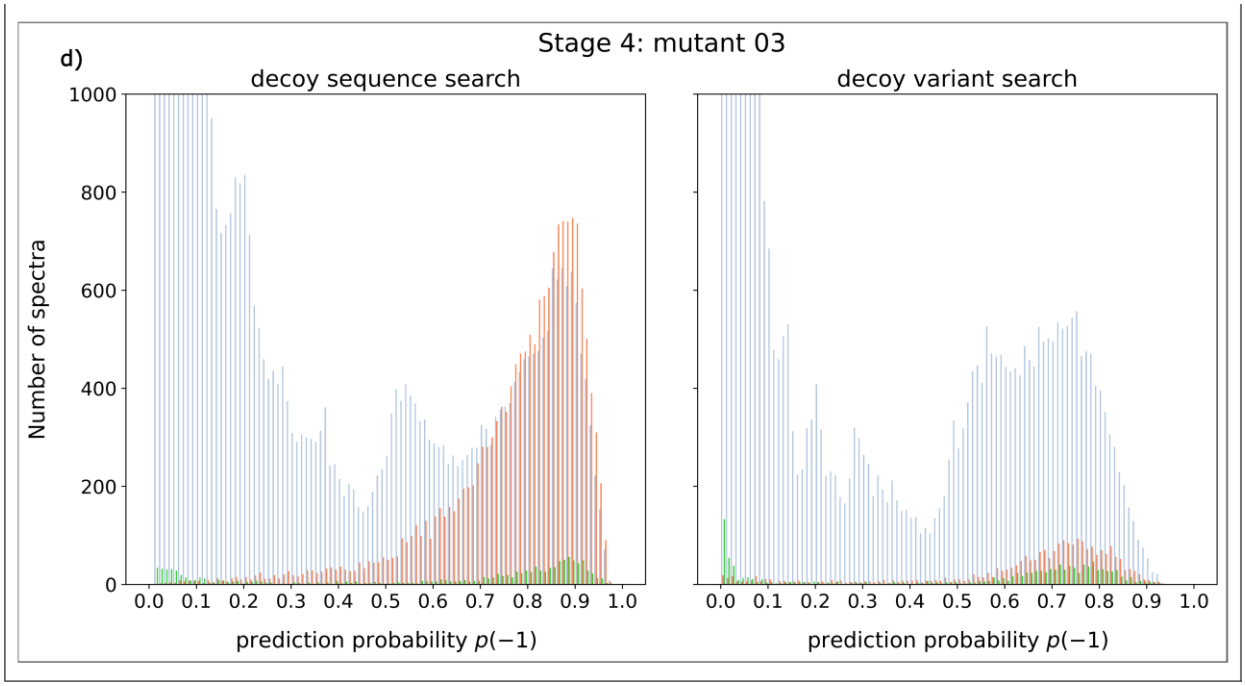


Figure 3.7 Histograms of four processed iPSC PSM datasets. The PSMs for two development stages (Stage 0 and Stage 4) of wild type and mutant iPSCs are ranked in ascending order by the prediction probability p(-1) test statistic. Cell line 03 is observed in these histograms: **(a) Stage 0: wild type 03**, **(b) Stage 0: mutant 03**, **(c) Stage 4: wild type 03**, and **(d) Stage 4: mutant 03**. Three PSM class types are observed for performance: canonical (blue peaks), decoy (red peaks) and variant (green peaks). Two histograms representing search strategies are provided for each iPSC PSM processing; **left plot**: decoy sequence, and **right plot**: decoy variant. The spectra are dispersed into 100 bins for each histogram.

Protein expression was different for each cell line, and various levels of peaks are to be expected. Additionally, all four PSM class types existed in varying frequencies, e.g. the iPSC dataset *Stage 0: wild type 03* contained 88,331 *canonical* PSMs, 9,854 *decoy sequence* PSMs, 864 *variant* PSMs and 1,344 *decoy variant* PSMs. The other iPSC datasets contained PSM class type ratios at similar portions. The canonical PSMs are consistently more frequent, producing a higher volume of classified spectra, particularly around the p(-1) interval 0.0 - 0.1. Regarding the histogram plots, it is more relevant to observe the distribution trends of the decoy and variant PSM class types. Therefore, for all of the histograms produced for Task 2, the y-axis is scaled down for better peak visibility of their lower frequencies, and the blue peaks of the canonical PSMs are shortened.

If it is questioned at this point as to why there are high portions of PSMs in *Stage 0* iPSCs it is explained that they are early developed pancreatic β-cells and are not 'day 0' or 'empty' (van de Bunt *et al*., 2016). They are expected to produce protein products. Importantly, the histogram

distribution trends are not an indicator of protein expression levels and simply indicate the PSMs observed after processing.

The distribution of classified iPSC PSMs is observed using the *prediction probability* p(-1) scores applied by the learning algorithm under Scikit-learns framework (Pedregosa *et al*., 2011). To recap, these scores range from 0.0 to 1.0, where values closer to 0.0 indicate a lower likelihood of the PSMs being incorrect matches or false positives. The decoy PSMs in both strategies represent the negative model for the null hypothesis. The expected distribution is that the positive class type (blue peaks) skews toward the left (closer to 0.0), while the negative class type (red peaks) skews to the right (closer to 1.0). Spectra at $p(-1) < 0.5$ are considered statistically significant, and those above this threshold are interpreted as false positives.

Three main observations can be made from the iPSC histograms (**Figure 3.7**) regarding the distribution trends of the *canonical* PSMs, *variant* PSMS and both decoy strategy PSMs, with the latter two being of greater importance. First, the *canonical* PSMs clearly skew left despite the limited y-axis scale. Second, the *decoy sequence* PSMs (**Figure 3.7(left)**) are skewing right as expected; however, interestingly this is similarly observed for the *decoy variant* PSMs (**Figure 3.7(right)**). This is an indicator that the concept of *decoy variant* PSMs has potential as a null model. This is of importance and will be explored in the discussion chapter of this work (see Chapter 4.3).

Lastly, the *variant* PSMs in both search strategies appear to skew left and right at first. However, variant peptides are highly identical to canonical proteome sequences, and in actuality the PSMs are displaying similar trends, although at a lower frequency. This can be observed with the dip in *canonical* PSMs nearing the p(-1) score of 0.5 in each of the histograms. More importantly, distinct spikes in *variant* PSMs are observed at a p(-1) value of 0.0 in the *decoy variant* search strategy (**Figure 3.7(right)**). This is particularly interesting in the *wild type* iPSCs **Figure 3.7[a, c]**, where no gene alteration occurred. This is another promotional point for *decoy variants* as a null model and will also later be discussed within the context of MODY regulation.

The false positive *variant* PSMs also appear to follow the trend observed in the *decoy variant* PSMs. The *mutant* iPSCs underwent a single gene alteration during experimentation, and the resulting impact on protein expression is undetermined at this point. However, the histograms of **Figure 3.7 [b, d]** suggest that more *variant* PSMs were classified as significant in the *mutant* iPSCs compared to the *wild type* samples (**Figure 3.7[a, c]**). To establish the frequency of statistically significant PSMs, the posterior error probability was employed.

3.5.2 Posterior error probability analysis

In the histograms presented in **Figure 3.7**, the *canonical* and *variant* PSM class types were exceptionally well discriminated from both *decoy sequence* and *decoy variant* search strategies. Spectra achieving a *prediction probability* p(-1) score below 0.5 are considered as acceptably significant, and it appeared there was an influx of *canonical* and *variant* PSMs within this range. To determine the viability of these spectra, the PEP was estimated on the iPSC PSMs during processing using the novel 'container' method. The PEP is a probability statistic with a range of $0.0 - 1.0$. Target spectra (*canonical* and *variant* PSMs) with PEP values closer to 0.0 are estimated as having a low chance of being a false positive among decoy spectra with similar p(-1) scores. They are thus interpreted as having a strong likelihood of being a true match (Käll *et al*., 2008).

In **Figure 3.8**, two linear graphs are provided, which depict the distribution of the PEP against the *prediction probability* p(-1) values of *canonical* and *variant* PSMs for the 12 iPSC PSM datasets. The first graph **Figure 3.8[a]**, displays the distribution under the *decoy sequence* search strategy and **Figure 3.8[b]** is the *decoy variant* search strategy. The graphs are marked by dashed lines to indicate thresholds of significance for both the p(-1) and PEP statistics. The PEP distribution in **Figure 3.8[a]** is observed to extend beyond 1.0 for spectra as p(-1) values increase. These spikes indicate under-sampling as there were more decoy spectra than target spectra beyond the p(-1) score of 0.5. Owing to the low frequency of decoy variant PSMs, the spectra in **Figure 3.8[b]** generally do not exceed a PEP above 0.5 expect for two occurrences within the *Stage 0: mutant 02* and *Stage 4: mutant 02*, likely due to single outliers. The increasing slope for all the iPSC datasets in both graphs somewhat indicates there was a fair distribution of positive and negative class labels. Reporting the verdict on this will be debated in the discussion chapter, as there was generally a larger portion of target spectra than either decoy class type PSMs.

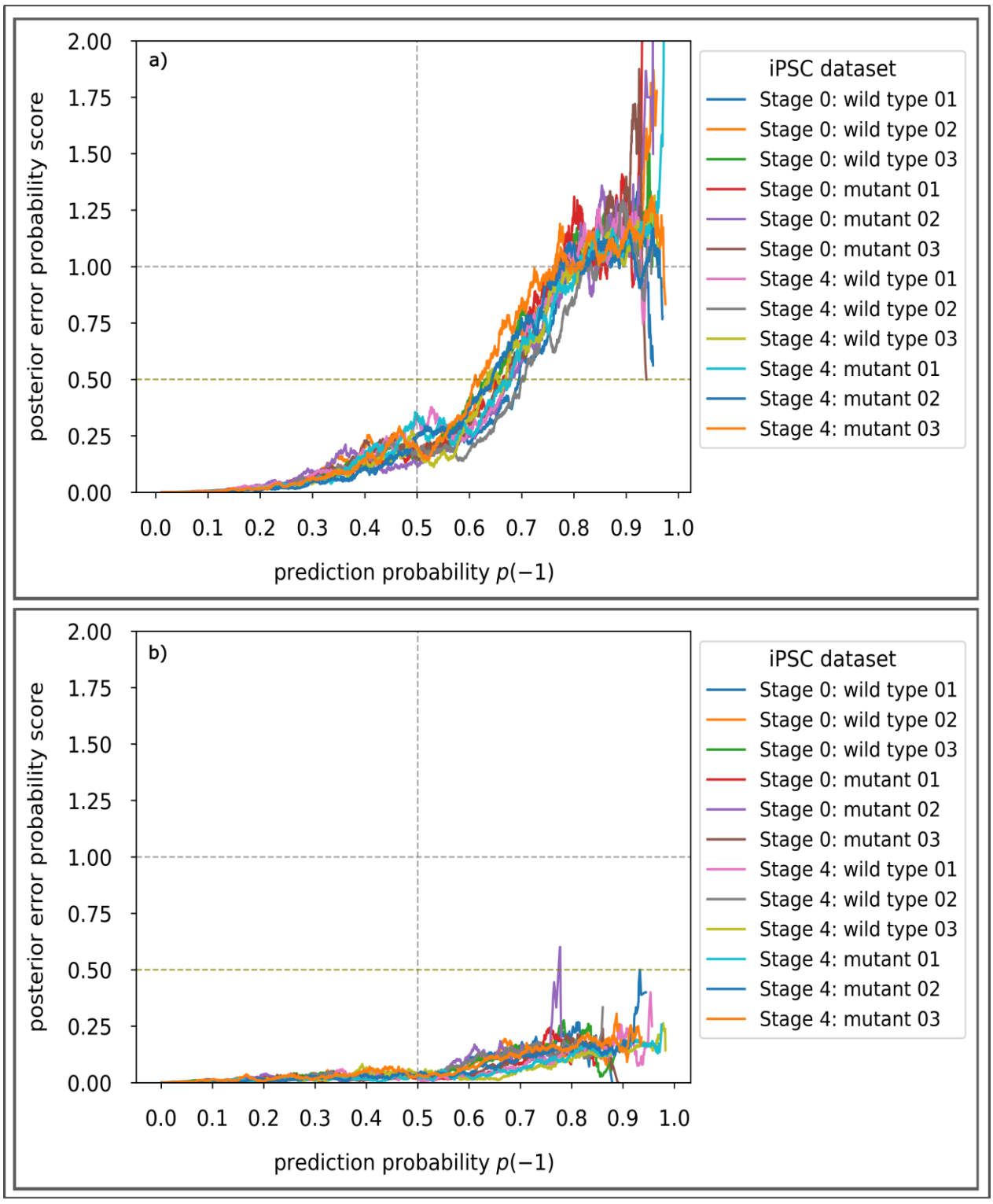


Figure 3.8 Distribution of significant spectra for two decoy search strategies. The posterior error probability of 12 processed iPSC PSM datasets is plotted against the *prediction probability* $p(-1)$ estimates. A threshold of $p(-1) < 0.5$ (grey vertical dotted line) and PEP < 0.5 (olive horizontal dotted line) is the limit for significantly classified spectra. The second grey dotted line at 1.0 is the cut-off for PEP estimates. **(a) Decoy sequence search strategy**. **(b) Decoy variant search strategy**. The legend on the right colour codes the 12 iPSC PSM datasets for each plot.

Generally, a PEP threshold of $\{< 0.1\}$ is specifically applied to peptide identification tasks (Käll *et al*., 2008; Mayo and David, 2022) where a stringent criterion is essential for reporting significance. Here, a PEP threshold of $\{< 0.5\}$ is set for general comparative purposes as two thresholds are applied. The first is the PEP and the *prediction probability* $p(-1)$ is the second. As stated previously, the spectra are given a threshold of $p(-1) < 0.5$ to be accepted as significant with a low possibility of being a false positive. Any spectra beyond this are deemed insignificant regardless of whether the PEP is below 0.1 or 0.5. Remarkably, both search strategies that produce spectra with a $p(-1) < 0.5$ also have PEP values well below 0.5. This selection of spectra is interpreted as highly significant. Continuing with the explanation of this interval (lower left quadrant in **Figure 3.8**), the *decoy variant* search strategy produced spectra with lower PEP values compared to those of the *decoy sequence* strategy. To evaluate this performance, a total count of the *canonical* and *variant* PSMs for both search strategies is provided.

In **Figure 3.9** two plots are provided, which present the total number of *canonical* spectra, **Figure 3.9[a]** and *variant* spectra **Figure 3.9[b]** within the thresholds of $p(-1) < 0.5$ and $PEP < 0.5$, produced from both search strategies. Interestingly, the *decoy variant* search strategy generated the most spectra across both plots, including the *canonical* PSMs. In certain cases, the *decoy sequence* strategy produced more canonical spectra, such as in the *Stage 0: mutant 01* and *Stage 4: wild type 01* datasets, and more variant spectra from the *Stage 4: mutant 01* dataset. Pancreatic development Stage 0 iPSCs are early-stage cells that express a diverse range of proteins. As iPSCs mature they produce proteins specific to the differentiated cell type. The assumption is that more spectra would be produced in Stage 4 than Stage 0; however, they are roughly the same for the *wild type* cells in both plots of **Figure 3.9**.

