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

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: Dr. Marc Vaudel

March 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.

Lorensha Naidoo

March 2025

Abstract

Monogenic diabetes is a rare form of paediatric diabetes caused by the dysfunction of a gene responsible for pancreatic β-cell insulin production due to a single variation resulting in hyperglycaemic complications. Maturity-onset diabetes of the young (MODY) is a sub-type of monogenic diabetes accounting for 2 – 5 % of diabetic cases. Patient classification has revealed undiagnosed symptomatic cohorts that are speculated to be a result of unknown genetic variants. Identification of these variants is important for improving precision medicine of MODY by avoiding misdiagnosis 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) of different levels of confidence are produced however, there is no clear distinction between correct and incorrect PSMs. Additionally, discriminating variant peptides from canonical peptides remains challenging due to their low frequency and similarities between sequences.

The target-decoy approach (TDA) is a common method to classify incorrect and correct 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 discrimination performance of variant PSMs.

To this end, this study conducts an exploratory analysis to improve classification of both canonical and variant PSMs by 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 labelled target and decoy spectra, characterised by multiple scoring functions issued during annotation. ML base models are built using Scikit-learn which produces a prediction probability $p(\text{-}1)$ 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 three implementation tasks investigated in this work.

Firstly, the discrimination performance of five decision-tree ensemble architectures is evaluated: Random Forest, Gradient Boosting, Histogram Gradient Boosting, ExtraTrees and XGBoost. Secondly, a novel concept using decoy variant PSMs is evaluated against the classical decoy approach by classifying canonical and variant PSMs of proteomic data from pluripotent stem cells induced with the HNF1A(MODY3) variant. Lastly, a novel Bayesian inferred posterior

error probability (PEP) method is investigated for comparing the peptide significance of MODY genes of the two decoy strategies.

Spectra probability distribution of ExtraTrees provided the most reassuring performance and was used for PSM classification of 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 as an optimistic starting point for future works. 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 **Dr. 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 Center 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 particular cold winter in Bergen.

I extend my appreciation to my fellow colleagues of Dr. Vaudels' 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 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 practiced such as localising conditions to human anatomy as an early approach to precision medicine. The 'Father of Western Medicine', Hippocrates, later then 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 generalized approach (Kurland and Molgaard, 1981). The ability to archive patient data allowed practitioners to analytically develop standardized 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 the genetic factors influencing biological processes of complex diseases through Genome wide association studies (GWAS) has provided a more accurate measure for diagnostics than observing phenotypical symptoms. The genetic diversity amongst human populations can be recognized and health conditions with overlapping traits, including rare subtypes, can be identified with unique markers (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 personalized medicine. This is especially beneficial in the fields of complex non-communicable disorders such as cancer, cardiovascular diseases and diabetes.

1.2 Diabetes Mellitus

Glucose is a primary energy molecule responsible cellular function in vertebrate species. It enters the blood stream 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 bioprocess of these cells is activated by the reception of glucose (**Figure 1.1**).

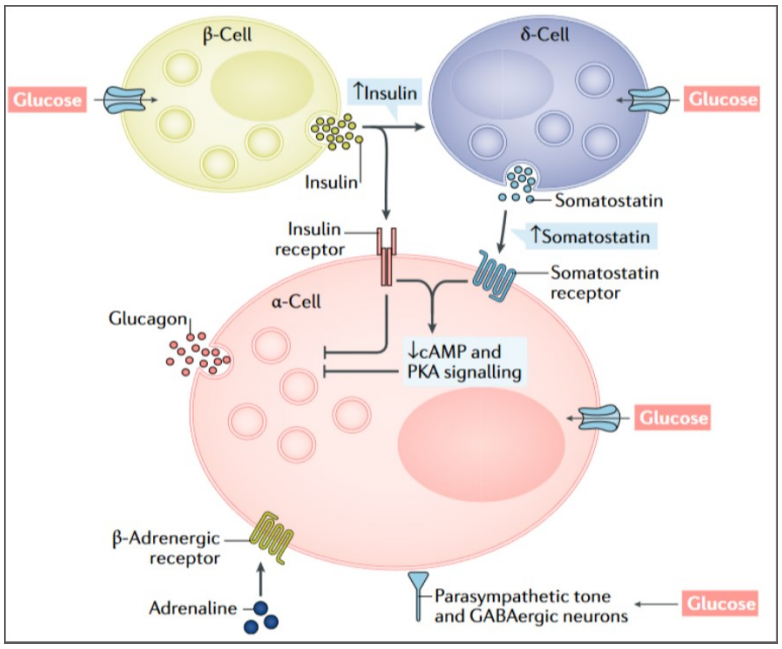


Figure 1.1 Pancreatic cellular interaction during glucose regulation. The biochemical response of pancreatic cells is activated in the presence of glucose. The a-cells process stored glucagon, and the B-cells release insulin. This figure is unmodified from Gromada *et al.* (2018).