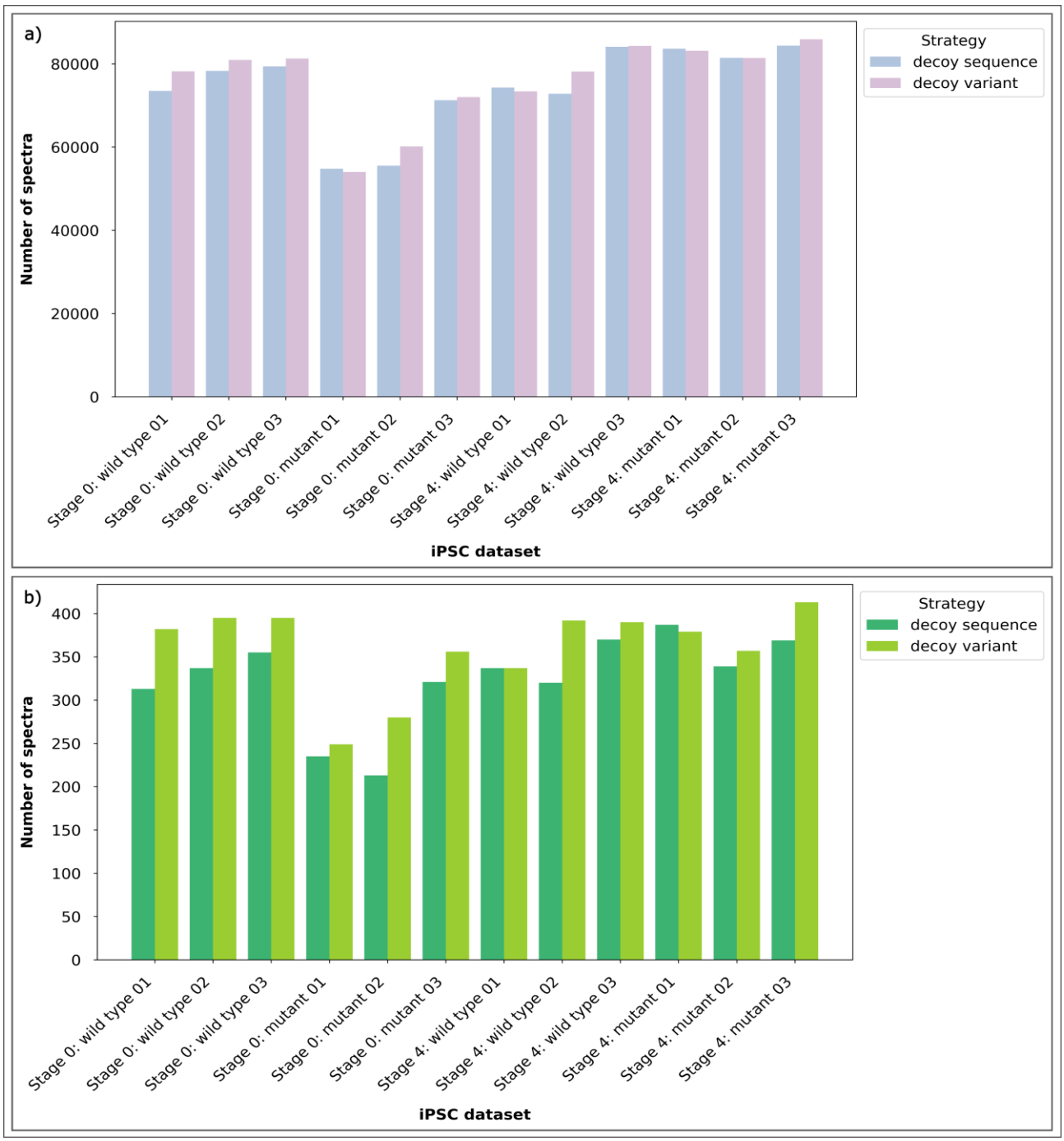


Figure 3.9 Performance evaluation of significant spectra produced by the decoy sequence and decoy variant search strategies. Twelve iPSC PSM datasets were processed using two search strategies. The number of canonical and variant spectra is within the thresholds of PEP $<$ 0.5 and $p(-1) <$ 0.5. **(a) Canonical PSMs**. **(b) Variant PSMs**. The legends in the upper-right corner of each plot provide a colour key for each search strategy.

Stage 4 represents the pancreatic progenitor phase, where cells are not fully differentiated but do express pancreatic protein markers. Here, iPSCs were grown into pancreatic β-cells, which produce insulin. A variant was engineered in the HNF1A (MODY3) gene to create mutated iPSCs. This variant does not inhibit insulin production directly but acts as a suppressor, affecting associated protein expression indirectly. This 'suppression' is possibly observed in effect from the fewer number of spectra of the *Stage 0: mutant* iPSCs, compared to the *Stage 0: wild type* cells, in both plots of **Figure 3.9**. However, by Stage 4 the *mutant* cells seem to have normalised this dysfunction and continue to develop at a rate that appears to surpass the *wild type* cells in the number of spectra produced. This increase is a possible indicator of compensatory overexpression due to the defective gene, though this remains undetermined at this point.

3.5.3 MODY genes

Due to the extent of this work, the protein-protein interaction network is not included in these results. However, to assess the performance of both decoy search strategies in identifying MODY-associated genes, two plots are provided in **Figure 3.10**: the decoy variant strategy is **Figure 3.10[a]** and the decoy sequence strategy is presented in **Figure 3.10[b]**. In these plots, the PEP distributions are used to investigate which MODY genes are present in each iPSC cell type and development stage. It is important to note that the discussion of these results will not delve into gene expression, as that cannot be extrapolated from these results. The PEP score is not an indicator of gene regulation or peptide abundance. The MODY genes displayed here are from the *canonical* labelled PSM class types. Upon inspecting the 12 iPSC PSM datasets, there were next to none variant PSMs with a matching MODY gene name. A single MODY gene was observed of the *variant* PSM class type from a *wild type* iPSC of the *decoy variant* strategy for ABCC8 (MODY12) visualised in the supplementary figure on GitHub: Supplementary figures. Four additional plots are included in this folder: two display the PEP distribution of the *canonical* and *variant* PSM class types created for the full list of possible MODY genes, and two others observe the *variant* PSM class for the 14 MODY genes. These plots are included to support future work in protein-protein interaction investigations.

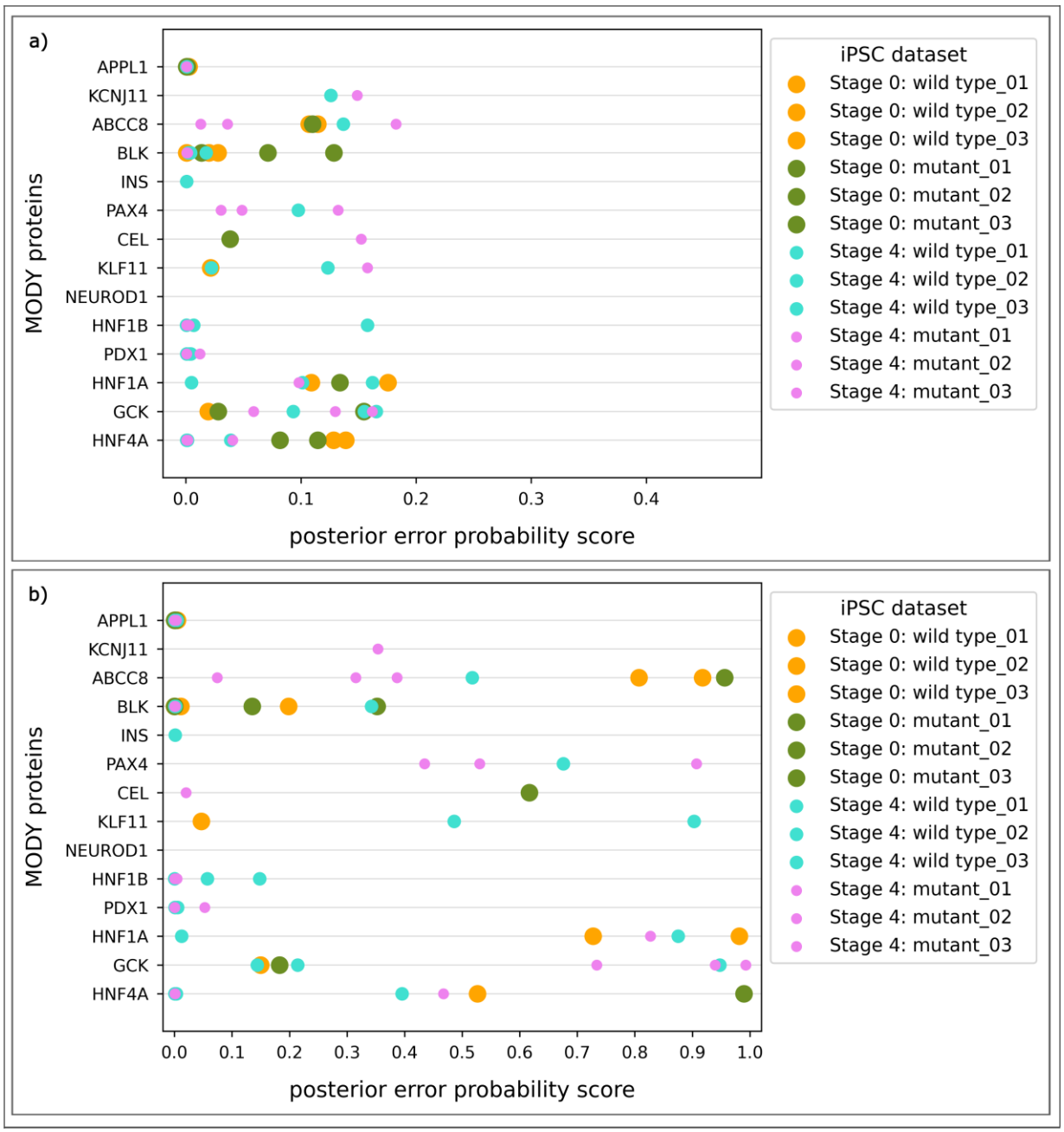


Figure 3.10 PEP distribution of 14 MODY genes. Gene identifiers of the 14 MODY genes are collected from the 12 classified iPSC PSM datasets for two decoy search strategies: **(a) Decoy variant search strategy**. **(b) Decoy sequence search strategy**. The legends on the right for both plots indicate the development stage and cell type of the iPSC datasets.

The *mutant* iPSCs were differentiated into pancreatic β-cells and engineered with a variant in the HNF1A gene to simulate the conditions of MODY3. Observing **Figure 3.10,** it appears the *mutant* cells of the HNF1A gene were not as expressed as the *wild type* cells. The *decoy variant* strategy (**Figure 3.10[a]**) produced only a single cell with associated *canonical* PSMs for Stage 0 and Stage 4, and the *decoy sequence* strategy had one at Stage 4 **Figure 3.10[b]**. This possibly confirms that the HNF1A gene was suppressed, and despite the increase in spectra observed for *mutant* cells of Stage 4 in **Figure 3.9**, it appears there was more of an indirect effect on interacting proteins. This is suggested by, **Figure 3.9[b]** confirming the presence of *variant* PSMs in all iPSC datasets. Although statistically significant spectra were produced from both strategies, none are associated with these 14 MODY genes. These are all canonical PSMs, and if these results are accurate, this suggests that dysfunction of the HN1A gene does not lead to the suppression of other MODY genes, and the resulting symptoms beyond insulin loss are due to interacting proteins. It is thus important to conduct protein-protein interaction analysis and assess the genes with which these *variant* PSMs are associated.

The gene identifiers for the PSMs of each iPSC dataset were included in the theoretical dataset prior to peptide annotation. Assuming that the experimental and theoretical datasets were accurate, the MODY genes presented here indicate that a single peptide associated with their protein structure was present within the iPSC files. However, the number of peptide sequences composite of each MODY protein is not confirmed, as no PSM to protein-inference study was conducted; therefore, true protein expression cannot be confirmed. What can be learned from these results, though, is how the different decoy search strategies impact the confidence of peptide identification.

At this juncture, it appears that the NT pipeline was able to capture the behaviour of the HNF1A variant. In addition to this, the *decoy variant* strategy was able to identify one more PSM associated with MODY than the *decoy sequence* strategy, although neither could produce the engineered variant of the HNF1A gene. The PEP is a probability distribution from 0.0 – 1.0 representing the most significant PSM to the least significant, respectively. In **Figure 3.10[a]**, the PEP range is no more than 0.2, whereas **Figure 3.10[b]** is at full scale. It appears that the *decoy variant* search strategy improved PEP resolution of the MODY genes; however, the ratio of *decoy variant* PSMs to the combined *canonical* PSM and *variant* PSM target class type was disproportionate (see Chapter 3.5.1). The topic of imbalanced datasets impacting statistical inference is explored in the discussion chapter.

4: Discussion

4.1 Introduction

The purpose of this study was to assess the discoverability of variant proteins causing paediatric diabetes. This was explored using proteogenomic techniques, and the investigation methods focused on the downstream post-processing of PSMs. A pipeline system called Nagilums Tree was designed to process a PSM dataset composed of target and decoy spectra using ML techniques. Then, two main implementation tasks were investigated using this system. The first task concerned the discrimination performance of five decision tree ensemble architectures. The second task explored the concept of decoy variants as a null hypothesis to improve the discriminative performance of variant peptides associated with MODY.

The broader topic of this work is rather expansive as it is positioned amidst the fields of computational methods, statistical inference, data analytics and human health research. To simplify the discussion, these areas will be explored as pertaining to a few key elements of the project overall. In essence, these explanations will explore general concepts of proteogenomics along with other concerns and areas of interest while using the results obtained in this work to expand on certain topics. It is presumptuous to comment on whether the construct of the NT pipeline is successful in itself; therefore, the implementation results are reviewed in proxy. A few details of the design are evaluated, complementary to the overall scope of this project. This chapter will holistically explore the relevancy of this project, erring on optimistic scepticism while addressing the caveats of proteogenomic methods, decoy variants as a null hypothesis, statistical inference and class imbalance.

4.2 Caveats of proteogenomics: from peptide extraction to PSM processing

Mass spectrometry technologies have made it possible to extract peptide sequences of proteome mixtures and are the *de facto* standard for protein identification research. LC/MS-MS uses high-throughput methods that produce massive amounts of peptide sequences at increasing rates and consequently have created a downstream analysis 'bottleneck' (Park *et al*., 2008). Therefore, there is an increasing need to interpret this data and determine its quality for proteomic research. The evolution of next-generation sequencing technologies is similarly improving genomic definition. More specifically, protein identification for gene expression analysis is paramount in investigations concerning disease pathology, and therefore there are particular requirements for reporting molecular bioprocesses (Nesvizhskii, 2014; Chick *et al*., 2015). The development of the proteogenomic field purposefully resolves these needs.

A classic proteogenomic workflow (Nesvizhskii, 2014) is designed to identify peptide sequences acquired through mass spectrometry by matching to a theoretical dataset for downstream analysis. An array of database search engines is available for peptide annotation, e.g. SEQUEST (Eng *et al*., 1994), X!Tandem (Craig and Beavis, 2004) and Mascot (Perkins *et al*., 1999), each offering unique algorithms to identify the most similar sequences. This is achieved using scoring functions to measure the ion intensity of peptide fragments, which can experience negative distortion resulting in varying MS/MS spectra peaks. Owing to this, it is possible for spectra to be misaligned or annotated to different peptide sequences across either search engine (America *et al*., 2006; Yu *et al*., 2006; Kwon *et al*., 2011).

Moreover, proteogenomic analysis often assumes that genetic databanks, e.g. Ensembl (Harrison *et al*., 2024), UniProt (Bateman *et al*., 2024) and RefSeq (Pruitt *et al*., 2007), contain correctly identified protein-coding genomic sequences, and their corresponding protein products of the entire human genome and proteome. This is unlikely to be true; therefore, peptide-spectrum matches inferred from a theoretical dataset are understood as presumptuous during the explanation of proteogenomic data analytics. Database search engines operate on the assumption that each observed spectrum is an accurate match (Nesvizhskii, 2014), and it is difficult to conclude which tool provides the better alignments without conducting an exploratory search (Kwon *et al*., 2011).

Another caveat of search engine annotation is that ion peak intensity measurements are skewed towards the canonical peptide weights (Chick *et al.*, 2015). This is mainly due to protein databanks being more abundant with canonical proteome sequences over variant isotopes. In turn this creates biased scoring functions, which rely on probability scoring metrics to infer peptide identity. In Chapter 1.5.1 and **Figure 1.10**, peptide similarity was explained as a threshold to infer the identification of a reference peptide within the search space. Restricting the threshold can reduce incorrect matches; however, peptides with low annotation scores are vulnerable to being unidentified or mismatched, despite being correct alignments (Lam, 2011).

A study by Chick *et al*. (2015) defined the differences in mass spectrometry measurement weights between variant and canonical peptide sequences. It was revealed that unmatched spectra are likely due to peptides with substoichiometric modifications (post-translational modifications). In the same study, it was further revealed that amino acid modifications were only correctly identified for sequences with a frequency above 90%. In Chapter 1.3.2.2 (**Figure 1.4**), the relationship between allele frequency and penetrance in terms of phenotypic effect regarding human health conditions was illustrated. These conundrums are considered given that this work aimed to identify rare, low-frequency MODY variants.

4.2.1 MODY gene expression: decoy variant concept

Although the iPSCs were not fully developed pancreatic β-cells, the pancreatic progenitor phase (Stage 4) does express proteins relevant to the organ (van de Bunt *et al*., 2016). The early phase (Stage 0) represents premature cells (van de Bunt *et al*., 2016). While protein expression levels cannot be determined from this work, it is reasonable to expect a greater diversity and abundance of proteins at Stage 4 compared to Stage 0. In **Figure 3.9**, the *wild type* cells produced similar levels of PSMs in Stage 0 and Stage 4, with slight increases and decreases across some iPSCs. The MODY gene expression plot **Figure 3.10** displays which MODY genes were identified from the 12 classified iPSCs datasets. Disregarding the influence of the decoy search strategies and PEP estimates for now, an initial observation reveals that the MODY genes identified at Stage 0 in both **Figure 3.10[a & b]** are much less frequent (around half the amount) than those at Stage 4. More specifically, certain MODY-associated genes, such as INS (MODY10), PDX1 (MODY4), PAX4 (MODY9) and HNF1B (MODY5), were identified only in Stage 4 *wild type* iPSCs, and not in Stage 0. This pattern aligns with the expectations of iPSC maturation; which is as cells progress through development, expression of pancreas-specific proteins increases. Based on these results, the classification system of NT appears capable of capturing relevant protein expression changes with iPSCs differentiation.

Continuing with **Figure 3.10**, although a few *mutant* iPSC PSMs associated with MODY genes were observed, the majority belonged to the *canonical* PSM class. A single PSM was observed (Supplementary figure GitHub link) as a *variant* PSM class of the *wild type* cell line; however, it was not associated with the HNF1A (MODY3) gene. Proteins are comprised of a family of fragmented peptides, which are each assessed for classification confidence to conduct PSM-to-protein inference (Shi and Wu, 2009). This was not conducted in this work, and instead inference was assumed from a single PSM with multiple occurrences were combined into an average (see Chapter 2.8.2.6).

The single variant occurring on the unaltered gene may be associated with other proteins or a random occurrence. Additionally, no *variant* PSMs were detected for the HNF1A gene or the other MODY genes of **Figure 3.10** as there was next to none listed in the iPSC PSM datasets. These observations underscore the challenges of identifying rare variant peptide sequences, as discussed previously in the caveats of proteogenomics. The scope of this project aimed to improve the resolution of rare MODY variants, although it is very possible that they were never isolated during experimentation, mismatched or unidentified during annotation, or the theoretical dataset was incomplete to begin with prior to PSM processing.

The histograms presented in **Figure 3.7** displayed a high frequency of confidently predicted *variant* PSMs under the *decoy variant* strategy; however, it is unknown which are attributed to high and low allele frequencies (**Figure 1.4**). The confidently predicted *variant* PSMs from these histogram plots are likely byproducts of unidentified MODY genes or other diabetic conditions. The diagram of **Figure 4.1** illustrates the wide range of genes known to be associated with major diabetes types, including monogenic diabetes (MODY and NDM). Considering that both T1D and T2D are afflictions of insulin dysfunction or resistance, it is expected to observe shared regulatory genes across subtypes, including interactive proteins influencing related metabolic disorders (see Chapter 1.2, **Figure 1.2**).

While it was the HNF1A gene with the frameshift variant, the possibility of associated pancreatic β-cell genes being affected is well-considered. To this end, the extended list of MODY genes included in the supplementary analysis holds interest due to more confidently predicted *variant* PSMs being observed in these plots for both decoy search strategies (Supplementary figures GitHub link). Protein-protein interaction network maps based on this extensive list may uncover novel connections and would be an exciting topic for future research. This is particularly compelling given the increased detection of *variant* PSMs under the *decoy variant* strategy. While definitive conclusions cannot yet be drawn, the concept of the *decoy variant* strategy merits further critique.

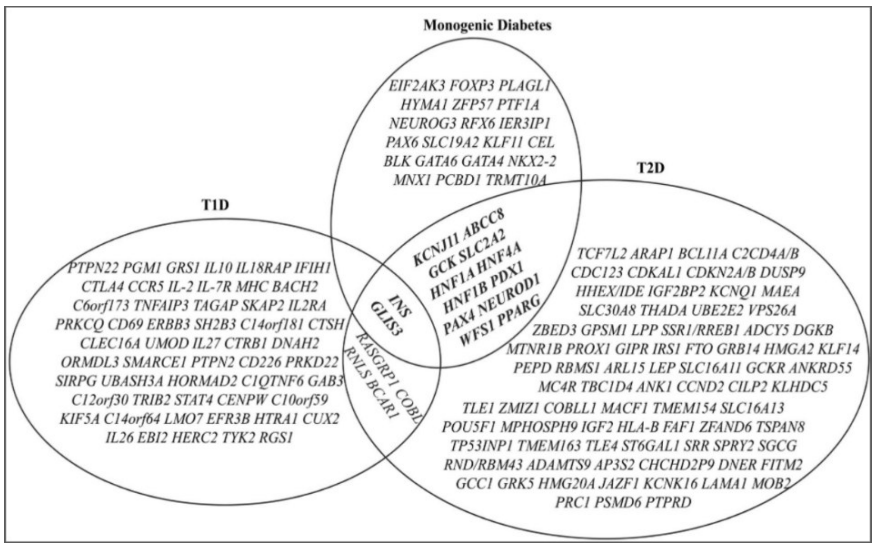


Figure 4.1 Diagram of genes associated with diabetes subtypes. The protein products of multiple genes have been identified as biomarkers of diabetes types: type 1 diabetes, type 2 diabetes and monogenic diabetes. Diabetes subtypes are known to share genes, indicated by the overlapping circles of the Venn diagram. This unmodified figure is from Yang and Chan (2016).

4.3 Decoy variant concept

A null model serves as a baseline reference in data analytics for hypothesis testing (see Chapter 1.5.2). It represents a simplistic assumption that observed effects are absent or unimportant, arising by chance with no meaningful influence from unrelated variable types. In the context of the TDA applied for classification assignments, decoy PSMs represent the null model. These decoy spectra are intentionally designed to simulate random or false matches, while target spectra are true protein sequences. Currently, the TDA explores the traditional competition between target and decoy peptides (Elias and Gygi, 2007), and there is no method to model the null distribution for incorrectly matched variant peptides specifically. The creation of the *decoy variant* PSMs for this work hence differed from the standard approach of reversing and shuffling the sequences. Instead, they were required to simulate deliberately mismatched variant peptides (see Chapter 2.8.2.1).

The purpose of Task 2 was to compare the standard decoy PSM method against the *decoy variant* PSMs for variant PSM prediction. Interestingly, the *decoy variant* strategy displayed an improved null distribution compared to the *decoy sequence* strategy, as observed from the histograms (**Figure 3.7**). This observation is reinforced by **Figure 3.9**, which displayed the total count of *canonical* and *variant* PSMs with a p(-1) and PEP value below 0.5 for both decoy strategies. Spectra within these thresholds are translated as possessing a variant with a strong likelihood of being correct and not a random occurrence.

While decoy PSMs may be common practice today, they are not a new concept. Earlier approaches retained the top-scoring PSMs from search engine annotation and labelled them as the positive class, while the remaining received a negative class label. This was explored in the studies by Ulintz *et al*. (2006) and Gonnelli *et al*. (2015) in which these 'target' and 'decoy' spectra were used to train and test a classifier validated on different PSM datasets, unlike in this work that used a self-trainer system. A study by Choong and Sung (2021) investigated eight variations of decoy sequence generation methods and their impact on classification and FDR metrics. Alternative studies, such as the work by Cannon *et al*. (2005) conducted search annotation using ion peak intensities of incorrect peptides as the null model.

There are many null model strategies to improve the accuracy of spectra ranking, and the *decoy variant* concept may have a place amongst these tactics based on the findings in this work. The *decoy variant* peptides possessed minor sequence adjustments, as opposed to reversing or shuffling, and were discovered to be highly similar to the *target variant* spectra during their development. Considering this, an overlap between the *true variant* PSMs and *decoy variant*

PSMs was expected to be observed from the histograms (**Figure 3.7(right)**). Instead, the *decoy variant* PSMs were clearly skewing to the right, towards a $p(-1)$ of 1.0 indicating a high likelihood of being random matches. The *variant* PSMs were also clearly skewing to the left, indicating an increasing chance of being a correct true match. This clear distribution supports the *decoy variant* concept.

The use of the *decoy variant* concept as it stands in its current form should, however, be approached with caution and critically evaluated. Arguably, it may not be a reliable method due to the mass spectrometry weights and search engine scoring functions not being adjusted to account for the classification of pure variant PSMs (as previously stated in the caveats of proteogenomics). Additionally, the *decoy variant* peptides were created with SNP alterations only, and other structural variants, such as indels and altered protein expression of frameshift events (see Chapter 1.3.2.1), were not considered. This may explain why no *variant* PSMs were observed for HNF1A (MODY3) in the MODY expression plot (**Figure 3.10**). These shortcomings should be addressed in the future to improve the reliability of this approach.

Another important consideration is that the *decoy variant* PSMs were disproportionately lower than the *decoy sequence* PSMs. To recap, iPSC cell line 03 contained only 1,344 *decoy variant* PSMs compared to 9,854 *decoy sequence* PSMs, which were modelled against 864 *variant* PSMs. Although the *variant* PSMs were combined with the *canonical* PSMs for classification, PEP analysis of the MODY genes revealed a big jump in significant spectra from the *decoy variant* PSM strategy (**Figure 3.10[b]**) compared to the *decoy sequence* PSM strategy (**Figure 3.10[a]**). The PEP plots of the classified iPSC PSM datasets likewise revealed a difference in trends for both of the decoy strategies (**Figure 3.8**). The concept of class-imbalanced datasets is of importance in proteomic studies and was mentioned throughout this work. It is an expansive topic regarding statistical inference and is explored next as pertaining to ML applications for PSM processing.

4.4 Imbalanced datasets

The creators of the target-decoy PSM concept Elias and Gygi (2007), outlined two key conditions for effective classification: 1) class proportions should be roughly equal to ensure fair discrimination, and 2) there should be an even distribution of target and decoy spectra when training a learning model to infer a test statistic. The second condition was accounted for in this work by using the *StratifiedKFold* method from Scikit-learns framework. The first condition, however, is not always achievable. This is due to the removal of duplicate and poor-quality decoy spectra during database creation, and PSM generation being dependent on search

engine annotation. As discussed in Chapter 4.3, various methods exist for creating decoy spectra, and each may impart unique influencing factors that can impact spectra abundance. In Task 2, the iPSC PSM datasets were divided into four class types, further reducing class proportions for classification. According to the logic of the TDA, the frequency of decoy PSMs represents the proportion of incorrect matches of the target spectra (false positives) (Elias and Gygi, 2007, 2010). However, this is debated within the community as shown by the rise in alternative error rate metrics that accommodate datasets with imbalanced target and decoy classes (see Chapter 1.5.3 and 1.5.4).

Imbalanced datasets are a phenomenon in data analytics that describes a disproportionate distribution of class types (Ryan and Chawla, 2013). This impacts two aspects of classification performance. The first is dependent on the ML architecture and classification style, and the second considers the statistical inference. Take for example, binary classification of one class accounting for 80% of the data points while the other represents 20%. This imbalance poses an issue for ML problems, as learning algorithms would have a bias towards the majority class. Additionally, error rate metrics can overestimate confidence scores leading to misleading conclusions. Typically, computational processes and error metrics can be adjusted to account for such occurrences; however, whether this imbalance poses an issue for determining the fitness of PSMs will be assessed in the succeeding subchapters. The prediction performance of the decision tree ensemble models is evaluated first, with particular attention to class imbalance and overall functionality for each implementation task. Lastly, the statistical inference methods and error rate metrics used in this work are evaluated regarding reporting PSM significance.

4.4.1 Machine learning

4.4.1.1 Imbalanced learning

Imbalanced datasets are a general concern in semi-supervised learning tasks (Huynh *et al*., 2021), as they can lead to learning algorithms being improperly trained, resulting in poorly predicted class labels for unseen data points. However, class label prediction accuracy is not the primary goal in PSM processing. Instead, the objective is to rank spectra based on annotation quality, which is achieved by assigning a probability metric that describes the likelihood of a correct or incorrect match. ML classification algorithms trained with labelled samples issue a probability test statistic onto unlabelled testing samples, providing more informative results. For this, the *prediction probability p*(-1) was employed under the Scikit-learn framework in this study. 'Prediction accuracy' in this sense was determined by the probability distribution trends

of the classified PSMs, which should reflect the characteristics of the target and decoy spectra. This is mainly to probabilistically justify any possible incorrect and correct matches of the target spectra (Elias and Gygi, 2010).

The target PSMs, which are a composite of canonical and variant peptides, are expected to skew towards a p(-1) of 0.0 indicating a strong likelihood of being a correct match (true positive). Target spectra below a p(-1) of 0.5 are still acceptable as significant peptides. The decoy PSMs, which represent random spectra, are expected to skew towards a p(-1) of 1.0. Target and decoy spectra above a p(-1) of 0.5 are identified as insignificant random matches. The frequency of decoy spectra across the p(-1) range indicates the degree of randomness of the target spectra, which was how significance was inferred. However, imbalanced PSM class types can influence the learning algorithm to learn patterns of the majority class type and overestimate spectra probabilities. These expected TDA trends are used to assess the performance of the decision tree architectures by referring to the characteristics of the PSM datasets used in both Task 1 and Task 2.