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 releases insulin. Insulin is a protein molecule required by tissue cells e.g. fat and muscles, to activate glucose receptors. Glucose, and interacting molecules, can then enter the cells as an energy source driving their metabolic processes (Stefan *et al.*, 1987). Absence of insulin results in congested glucose levels (hyperglycaemia) in the blood/plasma to which the extent negatively affects human health. Complications arising from severe hyperglycaemia include microvascular conditions: nephropathy (kidney dysfunction), retinopathy (eye loss), neuropathy (nerve damage), and macrovascular conditions: cardiovascular (heart),

cerebrovascular (stroke) and peripheral vascular (limb function) (Krentz *et al*., 2007). Diabetes mellitus is the chronic metabolic condition developing from insulin dysfunction or poor regulation.

In 2021, the International Diabetes Federation reported that 537 million people globally are diagnosed with diabetes (IDF, 2021). Cases are continually increasing along with the development of socio-economic landscapes. The disorder, however, has been acknowledged over thousands of years. Ancient civilizations such as the Indians, Egyptians and Greeks reporting the urine of 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 the blood/plasma/serum glucose under different conditions, such as: after consuming a glucose liquid (Oral Glucose Tolerance Test), 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 (T2D conditions developing during pregnancy). T1D is an immune system response detecting 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 identifying several subtypes as depicted in **Figure 1.2[a]**. The heterogeneity of diabetes is now characterized by genetic markers, molecular factors and age of onset. Influence of environmental factors are considered in the modern scope of precision medicine.