The distribution of spectra class types was notably diverse across the iPSC files in Task 2 (see Chapter 2.8.2). It was explained that a single iPSC file contained a class type frequency of 88,331 *canonical* PSMs, 9,854 *decoy sequence* PSMs, 864 *variant* PSMs and 1,344 *decoy variant* PSMs. Although *variant* PSMs were the main interest, they were combined with the *canonical* PSMs to form the target class for each decoy PSM classification strategy. This setup suggests a potential bias toward *canonical* PSMs, given their overwhelming majority. Acknowledging the *variant* PSMs alone, the learning models could also favour the decoy PSMs, considering the large difference in their frequencies. However, a good p(-1) distribution was observed for all PSM classes as viewed from the histograms (**Figure 3.7**). Interestingly, the *decoy variant* PSM strategy had similar distribution trends to that of the *decoy sequence* PSM strategy, despite having fewer spectra. Neither decoy class displayed harsh skewing toward higher p(-1) scores, indicating classification performance was not biased.

Furthermore, the resolution of the *variant* PSMs in the *decoy variant* strategy appeared to be better than the *decoy sequence* strategy, which offered more negative samples for classification without introducing performance bias. The proficiency of decision tree ensemble models depends on there being enough data points for a learning algorithm to efficiently sample and aggregate an accurate average (see Chapter 1.8). Therefore, including the *canonical* PSMs in the target class type for Task 2 may have aided the classification process of the Extra Trees architecture. This is especially noted given the 3-fold cross-validation in NT divided the training

samples into smaller subsets, and had only *variant* PSMs been used, the classification results may be unreliable.

The Extra Trees architecture operates by randomly selecting thresholds for each feature variable to create tree nodes with the best split (see Chapter 1.8.2.1). In Task 2, the proteogenomic pipeline included additional features (compared to Task 1), which combined with Extra Trees' inherent randomness, may have mitigated the effects of class imbalance. Ultimately, the distribution of spectra across the full p(-1) range provides reassurance in the decision-making process of Extra Trees. This observation may indirectly promote the feature harvesting process, and it would be interesting to rerun the iPSC classification without the additional features to compare their influence.

4.4.1.2 Performance evaluation of the decision tree architectures

The PSM dataset in Task 1 (see Chapter 2.8.1) consisted of 961,663 target spectra and 225,832 decoy spectra, meaning the ratio of decoy to target PSMs was roughly 1:4, respectively. The histograms in **Figure 3.3** displayed overall positive distribution trends for both target and decoy spectra across each of the decision tree ensemble architectures, despite unique outputs observed for the *Bagging* and *Boosting* model types. The *Boosting* models: Gradient Boosting, Histogram-based Gradient Boosting and XGBoost, produced distributions that were technically correct (**Figure 3.3[b, c, e]**), although a distinct divide in spectra was also observed. While this reflects an unbiased performance, it is questionable if there is an actual benefit to collecting target PSMs with a p(-1) of 0.0 as it may be viewed as overly optimistic and uninformative regarding annotation quality.

The ability to boost *weak learners* is a powerful technique; however, this strategy may not always be appropriate. *Boosting* models are known to be sensitive to noisy, complex data and tend to overfit in their effort to minimise errors. If the logic of the TDA is considered, the *Boosting* models (**Figure 3.3[b, c, e]**) appeared to deliberately collect random target PSMs to match the frequency of decoy PSMs across each architecture (**Table 3.2**). The question of whether *weak learners* and *strong learners* are equally comparable was postulated by Freund and Schapire (1997). It was concluded that it is possible, and their performance depends on the architecture's construction. If significant target PSMs are assumed to be correct, perhaps the idea of ranking PSMs for post-processing can be reconsidered when using *Boosting* models.

Random Forest or *Bagging* models appear to be well-suited for informative PSM processing, as viewed in the histograms from Task 1 (**Figure 3.3[a, c]**). Although Random Forest produced the least amount of significant target spectra (**Table 3.2**), this may be attributed to suboptimal

hyperparameter tuning. During the development of NT, Random Forest proved to be the most computationally expensive architecture, and certain default variables were producing unreliable spectra distributions that were seemingly biased. The hyperparameters were intentionally selected for Random Forest (**Table 2.4**) to increase randomness. However, they were not extensively tested as compared to Extra Trees, whose run time was the quickest and easiest to test more parameter variables. Therefore, Random Forest is not ruled out as a viable PSM classifier, and the benefit of increasing randomness for better spectra distribution is highlighted.

More importantly, the influence of hyperparameter tuning proves to be paramount to any classification assignment. In the research of Ulintz *et al*. (2006) it was argued that decision tree models do not need hyperparameter tuning due to the inherent robustness of their algorithms as compared to linear processes, such as SVMs. However, the work of Spivak *et al*. (2009) noted that ML architectures do not generalise well across different platforms and experimental conditions, necessitating parameter adjustment for new training samples. After reviewing NT, hyperparameter tuning was indeed necessary and altered the possible outcomes of this work, particularly for Random Forest. A reliable classification report would be incomplete without reviewing its influence on PSM processing.

If the goal is to rank spectra to infer significance, the strength of the probability test statistic matters. Take for example, a spectrum with a p(-1) of 0.01 from one classifier and a p(-1) of 0.15 in another. They are both significant values, although a threshold of p(-1) 0.1 would exclude the second probability as a highly confident spectrum. In this work, a threshold below a p(-1) 0.5 was set to collect acceptable spectra to mitigate this potential loss. However, the chance of random matches can be swayed by the learning potential of the model, especially if significant digits are considered. PSM datasets are not uniform in size, scoring function or quality; therefore, a classifier must process spectra optimally within these constraints. This has direct implications for the inference of error rate metrics, which rely on spectra ranking. Subsequently, this may explain the increase of random target spectra with a 1% FDR observed in the final classification of *Boosting* models (**Figure 3.6**), which exhibited a limited PSM distribution across the p(-1) range.

The AUROC (**Figure 3.1**) and PR curve (**Figure 3.2**) for each decision tree ensemble model demonstrated excellent distribution of spectra; however, the true behaviour of the target and decoy PSMs was revealed by the histograms and FDR analysis. While any ML architecture has the potential to function as a PSM processor, the data analytics of downstream proteogenomics does prove to require approaches that go beyond traditional ML techniques. Arguably, this is

less about the choice of algorithm and more about the application of thresholds and error rate metrics, which ultimately infer statistical significance beyond the baseline measure of probability tests.

4.4.2 Statistical inference

The statistical realm is filled with controversies, made so by statisticians debating the rationality of objective reasoning concerning frequentist tactics (Neyman-Pearson), against the subjective inference of conditional hypothesis testing (Mayo, 1997). Statistical philosopher Deborah Mayo eloquently rationalises these 'statistical wars' (Mayo, 2021) by prioritising the purpose of probability as a means to characterise data (Mayo, 1997). This is elaborated as a statistical inference determining the reliability of an alternative hypothesis by enumerating the errors, otherwise known as 'error probabilities', or 'error rates' in this work. Statistical inference analytics is not expected to obtain accurate findings and is instead aimed at conceding the closest approximation that has a frequent and strong likelihood of being successful. That is to say, the evaluation of a hypothesis metric is not a conclusive statement of whether it is good or bad, but rather a demonstration of a phenomenon, as stated by Sir Ronald A. Fisher (1955).

Computational techniques and the rise of high-throughput technologies allow for pattern detection in data, regardless of whether the algorithm is ideally suited to the particular process. Statistics for scientific research is regarded as a tool to process data into a workable product and therefore is generally not bound to the formal objections of statisticians (Mayo, 2000). However, statistical inference is subjective to the hypothesis at hand, and varying observations are expected when inferring analytics onto unassimilated data not included in the estimation process. Although the framework of NT attempts to improve the occurrence of false predictions by functioning as a self-trainer through cross-validation (see Chapter 1.6.2.2), it is acknowledged that its classification capabilities are nuanced to the PSM datasets used in this work. Results are vulnerable to unaccounted for characteristics, such as size, scoring functions and proportion of decoy PSMs, and may differ accordingly. As such, it is necessary to gauge the rationality of the statistical metrics.

The use of the TDA to infer confusion matrix metrics is a two-sided debate in the community. On one side, estimating a substantial proportion of false positives is seen as a way to endorse the reliability of significant true peptide-spectrum matches among target spectra (Elias and Gygi, 2007; Wang *et al*., 2009). However on the other side, it has been noted that not all spectral identification tasks are TDA compliant (Shen *et al.*, 2009; Gupta *et al.,* 2011). As previously mentioned, roughly 10 – 50% of peptides from search engines are identified downstream;

therefore, it is of interest to not further promote the loss of true positive spectra during PSM processing. While various aspects influence the occurrence of false positives, a key safeguard is to determine the reliability of processed PSMs. The statistical inference methods explored in this work, including the confusion matrix, FDR, PEP and p(-1) inference, are evaluated in this chapter.

4.4.2.1 Confusion matrix

In Task 1, the confusion matrix was implemented to estimate performance metrics of the AUROC and PR curves. Two other commonly used metrics for evaluating classical ML methods include the F1-score (Melo *et al.*, 2016) and the Matthews Correlation Coefficient (Chicco and Jurman, 2023). Including more metrics can provide a more comprehensive validation of a model's performance (Pang *et al.*, 2021). However, an opposing study by Caelen (2017) observed that each ML performance metric infers error rates indifferently and suggested that data points should be generalised using Bayesian tactics to properly adjust the confusion matrix. While the ROC and PR curves offered baseline insights into the distribution of classified spectra, classical ML methods are nuanced to PSM classification and alternative strategies specifically defined for this process are required. Whether these extended performance metrics would meaningfully enhance ML evaluation for PSM processing remains debatable, as it appears unnecessary. Nonetheless, they should not be disregarded as additional metrics can support statistical inference as needed.

4.4.2.2 False discovery rate

The FDR is not infallible and is heavily nuanced. The consensus is that the FDR tends to overfit (Tan and Xu, 2014) or display bias (Greenwald, 1975; Madej *et al.*, 2022) without careful control of the target and decoy proportions, which is also debated (Gupta *et al.*, 2011; Cooper, 2012). To curb these issues, various adaptations to the FDR have been proposed; however, there is no single agreed-upon method (Aggarwal and Yadav, 2016). Any formula type is acceptable as the reports of its effectiveness are situational to the research hypothesis, making their statistical inference subjective.

It is possible for a spectrum to receive a probability test statistic indicating a random match during post-processing, yet a flawed FDR model would acknowledge it as significant at a given threshold (Hernández *et al.*, 2014; Pascovici *et al.*, 2016). This is visualised in **Figure 3.6**, which depicts the p(-1) interval distribution of predicted random and non-random target spectra within a 1% FDR. The figure reflects the final classified PSM dataset and does not represent the

1% FDR target spectra selected for the training processes in NT. Nevertheless, this inference implies that random target PSMs were likely to be included in the training processes, despite the method's intent to optimally train the learning models.

While these are unattractive qualities in peptide identification tasks, the TDA for FDR estimation is still the *de facto* method in proteomic studies (Ebadi *et al.,* 2023), including this work. Despite its limitations, the application of the FDR is useful for isolating a generous proportion of significant spectra matches globally, as illustrated in **Figure 3.6** by the larger proportion of non-random target PSMs across all architectures. Although the p(-1) scores are a baseline measure; applying a second threshold in future studies could help refine confidence estimates. Alternatively, a local error rate estimation, such as the PEP, can be incorporated to improve the resolution of significant spectra used to train learning models.

4.4.2.3 Posterior error probability

Various strategies have been investigated to estimate PEP values through Bayesian inference for mass spectrometry spectra and PSMs from search engines. Zhang *et al*. (2008) explored the potential of PEP by proposing a Bayesian-based filtering model. Similarly, Alterovitz *et al*. (2007) reviewed the Bayesian framework in the context of likelihood estimation and parameter control for disease diagnostics and peptide identification. The opposing work of Serang *et al*. (2010), challenged the sensitivity of posterior probabilities, arguing they are insufficient for error rate inference in peptide-to-protein identification tasks. They highlighted that a protein may contain a single peptide with a significant PEP estimate while the remaining score poorly, undermining confidence in identification. The same study raised concerns regarding an increase in annotation errors when matching spectra to large datasets and ultimately discredited the use of post-processing tools.

In this work, a novel approach to PEP estimates was proposed, which entailed a 'container' method (see Chapter 2.8.2.5) to collect spectra within a 1% p(-1) range, offering an alternative to the Percolator method (see Chapter 1.7.1). In essence, this newer approach is a localised 'global' estimate. In the PEP distribution plots for Task 2 (**Figure 3.8**), each of the classified iPSC PSM datasets displayed an increasing PEP value beyond the p(-1) of 0.5, matching the degree of random matches (FPs and TNs) at this interval. Owing to this, the PEP 'container' method explored in this work may have a place amongst the existing Bayesian methods for peptide statistical inference.

It is expected that a small proportion of decoy spectra will receive probability scores indicating non-random matches. Additionally, larger datasets are more susceptible to the loss of non-random target spectra during error rate assessment. The 'container' PEP method proposed in this work considers the p(-1) estimates as a secondary threshold and optimistically counters this loss while producing reliable trends. Unlike the FDR, which is vulnerable to including false positives in its estimates (**Figure 2.5**), and requires scaling for imbalanced decoy spectra (see Chapter 1.5.3), the PEP relies solely on the classifier's power and the probability test statistic inferred onto the spectra.

4.4.2.4 Prediction probability p(-1)

The p-value metric is a household baseline test statistic that is derived from hypothesis testing as alluding to the Neyman-Pearson concept stipulated by Mayo (2019). This work employed the *prediction probability* p(-1) measurement, which was inferred from Scikit-learn (Pedregosa *et al*., 2011) and represented the probability of a spectrum being a random match. Although p-values and p(-1) measurements are obtained differently, they do share a similar degree of statistical inference in that both are continuous measurements of 0.0 – 1.0, indicating the degree of randomness (Hung *et al.*, 1997) (see Chapter 2.3.2). Additionally, neither are uniform estimates, meaning they are subject to change with estimation method and dataset size, for example.

The p-value, however, is under criticism from statisticians against its use as a measurement of evidence. This is due to its reliance on thresholds to infer statistical significance from a preliminary null hypothesis rejection region (H_0) (see Chapter 1.5.2), which are generally at 0.05 or less, and the assumption that the power of the probability metric is reliable and accurate (Mayo, 2019). Low p-value estimates representing significance are therefore judged as misguided, implying that real errors are underestimated for practical application and represent the dataset used for estimation only (Mayo and Cox, 2006; Andrade, 2019). Fortunately, the self-training classification procedure of NT has no intention to apply a trained learning algorithm to new PSM datasets, thus reassuring the use of the p(-1) values. Unlike the p-value, which uses thresholds to reject the null hypothesis, here, the decoy PSMs are the null model, and the p(-1) scores are an error metric in addition to the FDR and PEP. Thresholds are then inferred after classification.

The benefit of ML libraries from Scikit-learn is the decision probability threshold of 0.5 for predicting labels (Sweidan and Johansson, 2021), meaning that spectra with this score have a 50% chance of being a target or decoy spectra as estimated by the model (Mayo, 1980). Spectra

above this boundary are viewed as random matches due to the ambiguity inferred by the learning algorithm. Therefore, spectra with a p(-1) score below 0.5 can be considered of interest, such as in the total count analysis of **Figure 3.5**, which provides an inclusive comparison that observed Extra Trees as the better model. The ability to minimally recognise spectra with a p(-1) below 0.5 guards against the unnecessary loss of PSMs during processing, and the performance can be reviewed completely, such as for the iPSC PSM datasets viewed in **Figure 3.9**. The decision threshold can be adjusted using confusion matrix tactics through manual curation of ML architectures (Gao *et al.*, 2020). Fortunately, this is unnecessary for PSM processing, where the prioritisation of probability-based ranking takes precedence.