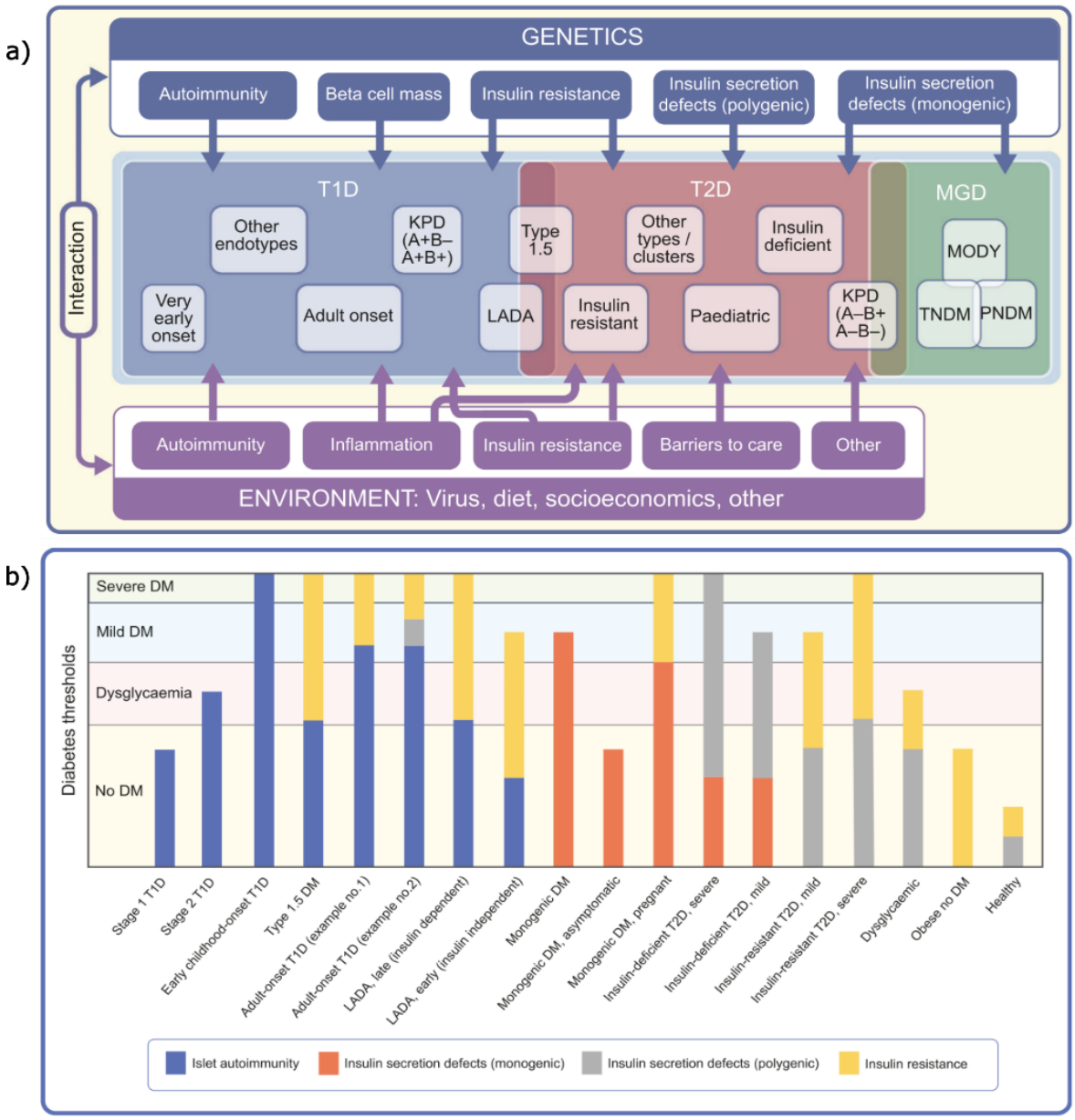


Figure 1.2 Heterogeneity of diabetes. Diabetes subtypes are classified by genetic dysfunction and biochemical onset. **(a)** Spectrum of diabetes subtypes within the broader divisions of T1D, T2D and monogenic diabetes. Influence of environmental factors are considered. **(b)** composition of biochemical manifestation of diabetes subtypes. Severity of subtypes and treatment response are determined by their complexity. Both figures are unmodified from Redondo *et al*. (2020).

The advanced 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. The expression of certain alleles, for example, of the human leukocyte antigen (HLA) are markers 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 that had previously been misclassified as dominant class types (T1D and T2D), are also currently receiving better acknowledgement. This is the case for 'special' types such as 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 are forms of 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 conducting an extensive diabetes familial 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 to 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 on-going research revealing new insights into the condition.

1.3.2 Genetics factors characterising MODY

The heterogeneity 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 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 (HNF1A) are responsible for majority of MODY cases. An increase in rare MODY subtypes is continuously being uncovered in current research such as by Patel *et al*. (2017) and Mohan *et al*. (2018) for example. Although this an exciting and interesting new area of research, the scope of this work will focus on the 14 MODY genes established in literature for brevity. A single variant (haploinsufficiency) (Garin *et al.*, 2008) in one of these genes results in the disruption of β-cell function causing insulin suppression. The term 'variant' describes changes in the deoxyribonucleic acid (DNA) sequence that influences gene functionality and protein expression. The term 'mutation' is often used in literature, referring to the same process, however regarding topics 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	Pathophysiology	Treatment
HNF4A (MODY 1)	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 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). Frequency of SNPs differs amongst population groups (Choudhury *et al*., 2014) and can serves as markers in drug response for personalised medicine. Gene alterations manifest as various types such as insertions and 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]**) regarding codon regions (three nucleotides forming an amino acid). The response may be silent (no change), missense (amino acid change), nonsense (stop codon) or a frameshift mutation (insertion or deletion disrupting amino acid sequence). Alterations in the codon produces protein products irregular to metabolic bioprocesses and influences gene expression. A negative response disrupts metabolic bioprocesses leading to severe health concerns.