Typically, confidence intervals are used to compare estimates under varying conditions and provide a range of values with a strong likelihood of containing the true threshold for inferring significance (Neyman, 1941). Given that the threshold for acceptable spectra is quite wide at 0.5, a p(-1) threshold of 0.1 (10%) was arbitrarily stipulated in this work to infer statistical significance and monitor the outcome of the *counting function* for NT (**Figure 2.1**, Step 6). Perhaps a smaller threshold of 1% should have been selected to be on par with the FDR and PEP metrics; however, as the goal of NT was to maximise the number of significant spectra, a wider range was selected. The total counts of significant PSMs presented in **Figure 3.4** served as a starting point to observe the classification power of the architectures, from which the better model appeared to be XGBoost. Future statistical investigations could explore confidence interval assessment and assess whether it improves the *counting function* process for architecture performance comparison and PSM classification for protein inference.

Ultimately, where these probability test statistics and thresholds matter is spectra ranking and PSM-to-protein inference as the following step in the proteogenomic workflow. The usage of p(-1) and error metrics circle back to the arguments posed previously in Chapter 4.4.1 concerning the potential of ML applications. The choice of using decision tree ensemble models is additionally advantageous owing to the random sampling process to create strong learning algorithms and hyperparameter tuning for optimisation. In fairness, this may be viewed as overoptimistic from a statistician's perspective (Radzvilas *et al.*, 2021); however, the p(-1) values and statistical metrics employed in this work did serve their purpose as tools to process PSMs, and their potential should not be discouraged.

5: Conclusion

Peptide identification is a challenging endeavour, and the phases of the proteogenomic workflow are continuously being improved in their own right in testament to the potential of this field; however, none are infallible. Numerous strategies have been proposed to improve peptide identification, and here this work suggests another that hopes to find its place amongst the many. The discussion chapter discussed several concepts that were fundamental for this research project with concluding remarks regarding forethoughts for future work. Here, a few final thoughts are diarised to conclude the nature of this work.

Is Machine Learning an effective tool for post-processing PSMs?

The main observation from the application of ML techniques in this study is that there is no 'universal' architecture, and ultimately, the strategy invoked is dependent on the data and problem statement. This is to say, PSM datasets are not heterogeneous (Käll *et al*., 2007), and given the uncertainty in their quality, it remains unclear as to which ML architecture offers the appropriate robustness for the task of classification. This is particularly so for binary classification of target and decoy PSMs, which enables error rate estimation. While the TDA is a classic technique for null model distribution, the use of ML makes it difficult to comment on whether or not it is the most appropriate strategy. ML is a powerful tool, and paired with hyperparameter tuning, is purposefully designed to provide optimised predictions to the best of the learning algorithm's ability. Therefore, as long as a model can classify PSM datasets and produce probability statistics, it is considered successful (Spivak *et al.*, 2009).

In Task 1 of this work, 18 scoring functions (**Table 2.5**) were used to compare the prediction performance of five architectures, of which Extra Trees was selected; however, in Task 2 this was increased to 41 (**Table 2.7**). This increase in dimensionality (see Chapter 1.6, **Figure 1.12**) changes the landscape of PSM classification, and the comparison assignment of Task 1 is not an authentic report for canonical PSMs nor variant PSM analysis. It is acknowledged that the outcome of Extra Trees from Task 1 (although positive) is not transferable to Task 2, and its selection was based on a heuristic observation. The iPSC PSM datasets were significantly smaller than the one used in Task 1 (see Chapter 4.4) and class imbalance is a characteristic in target-decoy PSM analysis. Owing to this, Random Forest and Gradient Boosting could very well have been better options, and perhaps the comparison of ML architectures should focus on the decoy variant concept solely.

Additionally, separating the spectra into four individual class types for Task 2 opens the avenue to explore multi-class classification instead of the traditional binary classification of the TDA (Tanha *et al*., 2020). It was noted that combining the canonical PSMs with the variant PSMs for the target class types could have 'saved' the classification assignment. However, owing to the increased dimensions increasing model complexity, this creates the opportunity to explore the full potential of ML applications.

A high-dimensional learning environment and multi-class labelled data points justify the exploration of Deep Neural Networks for improved prediction quality (Wang *et al.*,2016). Percolator (Käll *et al*., 2007), which harnesses the linear approach of an SVM, is known to be well-suited for smaller datasets and performs effectively in high-dimensional classification. To comprehensively validate NT, a direct comparison to Percolator and other existing tools should be explored in future work for legitimate reporting. Decision tree models offer a middle ground in data analytics and have been proven to be a well-rounded learning style for various classification scenarios in this study. Future works could therefore compare prediction performance across various ML architecture styles to further validate the usage of decision trees for PSM classification.

Is Nagilums Tree an efficient system?

The workflow of NT is not inherently unique, given its self-training system; however, the *counting function* (**Figure 2.1**, Step 6) is an interesting dynamic that proposes an alternative to the traditional PSM classification ranking system. Ranking PSMs is essential for error rate estimation (Spivak *et al.*, 2009); however, as observed from the performance of *Boosting* architectures (see Chapter 4.4.1.2) there may be alternative strategies yet to be explored. Bypassing the ranking requirement opens the door for a filtering strategy instead, which can prioritise refining probability estimates and collecting quality PSMs.

Overall NT was able to complete the task of classifying PSMs as a self-training system, and as previously mentioned, it would be fair to compare its performance to that of existing tools, such as Percolator or PeptideProphet. However, NT was created as a system of scripts and was not intended to replace or improve existing methods, but rather to explore the potential of alternative concepts. Its free-flowing style allows for all the alternative options stated in this work to be easily implemented. The *counting function* for example, can be altered to incorporate PEP values instead of the prediction probability estimates by simply including its function in the script (see Chapter 2.8.2.5).

Are there practical application complications?

Whether or not the outcomes of this research are directly applicable to real-world genomic data, i.e. patient registries (see Chapter 1.3.4), remains to be proven. The iPSC data used in this work consisted of three cell lines, which is a marginal representation of any registry. Moreover, there are considerable differences between artificially generated experimental data and clinical data (Winkler and Murphy, 1973). Epigenetic modifications are erased and replaced with pluripotency patterns in PSCs, as stated by Efrat (2021). Therefore, although missing transcription factors do not impact gene expression of iPSCs, it does lead to incomplete findings of specific cells, such as pancreatic β-cells, which limits the practical application (Efrat, 2021). Additionally, theoretical databases are often presumed to be representative of varying populations, and as this is unlikely, these investigations are subject to missing or incomplete data (see Chapter 1.3.4). The complexities of patient clinical data may, in fact, require a processing structure that is both rigid and flexible enough to account for anomalies, such as variant peptides. The nature of this work paints a picture of this statement.

The purpose of this work was driven by the need to identify unknown peptide sequence variants of MODY genes. The work of Bogdanow *et al*. (2016) and Anders *et al*. (2021) attributed around 50% of false positive identifications to modified peptides, largely due to poor alignment during annotation. The classification structure of NT (**Figure 2.1**) prioritised obtaining statistically significant true positive spectra with a high likelihood of being correctly identified and imposed error rates and thresholds to achieve the task. In retrospect, it is possible that the statistical direction taken in this work was misguided. The work of Savitski *et al*. (2015) proposes the 'picked' FDR approach, which tracks the decoy counterparts of target spectra and compares their probability scores for significance. Anders *et al*. (2021) likewise used FDR estimates to prove that unannotated peptides, likely due to variation, occur as false positive PSMs. The same study reviewed the benefit of 'aggregating statistics' to produce confidence scores for protein identification and evaluated candidate proteins to determine variant proteins.

Although the ML techniques and PEP 'container' method in this work accomplished aggregated statistics, the downstream task of PSM-to-protein inference was not conducted. Furthermore, the decoy variant concept task did not evaluate unknown variant peptides to the depth of the aforementioned studies. Considering this, future works should place more emphasis on target-labelled false positives in tandem with exploring alternate statistical and classification strategies to complete the objective of this work. At this juncture, it could remain that the concepts of this work are applicable to experimental research only.

Final sentiment

This study ventured to explore the challenges of proteogenomics while facing some unique trials, and in the end, was able to return an exhaustive report. Ultimately, the final take-home message is that processing PSMs is fundamentally a statistical problem. However, statistical inference is heavily nuanced in itself, more so than peptide identification, which is susceptible to false positives entering the annotation space from varying sources. Serang and Käll (2015) stated that the key to improving peptide classification is to incorporate more statistical measures. Upon reviewing the results from this work, however, it is questioned whether it is misguided to strive towards highly accurate error rate estimations.

This conundrum is echoed by the thoughtful mind of Dyson (2004), who recalled a meeting with the esteemed Enrico Fermi to critique certain results in theoretical physics. Fermi prescribed that formulae should either be precise and 'self-consistent' or rather define a system with a 'clear physical picture' and remarked that the overuse of thresholds to perfect a process is uncharacteristic of nature. This idea is foundational for any statistical practice. As the mathematician Good (1988) mentions, formulae encapsulate an idea and set a goal that is beneficial towards the application of statistics. It is with this mindset that the exploratory analysis of this study, alongside those of likeness, hopes to contribute to the foundation of proteogenomic research for precision medicine and help pave the way toward meaningful discoveries in this field.

6: References

Adeyemo, A., Gerry, N., Chen, G., Herbert, A., Doumatey, A., Huang, H., Zhou, J., Lashley, K., Chen, Y., Christman, M. & Rotimi, C. 2009. A genome-wide association study of hypertension and blood pressure in African Americans. *PLoS Genetics*. 5(7):e1000564.

Aggarwal, S. & Yadav, A.K. 2016. False discovery rate estimation in proteomics, in *Methods in Molecular Biology*, vol. 1362, Springer. 119–128.

Ahmad, A., Safi, O., Malebary, S., Alesawi, S. & Alkayal, E. 2021. Decision tree ensembles to predict coronavirus disease 2019 infection: a comparative study. *Complexity*. 2021:550344.

Al-Azzam, N. & Shatnawi, I. 2021. Comparing supervised and semi-supervised machine learning models on diagnosing breast cancer. *Annals of Medicine and Surgery*. 62:53–64.

Ali, Y.A., Mahrous Awwad, E., Al-Razgan, M. & Maarouf, A. 2023. Hyperparameter search for machine learning algorithms for optimizing the computational complexity. *Processes*. 11(2):349.

Alizadeh, H., Yousefnezhad, M. & Bidgoli, B.M. 2015. Wisdom of crowds cluster ensemble. *Intelligent Data Analysis*. 19(3):485–503.

Alterovitz, G., Liu, J., Afkhami, E. & Ramoni, M.F. 2007. Bayesian methods for proteomics. *Proteomics*. 7(16):2843–2855.

America, A.H.P., Cordewener, J.H.G., Van Geffen, M.H.A., Lommen, A., Vissers, J.P.C., Bino, R.J. & Hall, R.D. 2006. Alignment and statistical difference analysis of complex peptide data sets generated by multidimensional LC-MS. *Proteomics*. 6(2):641–653.

Anders, J., Petruschke, H., Jehmlich, N., Haange, S.B., von Bergen, M. & Stadler, P.F. 2021. A workflow to identify novel proteins based on the direct mapping of peptide-spectrum-matches to genomic locations. *BMC Bioinformatics*. 22:277.

Anderson, D.C., Li, W., Payan, D.G. & Noble, W.S. 2003. A new algorithm for the evaluation of shotgun peptide sequencing in proteomics: support vector machine classification of peptide MS/MS spectra and SEQUEST scores. *Journal of Proteome Research*. 2(2):137–146.

Anderson, D.R., Burnham, K.P. & Thompson, W.L. 2000. Null hypothesis testing: problems, prevalence, and an alternative. *The Journal of Wildlife Management*. 64(4):912–923.

Andrade, C. 2019. The P Value and statistical significance: misunderstandings, explanations, challenges, and alternatives. *Indian Journal of Psychological Medicine*. 41(3):210–215.

Armstrong, R.A. 2014. When to use the Bonferroni correction. *Ophthalmic and Physiological Optics*. 34(5):502–508.

Arnold, R.J., Jayasankar, N., Aggarwal, D., Tang, H. & Radivojac, P. 2006. A machine learning approach to predicting peptide fragmentation spectra. *Biocomputing*. 219–230.

Aukrust, I. *et al*. 2013. SUMOylation of pancreatic glucokinase regulates its cellular stability and activity. *Journal of Biological Chemistry*. 288(8):5951–5962.

Bai, Z. & Saranadasa, H. 1996. Effect of high dimension: by an example of a two sample problem. *Statistica Sinica*. 6(2):311–329.

Bai, Y., Chen, M., Zhou, P., Zhao, T., Lee, J.D., Kakade, S., Wang, H. & Xiong, C. 2021. How important is the train-validation split in meta-learning?, *Proceedings of Machine Learning Research*. 139:543–553.

Bainbridge, M.N. 2020. Determining the incidence of rare diseases. *Human Genetics*. 139(5):569–574.

Baker, M. 2012. Structural variation: the genome's hidden architecture. *Nature Methods*. 9(2):133–137.

Banting, F.G., Best, C.H., Collip, J.B., Macleod, J.J.R. & Noble, E.C. 1922. the effects of insulin on experimental hyperglycemia in rabbits. *American Journal of Physiology*. 62:559–580.

Barbieri, R., Guryev, V., Brandsma, C.A., Suits, F., Bischoff, R. & Horvatovich, P. 2016. Proteogenomics: key driver for clinical discovery and personalized medicine, in *Advances in Experimental Medicine and Biology*, vol. 926, Springer. 21–47.

Barg, S. 2003. Mechanisms of exocytosis in insulin-secreting B-cells and glucagon-secreting A-cells. *Pharmacology and Toxicology*. 92:3–13.

Bateman, A. *et al.* 2024. UniProt: the Universal Protein Knowledgebase in 2025. *Nucleic Acids Research*. 53(D1):D609-D617

Bayes, T. 2003. An essay towards solving a problem in the doctrine of chances. *Resonance*. 80-88

Bayes, T. & Price, R. 1763. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. *Philosophical Transactions of the Royal Society of London*. 53(53):370–418.

Beltrand, J., Busiah, K., Vaivre-Douret, L., Fauret, A.L., Berdugo, M., Cavé, H. & Polak, M. 2020. Neonatal diabetes mellitus. *Frontiers in Pediatrics*. 8.

Benjamini, Y. & Hochberg, Y. 1996. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*. 57:289–300.

Bern, M.W. & Kil, Y.J. 2011. Two-dimensional target decoy strategy for shotgun proteomics. *Journal of Proteome Research*. 10(12):5296–5301.

Berrar, D. 2018. Cross-validation, in *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, vol. 1–3, Elsevier. 542–545.

Biau, D.J., Jolles, B.M. & Porcher, R. 2010. P Value and the theory of hypothesis testing an explanation for new researchers. *Clinical Orthopaedics and Related Research*. 468:885–892.

Blanco, J.L, Porto-Pazos, A.B., Pazos, A. & Fernandez-Lozano, C. 2018. Prediction of high anti-angiogenic activity peptides in silico using a generalized linear model and feature selection. *Scientific Reports*. 8:15688.

Bogdanow, B., Zauber, H. & Selbach, M. 2016. Systematic errors in peptide and protein identification and quantification by modified peptides. *Molecular & Cellular Proteomics*. 15(8):2791–2801.

Bouwmeester, R., Gabriels, R., Hulstaert, N., Martens, L. & Degroeve, S. 2021. DeepLC can predict retention times for peptides that carry as-yet unseen modifications. *Nature Methods* . 18:1363–1369.

Bowman, P. *et al.* 2018. Effectiveness and safety of long-term treatment with sulfonylureas in patients with neonatal diabetes due to KCNJ11 mutations: an international cohort study. *The Lancet Diabetes and Endocrinology*. 6(8):637–646.

Breiman, L. 1996. Bagging predictors. *Machine Learning*. 24:123–140.

Breiman, L. 2001. Random forests. *Machine Learning*. 45:5–32.

Broome, D.T., Pantalone, K.M., Kashyap, S.R. & Philipson, L.H. 2021. Approach to the patient with MODY-monogenic diabetes. *Journal of Clinical Endocrinology and Metabolism*. 106(1):237–250.

van de Bunt, M., Lako, M., Barrett, A., Gloyn, A.L., Hansson, M., McCarthy, M.I., Beer, N.L. & Honoré, C. 2016. Insights into islet development and biology through characterization of a human iPSC-derived endocrine pancreas model. *Islets*. 8(3):83–95.

Caelen, O. 2017. A Bayesian interpretation of the confusion matrix. *Annals of Mathematics and Artificial Intelligence*. 81:429–450.

Cammidge, P.J. 1928. Diabetes mellitus and hereditary. *British Medical Journal*. (3538):738–741.

Cannon, W.R. *et al.* 2005. Comparison of probability and likelihood models for peptide identification from tandem mass spectrometry data. *Journal of Proteome Research*. 4(5):1687–1698.

Cargile, B.J., Bundy, J.L. & Stephenson, J.L. 2004. Potential for false positive identifications from large databases through tandem mass spectrometry. *Journal of Proteome Research*. 3(5):1082–1085.

Chang, J. *et al.* 2018. A rare missense variant in TCF7L2 associates with colorectal cancer risk by interacting with a GWAS-identified regulatory variant in the MYC enhancer. *Cancer Research*. 78(17):5164–5172.

Chen, T. & Guestrin, C. 2016. XGBoost: a scalable tree boosting system, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 785–794.

Chen, J. *et al.* 2019. Genome-wide association study of type 2 diabetes in Africa. *Diabetologia*. 62(7):1204–1211.

Chen, Y., Kwon, S.W., Kim, S.C. & Zhao, Y. 2005. Integrated approach for manual evaluation of peptides identified by searching protein sequence databases with tandem mass spectra. *Journal of Proteome Research*. 4(3):998–1005.

Chicco, D. & Jurman, G. 2023. The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification. *BioDataMining*. 16:4.

Chick, J.M., Kolippakkam, D., Nusinow, D.P., Zhai, B., Rad, R., Huttlin, E.L. & Gygi, S.P. 2015. A mass-tolerant database search identifies a large proportion of unassigned spectra in shotgun proteomics as modified peptides. *Nature Biotechnology*. 33(7):743–749.

Choi, H. & Nesvizhskii, A.I. 2008. Semisupervised model-based validation of peptide identifications in mass spectrometry-based proteomics. *Journal of Proteome Research*. 7(1):254–265.

Choong, W.K. & Sung, T.Y. 2021. Comparison of different variant sequence types coupled with decoy generation methods used in concatenated target-decoy database searches for proteogenomic research. *Journal of Proteomics*. 231:104021.

Choudhury, A. *et al.* 2014. Population-specific common SNPs reflect demographic histories and highlight regions of genomic plasticity with functional relevance. *BMC Genomics*. 15:437.

Clarke, R., Ressom, H.W., Wang, A., Xuan, J., Liu, M.C., Gehan, E.A. & Wang, Y. 2008. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nature Reviews Cancer*. 8:37–49.

Cole, T.J. 2015. Too many digits: The presentation of numerical data. *Archives of Disease in Childhood*. 100(7):608–609.

Colinge, J., Masselot, A., Giron, M., Dessingy, T. & Magnin, J. 2003. OLAV: Towards high-throughput tandem mass spectrometry data identification. *Proteomics*. 3(8):1454–1463.

Cooper, B. 2012. The problem with peptide presumption and the downfall of target– decoy false discovery rates. *Analytical Chemistry*. 84:9663–9667.

Cortes, C., Mohri, M. & Storcheus, D. 2019. Regularized Gradient Boosting, *Advances in Neural Information Processing Systems.* 32

Craig, R. & Beavis, R.C. 2004. TANDEM: matching proteins with tandem mass spectra. *Bioinformatics*. 20(9):1466–1467.

Crick, F.H.C. 1954. The structure of the hereditary material. *Scientific American*. 191(4):54–61.

Cujba, A. *et al.* 2022. An HNF1a truncation associated with maturity-onset diabetes of the young impairs pancreatic progenitor differentiation by antagonizing HNF1b function. *Cell Reports*. 38(9):110425.

Davis, J. & Goadrich, M. 2006. The relationship between Precision-Recall and ROC curves, *Proceedings of the 23rd International Conference on Machine Learning*. 233–240.

Degroeve, S., Martens, L. & Jurisica, I. 2013. MS2PIP: a tool for MS/MS peak intensity prediction. *Bioinformatics*. 29(24):3199–3203.

Deng, H. & Runger, G. 2012. Feature selection via regularized trees. *The 2012 International Joint Conference on Neural Networks*. 1–8.

Deutsch, E.W., Lam, H. & Aebersold, R. 2008. Data analysis and bioinformatics tools for tandem mass spectrometry in proteomics. *Physiological Genomics*. 33:18–25.

Dietterich, T.G. 2000. Ensemble Methods in Machine Learning. *International workshop on multiple classifier systems*. 1857:1-15

Dincer, A.B., Lu, Y., Schweppe, D.K., Oh, S. & Noble, W.S. 2022. Reducing peptide sequence bias in quantitative mass spectrometry data with machine learning. *Journal of Proteome Research*. 21(7):1771–1782.

Dunn, O, J. 1961. Multiple comparisons among means. *Journal of the American Statistical Association*. 56:52–64.

Dyson, F. 2004. A meeting with Enrico Fermi. *Nature*. 427:297.

Ebadi, A., Freestone, J., Noble, W.S. & Keich, U. 2023. Bridging the false discovery gap. *Journal of Proteome Research*. 22(7):2172–2178.

Eddes, J.S., Kapp, E.A., Frecklington, D.F., Connolly, L.M., Layton, M.J., Moritz, R.L. & Simpson, R.J. 2002. CHOMPER: A bioinformatic tool for rapid validation of tandem mass spectrometry search results associated with high-throughput proteomic strategies, *Proteomics*, 2(9):1097–1103.

Efrat, S. 2021. Epigenetic memory: lessons from iPS cells derived from human β cells. *Frontiers in Endocrinology*. 11:614234

Efron, B. & Tibshirani, R.J. 1993. An introduction to the bootstrap. Chapman & Hall.

Efron, B., Tibshirani, R., Storey, J.D. & Tusher, V. 2001. Empirical Bayes analysis of a microarray experiment. *Journal of the American Statistical Association*. 96(456):1151–1160.

Elias, J.E. & Gygi, S.P. 2007. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nature Methods*. 4(3):207–214.

Elias, J.E. & Gygi, S.P. 2010. Target-decoy search strategy for mass spectrometry-based proteomics. *Methods in molecular biology*. 604:55–71.

Elliott, D.L. & Anderson, C. 2023. The wisdom of the crowd: reliable deep reinforcement learning through ensembles of Q-Functions. *IEEE Transactions on Neural Networks and Learning Systems*. 34:43–51.

Eng, J.K., Mccormack, A.L. & Yates, J.R. 1994. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*. 5(11):976–989.

Ezkurdia, I., Vázquez, J., Valencia, A. & Tress, M. 2014. Analyzing the first drafts of the human proteome. *Journal of Proteome Research*. 13(8):3854–3855.

Fajans, S.S. & Bell, G.I. 2011. MODY: History, genetics, pathophysiology, and clinical decision making. *Diabetes Care*. 34(8):1878–1884.

Fajans, S.S. & Conn, J.W. 1954. An approach to the prediction of diabetes mellitus by modification of the glucose tolerance test with cortisone. *Diabetes*. 3(4):296–304.

Fajans, S.S. & Conn, J.W. 1958. The early recognition of diabetes mellitus. *Annals of the New York Academy of Sciences*. 82:208–218.

Fendler, W. *et al.* 2012. Prevalence of monogenic diabetes amongst Polish children after a nationwide genetic screening campaign. *Diabetologia*. 55(10):2631–2635.

Ferkingstad, E. *et al*. 2021. Large-scale integration of the plasma proteome with genetics and disease. *Nature Genetics*. 53(12):1712–1721.

Fisher, R. 1955. Statistical methods and scientific induction. *Journal of the Royal Statistical Society: Series B (Methodological)*. 17(1):69–78.

Fisher, R.A. 1925. Statistical methods for research workers. *Edinburgh, UK: Oliver and Boyd*.

Frank, A.M. 2009. A Ranking-Based scoring function for peptide-spectrum matches. *Journal of Proteome Research*. 8(5):2241–2252.

Freund, Y. & Schapire, R.E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*. 55(1):119-139.

Friedman, J.H. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*. 29(5):1189–1232.

Gabbay, D., Thagard, P. & Woods, J. 2010. Handbook of the philosophy of science. *Philosophy of Statistics*. vol. 7. Bandyopadhyay Prasanta S & Forster Malcolm R (eds.). Elsevier.

Galton F. 1907. Vox Populi. *Nature*. 75.

Gao, S., Dong, W., Cheng, K., Yang, Xibei, Shang Zheng, & Yu, H. 2020. Adaptive decision threshold-based extreme learning machine for classifying imbalanced multi-label data. *Neural Processing Letters*. 52:2151–2173.

Garin, I. *et al*. 2008. Haploinsufficiency at GCK gene is not a frequent event in MODY2 patients. *Clinical Endocrinology*. 68(6):873–878.

Geisser, S. 1975. The predictive sample reuse method with applications. *Journal of the American Statistical Association*. 70(350):320–328.

Geurts, P., Fillet, M., De Seny, D., Meuwis, M., Malaise, M., Merville, M. & Wehenkel, L. 2005. Proteomic mass spectra classification using decision tree based ensemble methods. *Bioinformatics*. 21(15):3138–3145.

Ghaddar, B. & Naoum-Sawaya, J. 2017. High dimensional data classification and feature selection using support vector machines. *European Journal of Operational Research*. 265(3):993–1004.

Gonnelli, G., Stock, M., Verwaeren, J., Maddelein, D., De Baets, B., Martens, L. & Degroeve, S. 2015. A decoy-free approach to the identification of peptides. *Journal of Proteome Research*. 14(4):1792–1798.

Good, I.J. 1988. The interface between statistics and philosophy of science. *Statistical Science*. 3:386–412.

Gorriz, J.M., Segovia, F., Ramirez, J., Ortiz, A. & Suckling, J. 2024. Is K-fold cross validation the best model selection method for Machine Learning? *arXiv*. 2401.16407.

Granholm, V. & Käll, L. 2011. Quality assessments of peptide-spectrum matches in shotgun proteomics. *Proteomics*. 11(6):1086–1093.

Granholm, V., Noble, W.S. & Käll, L. 2012. A cross-validation scheme for Machine Learning algorithms in shotgun proteomics. *BMC bioinformatics*. 13(Suppl 16):S3.

Grattan-Guinness, I. 2005. Landmark Writings in Western Mathematics 1640-1940. 1st ed. Cooke Roger, Corry Leo, Crepel Pierre, & Guicciardini Niccolo (eds.). Netherlands: Elsevier.

Greeley, S.A.W. *et al*. 2022. ISPAD Clinical Practice Consensus Guidelines 2022: The diagnosis and management of monogenic diabetes in children and adolescents. *Pediatric Diabetes*. 23(8):1188–1211.

Green, D.M. 1970. Application of detection theory in psychophysics. *Proceedings of the IEEE*. 58(5):713–723.

Green, P.E. 1970. Scanning the issue. Special. issue on detection theory *Proceedings of the IEEE*. 58(5):607–609.

Greenwald, A.G. 1975. Consequences of Prejudice against the null hypothesis. *Psychological Bulletin*. 82(1):1–20.

Gromada, J., Chabosseau, P. & Rutter, G.A. 2018. The α-cell in diabetes mellitus. *Nature Reviews Endocrinology*. 14(12):694–704.

Gronau, Q.F. & Wagenmakers, E.J. 2019. Limitations of Bayesian leave-one-out cross-validation for model selection. *Computational Brain and Behavior*. 2:1–11.

Gudbjartsson, D.F. *et al*. 2015. Large-scale whole-genome sequencing of the Icelandic population. *Nature Genetics*. 47(5):435–444.

Guerts, P., Ernst, D. & Wehenkel, L. 2006. Extremely randomized trees. *Machine Learning*. 63:3–42.

Guolin, K., Qi, M., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. vol. 30. Guyon I, Von Luxburg U, Benigo S, Wallach H, Fergus R, Vishwanathan S, & Garnett R (eds.). *Curran Associates Inc*.

Gupta, N., Bandeira, N., Keich, U. & Pevzner, P.A. 2011. Target-decoy approach and false discovery rate: When things may go wrong. *Journal of the American Society for Mass Spectrometry*. 22(7):1111–1120.

Habib, S.L., Prihoda, T.J., Luna, M. & Werner, S.A. 2012. Diabetes and risk of renal cell carcinoma. *Journal of Cancer*. 3:42–48.

Halloran, J.T. *et al*. 2019. Speeding up Percolator. *Journal of Proteome Research*. 18(9):3353–3359.

Handler, D.C.L. & Haynes, P.A. 2021. PeptideMind — Applying machine learning algorithms to assess replicate quality in shotgun proteomic data. *SoftwareX*. 13:100644.

Harries, L.W., Brown, J.E. & Gloyn, A.L. 2009. Species-specific differences in the expression of the HNF1A, HNF1B and HNF4A genes. *PLoS ONE*. 4(11):e7855.

Harrison, P.W. *et al*. 2024. Ensembl 2024. *Nucleic Acid*. 52(D1):D891–D899.

Hastie, T., Tibshirani, R. & Friedman, J. 2008. The Elements of Statistical Learning. Second ed. Springer.

He, K., Fu, Y., Zeng, W., Luo, L., Chi, H., Liu, C., Qing, L., Sun, R. & He, S. (2015). A theoretical foundation of the target-decoy search strategy for false discovery rate control in proteomics. *arXiv*. 1501.00537.

Hernández, B., Parnell, A. & Pennington, S.R. 2014. Why have so few proteomic biomarkers "survived" validation? (Sample size and independent validation considerations). *Proteomics*. 14(13-14):1587–1592.

Hossain, S.M.M. & Deb, K. 2021. Plant leaf disease recognition using histogram based gradient boosting classifier, Vasant P, Zelinka I, & Weber G (eds.). *Intelligent Computing and Optimization. ICO 2020. Advances in Intelligent Systems and Computing*, Springer. 530–545.

Huang, J. & Ling, C.X. 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*. 17(3):299–310.

Hung, H.M.J., O'Neill, R.T., Bauer, P. & Köhne, K. 1997. The behavior of the P-Value when the alternative hypothesis is true. *Biometrics*. 53:11–22.

Huynh, T., Nibali, A. & He, Z. 2021. Semi-supervised learning for medical image classification using imbalanced training data. *Computer Methods and Programs in Biomedicine*. 216:106628.

IDF. 2021. IDF Diabetes Atlas 10th edition. 10th ed. Brussels: International Diabetes Federation.

Irgens, H.U. *et al*. 2013. Prevalence of monogenic diabetes in the population-based Norwegian childhood diabetes registry. *Diabetologia*. 56(7):1512–1519.

Isaksson, A., Wallman, M., Göransson, H. & Gustafsson, M.G. 2008. Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recognition Letters*. 29(14):1960–1965.

Jackson, M., Marks, L., May, G.H.W. & Wilson, J.B. 2018. The genetic basis of disease. *Essays in Biochemistry*. 62(5):643–723.

Joshi, M. V, Agarwal, R.C. & Kumar, V. 2002. Predicting rare classes: can boosting make any weak learner strong? *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, USA: *Association for Computing Machinery*. 297–306.

Jouanna, J. 2012. Greek Medicine from Hippocrates to Galen. vol. 40. Scarborough John, van der Eijk Philip J, Hanson Ann Ellis, & Ziegler Joseph (eds.). Brill.

Juarez-Orozco, L.E., Martinez-Manzanera, O., Nesterov, S. V., Kajander, S. & Knuuti, J. 2018. The machine learning horizon in cardiac hybrid imaging. *European Journal of Hybrid Imaging*. 2:15.

Käll, L., Canterbury, J.D., Weston, J., Noble, W.S. & MacCoss, M.J. 2007. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature Methods*. 4(11):923–925.

Käll, L., Storey, J.D., MacCoss, M.J. & Noble, W.S. 2008. Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. *Journal of Proteome Research*. 7(1):29-34

Käll, L., Storey, J.D., MacCoss, M.J. & Noble, W.S. 2008. Posterior error probabilities and false discovery rates: Two sides of the same coin. *Journal of Proteome Research*. 7(1):40-44

Käll, L., Storey, J.D. & Noble, W.S. 2008. Non-parametric estimation of posterior error probabilities associated with peptides identified by tandem mass spectrometry. *Bioinformatics*. 24(16):i42–i48.

Kam Ho, T. 1995. Random decision forests, P*roceedings of 3rd International Conference on Document Analysis and Recognition*. 278–282.

Kavitha, B., Ranganathan, S., Gopi, S., Vetrivel, U., Hemavathy, N., Mohan, V. & Radha, V. 2023. Molecular characterization and re-interpretation of HNF1A variants identified in Indian MODY subjects towards precision medicine. *Frontiers in Endocrinology*. 14.

Kearns, M. 1994. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the Association for Computing Machinery*. 41(1):67–95.

Keller, A., Nesvizhskii, A.I., Kolker, E. & Aebersold, R. 2002. Empirical statistical model to estimate the accuracy of peptide identifications made by MS/MS and database search. *Analytical Chemistry*. 74(20):5383–5392.

Kim, H., Lee, S. & Park, H. 2019. Target-small decoy search strategy for false discovery rate estimation. *BMC Bioinformatics*. 20:438.

Kim, M.S., Jo, D.S. & Lee, D.Y. 2019. Comparison of HbA1c and OGTT for the diagnosis of type 2 diabetes in children at risk of diabetes. *Pediatrics and Neonatology*. 60(4):428–434.

Kneale, W. 1948. Boole and the revival of logic. *Mind*. 57(226):149–175.

Kohavi, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection, *Proceedings of the 14th International Joint Conference on Artificial Intelligence.* vol 2, Morgan Kaufmann Publishers. 1137–1143.

Krawczyk, B., Wozniak, M. & Schaefer, G. 2014. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*. 14(Part C):554–562.

Krentz, A.J., Clough, G. & Byrne, C.D. 2007. Interactions between microvascular and macrovascular disease in diabetes: Pathophysiology and therapeutic implications. *Diabetes, Obesity and Metabolism*. 9(6):781–791.

Kurland, L.T. & Molgaard, C.A. 1981. The patient record in epidemiology. *Scientific American*. 245(4):54–63.

Kwon, T., Choi, H., Vogel, C., Nesvizhskii, A.I. & Marcotte, E.M. 2011. MSblender: A probabilistic approach for integrating peptide identifications from multiple database search engines. *Journal of Proteome Research*. 10(7):2949–2958.

Laddach, A., Ng, J.C.F., Chung, S.S. & Fraternali, F. 2018. Genetic variants and protein–protein interactions: a multidimensional network-centric view. *Current Opinion in Structural Biology*. 50:82–90.

Lam, H. 2011. Building and searching tandem mass spectral libraries for peptide identification. *Molecular and Cellular Proteomics*. 10(12):R111.008565.

Lam, H., Deutsch, E.W., Eddes, J.S., Eng, J.K., King, N., Stein, S.E. & Aebersold, R. 2007. Development and validation of a spectral library searching method for peptide identification from MS/MS. *Proteomics*. 7(5):655–667.

LaPlace, P.S. de. 1814. Essai philosophique sur les probabilités. Mathematica. Paris: Courcier.

Levitsky, L.I., Ivanov, M. V, Lobas, A.A. & Gorshkov, M. V. 2017. Unbiased false discovery rate estimation for shotgun proteomics based on the target-decoy approach. *Journal of Proteome Research*. 16(2):393–397.

Li, H., Park, J., Kim, H., Hwang, K. & Paek, E. 2017. Systematic comparison of false-discovery-rate-controlling strategies for proteogenomic search using spike-in experiments. *Journal of Proteome Research*. 16(6):2231–2239.

Li, J., Su, Z., Ma, Z.Q., Slebos, R.J.C., Halvey, P., Tabb, D.L., Liebler, D.C., Pao, W. & Zhang, B. (2011). A bioinformatics workflow for variant peptide detection in shotgun proteomics. *Molecular and Cellular Proteomics*. 10(5):M110.006536.

Lin, A., Short, T., Noble, W.S. & Keich, U. 2022. Improving peptide-level mass spectrometry analysis via double competition. *Journal of Proteome Research*. 21(10):2412–2420.

Lin, A., See, D., Fondrie, W.E., Keich, U. & Noble, W.S. 2024. Target-decoy false discovery rate estimation using Crema. *Proteomics*. 24(8):e2300084.

Lin, W., Wang, J., Zhang, W.J. & Wu, F.X. 2012. An unsupervised machine learning method for assessing quality of tandem mass spectra. *Proteome Science*. 10(Suppl 1):S12.

Ling, C.X., Huang, J. & Zhang, H. 2003. AUC: a better measure than accuracy in comparing learning algorithms, Xiang Y & Chaib-draa B (eds.). *Advances in Artificial Intelligence*, vol. 2671, Canada: Springer. 329–341.

Liu, Z. & Bondell, H.D. 2019. Binormal Precision-Recall curves for optimal classification of imbalanced data. *Statistics in Biosciences*. 11:141–161.

Liu, J., Bell, A.W., Bergeron, J.J.M., Yanofsky, C.M., Carrillo, B., Beaudrie, C.E.H. & Kearney, R.E. 2007. Methods for peptide identification by spectral comparison. *Proteome Science*. 5:3.

Ló pez-Ferrer, D. *et al*. 2004. Statistical model for large-scale peptide identification in databases from tandem mass spectra using SEQUEST. *Analytical Chemistry*. 76(23):6853–6860.

Lu, Y. & Westfall, P.H. 2009. Is Bonferroni admissible for large m? *American Journal of Mathematical and Management Sciences*. 29(1–2):51–69.

Luque, A., Carrasco, A., Martín, A. & De Las Heras, A. 2019. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*. 91:216–231.

Lusted, L.B. 1971. Decision-making studies in patient management. *The New England Journal of Medicine*. 284(8):416–424.

Lycette, B.E., Glickman, J.W., Roth, S.J., Cram, A.E., Kim, T. H., Krizanc, D. & Weir, M.P. 2016. N-Terminal peptide detection with optimized peptide-spectrum matching and streamlined sequence libraries. *Journal of Proteome Research*. 15(9):2891–2899.

Ma, Z.Q. *et al*. 2009. IDPicker 2.0: Improved protein assembly with high discrimination peptide identification filtering. *Journal of Proteome Research*. 8(8):3872–3881.

Madej, D., Wu, L. & Lam, H. 2022. Common decoy distributions simplify false discovery rate estimation in shotgun proteomics. *Journal of Proteome Research*. 21(2):339–348.

Malikova, J. *et al*. 2020. Functional analyses of HNF1A-MODY variants refine the interpretation of identified sequence variants. *Journal of Clinical Endocrinology and Metabolism*. 105(4):e1377-e1386.

Manavalan, B., Subramaniyam, S., Hwan Shin, T., Ok Kim, M. & Lee, G. 2018. Machine-learning-based prediction of cell-penetrating peptides and their uptake efficiency with improved accuracy. *Journal of Proteome Research*. 17(8):2715–2726.

Manolio, T.A. *et al*. 2009. Finding the missing heritability of complex diseases. *Nature*. 461:747–753.

Marcot, B.G. & Hanea, A.M. 2021. What is an optimal value of k in k-fold cross-validation in discrete Bayesian network analysis? *Computational Statistics*. 36(3):2009–2031.

Matsha, T.E., Raghubeer, S., Tshivhase, A.M., Davids, S.F.G., Hon, G.M., Bjørkhaug, L. & Erasmus, R.T. 2020. Incidence of HNF1A and GCK MODY variants in a South African population. *Application of Clinical Genetics*. 13:209–219.

Mayo, D.G. 1980. The philosophical relevance of statistics. *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*. 1980:97–109.

Mayo, D.G. 1997. Error statistics and learning from error: Making a virtue of necessity. *Philosophy of Science*. 64:195–212.

Mayo, D.G. 2000. Experimental practice and an error statistical account of evidence. *Philosophy of Science*. 67(S3):S193–S207.

Mayo, D.G. 2019. P-value thresholds: Forfeit at your peril. *European Journal of Clinical Investigation*. 49:10.

Mayo, D.G. 2021. The statistics wars and intellectual conflicts of interest. *Conservation Biology*. 36:e13861.

Mayo, D.G. & Cox, D.R. 2006. Frequentist statistics as a theory of inductive inference. *Optimality: The Second Erich L. Lehmann Symposium*. 49:77–97.

Mayo, D.G. & David, H. 2022. Statistical significance and its critics: practicing damaging science, or damaging scientific practice? *Synthese*. 200:220.

Melo, R. *et al*. 2016. A machine learning approach for hot-spot detection at protein-protein interfaces. *International Journal of Molecular Sciences*. 17(8):1215.

Millikin, R.J., Shortreed, M.R., Scalf, M. & Smith, L.M. 2020. A Bayesian null interval hypothesis test controls false discovery rates and improves sensitivity in label-free quantitative proteomics. *Journal of Proteome Research*. 19(5):1975–1981.

Mohan, V. *et al*. 2018. Comprehensive genomic analysis identifies pathogenic variants in maturity-onset diabetes of the young (MODY) patients in South India. *BMC Medical Genetics*. 19:22.

Mulder, N. 2017. Development to enable precision medicine in Africa. *Personalized Medicine*. 14(6):467–470.

Muthukrishnan, R. & Rohini, R. 2016. LASSO: A feature selection technique in predictive modeling for machine learning. *IEEE International Conference on Advances in Computer Applications (ICACA)*. 18–20.

Nahm, F.S. 2022. Receiver operating characteristic curve: overview and practical use for clinicians. *Korean Journal of Anesthesiology*. 75:25–36.

Nakagawa, S. 2004. A farewell to Bonferroni: the problems of low statistical power and publication bias. *Behavioral Ecology*. 15(6):1044–1045.

Natekin, A. & Knoll, A. 2013. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*. 7:21.

Negahdar, M. *et al*. 2014. GCK-MODY diabetes as a protein misfolding disease: The mutation R275C promotes protein misfolding, self-association and cellular degradation. *Molecular and Cellular Endocrinology*. 382(1):55–65.

Nesvizhskii, A.I. 2014. Proteogenomics: concepts, applications, and computational strategies. *Nature Methods*. 11(11):1114–1125.

Neyman, J. 1941. Biometrika Trust Fiducial Argument and the Theory of Confidence Intervals. *Biometrika*. 32(2):128–150.

Nkonge, K.M., Nkonge, D.K. & Nkonge, T.N. 2020. The epidemiology, molecular pathogenesis, diagnosis, and treatment of maturity-onset diabetes of the young (MODY). *Clinical Diabetes and Endocrinology*. 6:20.

Nurk, S. *et al*. 2022. The complete sequence of a human genome. *Science*. 376(6588):44–53.

Oladipupo, A.T. 2010. New Advances in Machine Learning. Chapter 3: Types of Machine Learning Algorithms. Zhang Yagang (ed.). IntechOpen.

Pang, Y., Wang, Z., Jhong, J.H. & Lee, T.Y. 2021. Identifying anti-coronavirus peptides by incorporating different negative datasets and imbalanced learning strategies. *Briefings in Bioinformatics*. 22(2):1085–1095.

Park, C.Y., Käll, L., Klammer, A.A., Maccoss, M.J. & Noble, W.S. 2008. Rapid and accurate peptide identification from tandem mass spectra. *Journal of Proteome Research*. 7(7):3022–3027.

Pascovici, D., Handler, D.C.L., Wu, J.X. & Haynes, P.A. 2016. Multiple testing corrections in quantitative proteomics: A useful but blunt tool. *Proteomics*. 16(18):2448–2453.

Patel, K.A. *et al*. 2017. Heterozygous RFX6 protein truncating variants are associated with MODY with reduced penetrance. *Nature Communications*. 8:888.

Pedregosa, F. *et al*. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 12:2825–2830.

Perkins, D.N., Pappin, D.J.C., Creasy, D.M. & Cottrell, J.S. 1999. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*. 20:3551–3567.

Picard, R.R. & Cook, R.D. 1984. Cross-validation of regression models. *Journal of the American Statistical Association*. 79(387):575–583.

Pothuganti, S. 2018. Review on over-fitting and under-fitting problems in Machine Learning and solutions. *International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering*. 7:3692–3695.

Principi, N., Berioli, M.G., Bianchini, S. & Esposito, S. 2017. Type 1 diabetes and viral infections: What is the relationship? *Journal of Clinical Virology*. 96:26–31.

Pruitt, K.D., Tatusova, T. & Maglott, D.R. 2007. NCBI reference sequences (RefSeq): A curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Research*. 35(Suppl 1):D61–D65.

Qi, Y. 2012. Ensemble Machine Learning. Zhang C & Ma Y (eds.). New York: Springer.

Quinlan, J.R. 1987. Decision trees as probabilistic classifiers, In Langley Pat (Ed.). *Proceedings Of the Fourth International Workshop on Machine Learning*, Morgan Kaufmann. 31–37.

Radzvilas, M., Peden, W., De Pretis, F., Bangu, S., Ippoliti, E. & Antonutti William Peden, M.B. 2021. A battle in the statistics wars: A simulation-based comparison of Bayesian, Frequentist and Williamsonian methodologies. *Synthese*. 199:13689–13748.

Rafique, I., Mir, A., Saqib, M.A.N., Naeem, M., Marchand, L. & Polychronakos, C. 2021. Causal variants in Maturity Onset Diabetes of the Young (MODY) A systematic review. *BMC Endocrine Disorders*. 21:223.

Rakhshani, A., Lagani, V. & Tsamardinos, I. 2015. Performance-estimation properties of cross-validation-based protocols with simultaneous hyper-parameter optimization. *International Journal of Artificial Intelligence Tools*. 24(5):1540023.

Redondo, M.J., Hagopian, W.A., Oram, R., Steck, A.K., Vehik, K., Weedon, M., Balasubramanyam, A. & Dabelea, D. 2020. The clinical consequences of heterogeneity within and between different diabetes types. *Diabetologia*. 63(10):2040–2048.

Reiter, L *et al*. 2009. Protein identification false discovery rates for very large proteomics data sets generated by tandem mass spectrometry. *Molecular and Cellular Proteomics*. 8(11):2405–2417.

Romito, A. & Cobellis, G. 2015. Pluripotent stem cells: Current understanding and future directions. *Stem Cells Internationals*. 2016:9451492.

Rosen, B.E. 1996. Ensemble learning using decorrelated neural networks. *Connections Science*. 8(3–4):373–384.

Rosenberger, G. *et al* 2017. Statistical control of peptide and protein error rates in large-scale targeted DIA analyses. *Nature methods*. 14(9):921.

Ryan, H.T. & Chawla, N. V. 2013. Imbalanced datasets: From sampling to classifiers, He Haibo & Y. Ma (eds.). *Imbalanced Learning: Foundations, Algorithms, and Applications*, Wiley. 43–59.

Sagen, J. V. *et al.* 2004. Brief genetics report permanent neonatal diabetes due to mutations in KCNJ11 encoding Kir6.2 patient characteristics and initial response to sulfonylurea therapy. *Diabetes*. 53(10):2713–2718.

Saito, T. & Rehmsmeier, M. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*. 10:3.

Sampson, D.L., Parker, T.J., Upton, Z. & Hurst, C.P. 2011. A comparison of methods for classifying clinical samples based on proteomics data: A case study for statistical and Machine Learning approaches. *PLoS ONE*. 6(9):e24973.

Sanger, F., Nicklen, S. & Coulson, A.R. 1977. DNA sequencing with chain-terminating inhibitors (DNA polymerase/nucleotide sequences/bacteriophage 4X174). *Proceedings of the National Academy of Sciences*. 74(12):5463–5467.

Sankari, E.S. & Manimegalai, D. 2017. Predicting membrane protein types using various decision tree classifiers based on various modes of general PseAAC for imbalanced datasets. *Journal of Theoretical Biology*. 435:208–217.

Savitski, M.M., Wilhelm, M., Hahne, H., Kuster, B. & Bantscheff, M. 2015. A scalable approach for protein false discovery rate estimation in large proteomic data sets. *Molecular and Cellular Proteomics*. 14(9):2394–2404.

Searle, B.C. 2010. Scaffold: A bioinformatic tool for validating MS/MS-based proteomic studies. *Proteomics*. 10(6):1265–1269.

Serang, O. & Käll, L. 2015. Solution to statistical challenges in proteomics is more statistics, not less. *Journal of Proteome Research*. 14(10):4099–4103.

Serang, O., Maccoss, M.J. & Noble, W.S. 2010. Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data. *Journal of Proteome Research*. 9(10):5346–5357.

Shaffer, J.P. 1986. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*. 81(395):826–831.

Shen, C. 2019. Nucleic acid-based cellular activities, in *Diagnostic Molecular Biology*, Elsevier. 27–57.

Shen, H. & Chou, K. 2006. Ensemble classifier for protein fold pattern recognition. *Bioinformatics*. 22(14):1717–1722.

Shen, C., Sheng, Q., Dai, J., Li, Y., Zeng, R. & Tang, H. 2009. On the estimation of false positives in peptide identifications using decoy search strategy. *Proteomics*. 9(1):194–204.

Sheynkman, G.M., Shortreed, M.R., Frey, B.L., Scalf, M. & Smith, L.M. 2014. Large-scale mass spectrometric detection of variant peptides resulting from nonsynonymous nucleotide differences. *Journal of Proteome Research*. 13(1):228–240.

Shi, J. & Wu, F. 2009. Protein inference by assembling peptides identified from tandem mass spectra. *Current Bioinformatics*. 4(3):226–233.

Shi, Y., Inoue, H., Wu, J.C. & Yamanaka, S. 2017. Induced pluripotent stem cell technology: A decade of progress. *Nature Reviews Drug Discovery*. 16(2):115–130.

Shteynberg, D., Nesvizhskii, A.I., Moritz, R.L. & Deutsch, E.W. 2013. Combining results of multiple search engines in proteomics. *Molecular and Cellular Proteomics*. 12(9):2383–2393.

Sipper, M. 2022. High per parameter: A large-scale study of hyperparameter tuning for Machine Learning algorithms. *Algorithms*. 15(9):315.

De Smet, F., Moreau, Y., Engelen, K., Timmerman, D., Vergote, I. & De Moor, B. 2004. Balancing false positives and false negatives for the detection of differential expression in malignancies. *British Journal of Cancer*. 91:1160–1165.

Sofaer, H.R., Hoeting, J.A. & Jarnevich, C.S. 2019. The area under the precision-recall curve as a performance metric for rare binary events. *Methods in Ecology and Evolution*. 10(4):565–577.

Soltis, P.S., Nelson, G., Zare, A. & Meineke, E.K. 2020. Plants meet machines: Prospects in Machine Learning for plant biology. *Applications in Plant Sciences*. 8(6):e11371.

Soori, M., Arezoo, B. & Dastres, R. 2023. Artificial intelligence, Machine Learning and deep learning in advanced robotics, a review. *Cognitive Robotics*. 3:54–70.

Søvik, O., Irgens, H.U., Molnes, J., Sagen, J. V., Bjørkhaug, L., Ræder, H., Molven, A. & Njølstad, P.R. 2013. Monogenic diabetes mellitus in Norway. *Norsk Epidemiologi*. 23(1):55–60.

Spahr, C.S. *et al*. 2001. Towards defining the urinary proteome using liquid chromatography-tandem mass spectrometry I. Profiling an unfractionated tryptic digest. *Proteomics*. 1(1):93–107.

Spivak, M., Weston, J., Bottou, L., Käll, L. & Noble, W.S. 2009. Improvements to the Percolator algorithm for peptide identification from shotgun proteomics data sets. *Journal of Proteome Research*. 8(7):3737–3745.

Starovoitov, V. & Golub, Y. 2020. Comparative study of quality estimation of binary classification. *Informatics*. 17:87–101.

Stefan, Y., Meda, P., Neufeld, M. & Orci, L. 1987. Stimulation of insulin secretion reveals heterogeneity of pancreatic B cells in vivo. *Journal of Clinical Investigation*. 80(1):175–183.

Steiner, D.J., Kim, A., Miller, K. & Hara, M. 2010. Pancreatic islet plasticity: Interspecies comparison of islet architecture and composition. *Islets*. 2(3):135-145.

Stevens, S.M., Prokai-Tatrai, K. & Prokai, L. 2008. Factors that contribute to the misidentification of tyrosine nitration by shotgun proteomics. *Molecular and Cellular Proteomics*. 7(12):2442–2451.

Stone, M.H. 1936. The theory of representation for Boolean algebras. *Transactions of the American Mathematical Society*. 40(1):37–111.

Stone, M. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*. 36(2):111–147.

Storey, J.D. 2002. A direct approach to false discovery rates. *Journal of the Royal Statistical Society Series B: Statistical Methodology*. 64(3):479–498.

Van Stralen, K.J., Stel, V.S., Reitsma, J.B., Dekker, F.W., Zoccali, C. & Jager, K.J. 2009. Diagnostic methods I: sensitivity, specificity, and other measures of accuracy. *Kidney International*. 75(12):1257–1263.

Strauss, M.T. *et al*. 2024. AlphaPept: a modern and open framework for MS-based proteomics. *Nature Communications*. 15(1):2168.

Sweidan, D. & Johansson, U. 2021. Probabilistic prediction in Scikit-learn, *The 18th International Conference on Modeling Decisions for Artificial Intelligence*.

Sykiotis, G.P., Kalliolias, G.D. & Papavassiliou, A.G. 2005. Pharmacogenetic principles in the Hippocratic writings. *Journal of Clinical Pharmacology*. 45(11):1218–1323.

Tabb, D.L., Mcdonald, W.H. & Yates, J.R. (2002). DTASelect and contrast: Tools for assembling and comparing protein identifications from shotgun proteomics. *Journal of Proteome Research*. 1(1):21-26.

Tan, Y. De & Xu, H. 2014. A general method for accurate estimation of false discovery rates in identification of differentially expressed genes. *Bioinformatics*. 30(14):2018–2025.

Tanha, J., van Someren, M. & Afsarmanesh, H. 2015. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*. 8:355–370.

Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N. & Asadpour, M. 2020. Boosting methods for multi-class imbalanced data classification: an experimental review. *Journal of Big Data*. 7:70.

Tanner, S., Shu, H., Frank, A., Wang, L.C., Zandi, E., Mumby, M., Pevzner, P.A. & Bafna, V. 2005. InsPecT: Identification of posttranslationally modified peptides from tandem mass spectra. *Analytical Chemistry*. 77(14):4626–4639.

Tattersall, R.B., Fajans, S.S. & Arbor, A. 1975. A difference between the inheritance of classical juvenile-onset and maturity-onset type diabetes of young people. *Diabetes*. 24(1):44–53.

Tibshirani, R. 1996. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*. 58(1):267–288.

Timm, W., Scherbart, A., Böcker, S., Kohlbacher, O. & Nattkemper, T.W. 2008. Peak intensity prediction in MALDI-TOF mass spectrometry: A machine learning study to support quantitative proteomics. *BMC Bioinformatics*. 9:443.

Tirone, T.A. & Brunicardi, F.C. 2001. Overview of glucose regulation. *World Journal of Surgery*. 25(4):461–467.

Trowell, H.C. 1982. Ants distinguish diabetes mellitus from diabetes insipidus. *British Medical Journal*. 285(6336):217.

Udler, M.S. *et al*. 2018. Type 2 diabetes genetic loci informed by multi-trait associations point to disease mechanisms and subtypes: A soft clustering analysis. *PLoS Medicine*. 15(9):e1002654.

Ulintz, P.J., Zhu, J., Qin, Z.S. & Andrews, P.C. 2006. Improved classification of mass spectrometry database search results using newer Machine Learning approaches. *Molecular and Cellular Proteomics*. 5(3):497–509.

Urakami, T. 2019. Maturity-onset diabetes of the young (MODY): Current perspectives on diagnosis and treatment. *Diabetes, Metabolic Syndrome and Obesity*. 12:1047–1056.

Valkovicova, T., Skopkova, M., Stanik, J. & Gasperikova, D. 2019. Novel insights into genetics and clinics of the HNF1A-MODY. *Endocrine Regulations*. 53(2):110–134.

Vaudel, M., Burkhart, J.M., Zahedi, R.P., Oveland, E., Berven, F.S., Sickmann, A., Martens, L. & Barsnes, H. 2015. PeptideShaker enables reanalysis of MS-derived proteomics data sets. *Nature Biotechnology*. 33:22–24.

Vovk, V. & Wang, R. 2021. E-values: Calibration, combination, and applications. *The Annals of Statistics*. 49(3):1736–1754.

Wang, D. *et al*. 2019. A deep proteome and transcriptome abundance atlas of 29 healthy human tissues. *Molecular Systems Biology*. 15(2):e8503.

Wang, G., Wu, W.W., Zhang, Z., Masilamani, S. & Shen, R.F. 2009. Decoy methods for assessing false positives and false discovery rates in shotgun proteomics. *Analytical Chemistry*. 81(1):146–159.

Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q. & Kennedy, P.J. 2016. Training deep Neural Networks on imbalanced data sets, in *2016 International Joint Conference on Neural Networks*. 4368–4374.

Wang, X. *et al*. 2018. Target-decoy-based false discovery rate estimation for large-scale metabolite identification. *Journal of Proteome Research*. 17(7):2328-2334

Watson, J.D. 1990. The human genome project: Past, present and future. *Science*. 248(4951):44–49.

Weinreich, S.S. *et al*. 2015. A decade of molecular genetic testing for MODY: A retrospective study of utilization in the Netherlands. *European Journal of Human Genetics*. 23(1):29–33.

Winkler, L. & Murphy, A.; H. 1973. Experiments in the laboratory and the real world. *Organizational Behavior and Human Performance*. 10(2):252–270.

Wright, J.C. & Choudhary, J.S. 2016. DecoyPyrat: Fast non-redundant hybrid decoy sequence generation for large scale proteomics. *Journal of Proteomics & Bioinformatics*. 9(6):176–180.

Wu, X., Tseng, C.W. & Edwards, N. 2007. HMMatch: Peptide identification by spectral matching of tandem mass spectra using hidden Markov models. *Journal of Computational Biology*. 14(8):1025–1043.

Xu, Y. & Goodacre, R. 2018. On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing*. 2:249–262.

Yadav, S. & Shukla, S. 2016. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification, *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, Institute of Electrical and Electronics Engineers Inc. 78–83.

Yamashita, H. *et al*. 2011. Analysis of the HLA and non-HLA susceptibility loci in Japanese type 1 diabetes. *Diabetes/Metabolism Research and Reviews*. 27(8):844–848.

Yang, L. & Shami, A. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*. 415:295–316.

Yang, Y. & Chan, L. 2016. Monogenic diabetes: What it teaches us on the common forms of type 1 and type 2 diabetes. *Endocrine Reviews*. 37(3):190–222.

Yoo, C., Ramirez, L. & Liuzzi, J. 2014. Big data analysis using modern statistical and Machine Learning methods in medicine. *International Neurourology Journal*. 18(2):50–57.

Yorifuji, T. *et al*. 2012. Comprehensive molecular analysis of Japanese patients with pediatric-onset MODY-type diabetes mellitus. *Pediatric Diabetes*. 13:26–32.

Yu, W., Wu, B., Lin, N., Stone, K., Williams, K. & Zhao, H. 2006. Detecting and aligning peaks in mass spectrometry data with applications to MALDI. *Computational Biology and Chemistry*. 30:27–38.

Zhang, H. *et al*. 2022. Model for integration of monogenic diabetes diagnosis into routine care: The personalized diabetes medicine program. *Diabetes Care*. 45(8):1799–1806.

Zhang, J., Ma, J., Dou, L., Wu, S., Qian, X., Xie, H., Zhu, Y. & He, F. 2008. Bayesian nonparametric model for the validation of peptide identification in shotgun proteomics. *Molecular and Cellular Proteomics*. 8(3):547–557.

Zhang, S., Shan, Y., Jiang, H., Liu, J., Zhou, Y., Zhang, L. & Zhang, Y. 2017. The null-test for peptide identification algorithm in shotgun proteomics. *Journal of Proteomics*. 163:118–125.

Zhang, T., Lin, W., Vogelmann, A.M., Zhang, M., Xie, S., Qin, Y. & Golaz, J.C. (2021). Improving convection trigger functions in deep convective parameterization schemes using Machine Learning. *Journal of Advances in Modeling Earth Systems*. 13(5):e2020MS002365.

Zhou, Z. 2012. Ensemble methods foundations and algorithms. Herbrich Ralf & Graepel Thore (eds.). Cambridge, UK: Chapman & Hall / CRC.

Zimmermann, M., Xu, C., Coen-Pirani, P. & Jiang, X. 2023. Empirical study of overfitting in deep learning for predicting breast cancer metastasis. *Cancers*. 15(7):1969.