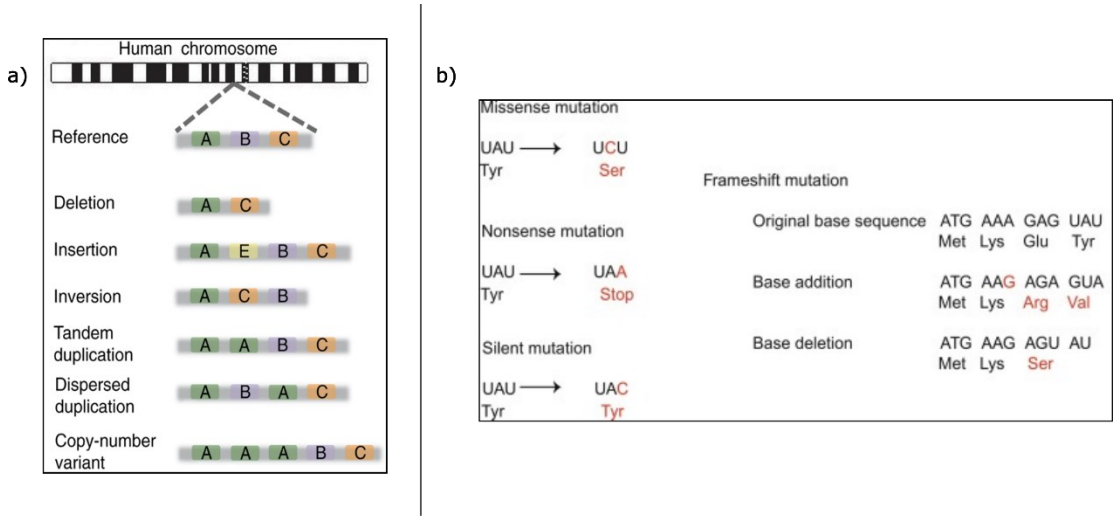


Figure 1.3 Alternative forms of genes and proteins. (a) Types of structural variants caused by nucleotide modifications. The letters (A, B, C, E) represent unspecific nucleotides and demonstrate the modification. This figure is unmodified from (Baker, 2012). **(b)** Point mutations in codons resulting in change of protein expression. This figure is unmodified from Shen (2019)

1.3.2.2 Influence of genomic variants causing MODY

Missense SNPs occurring on transcription factors responsible for the manufacturing process of proteins from DNA at multiple genomic sites increase the severity of a condition (Chang *et al*., 2018). The majority of MODY gene are 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, 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 individual 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 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).

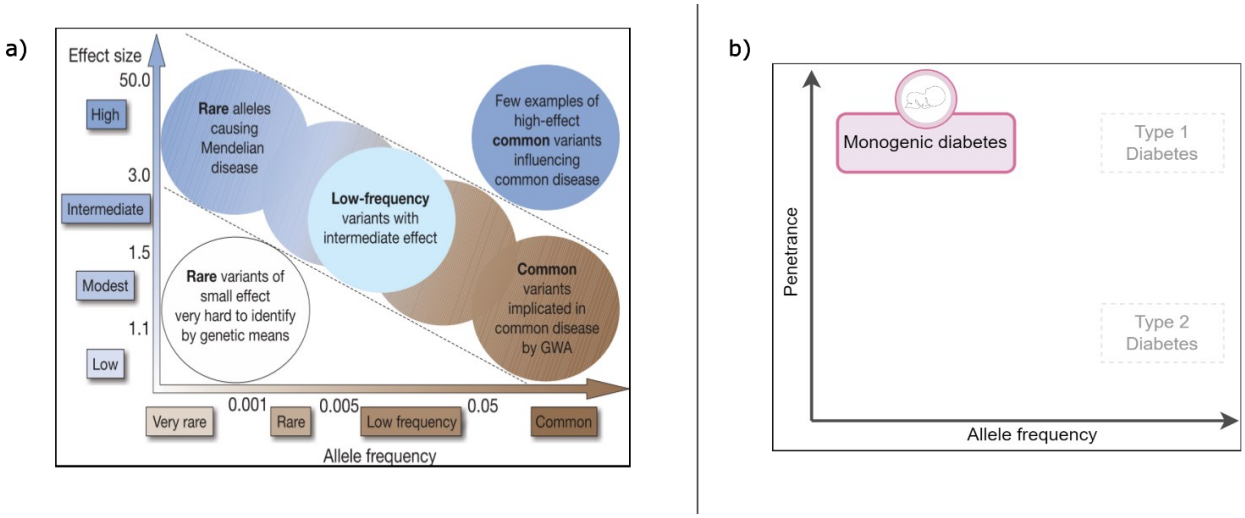


Figure 1.4. Pathogenicity of variants determined by allele frequency. (a) The relationship between frequency of allele variants and phenotypic effect directly correlates to pathophysiology of human diseases. This figure is unmodified from Manolio *et al.* (2009). **(b)** The impact of allele frequency influences 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 low allele frequency with 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. Not all variants cause MODY e.g. HNF1A has over 1200 variants (Valkovicova *et al.*, 2019), and further research into Indian populations has revealed 28.6% (Kavitha *et al.*, 2023) to be pathogenic and in different population groups (Indian and Czech) an average of 70% (Malikova *et al.*, 2020; Kavitha *et al.*, 2023) are potentially pathogenic. Considering this, the frequency of relevant MODY cases is lower than expected. However, chance for detection is improved in cases where penetrance is high such as with 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 tracking familial history of diabetes within populations.

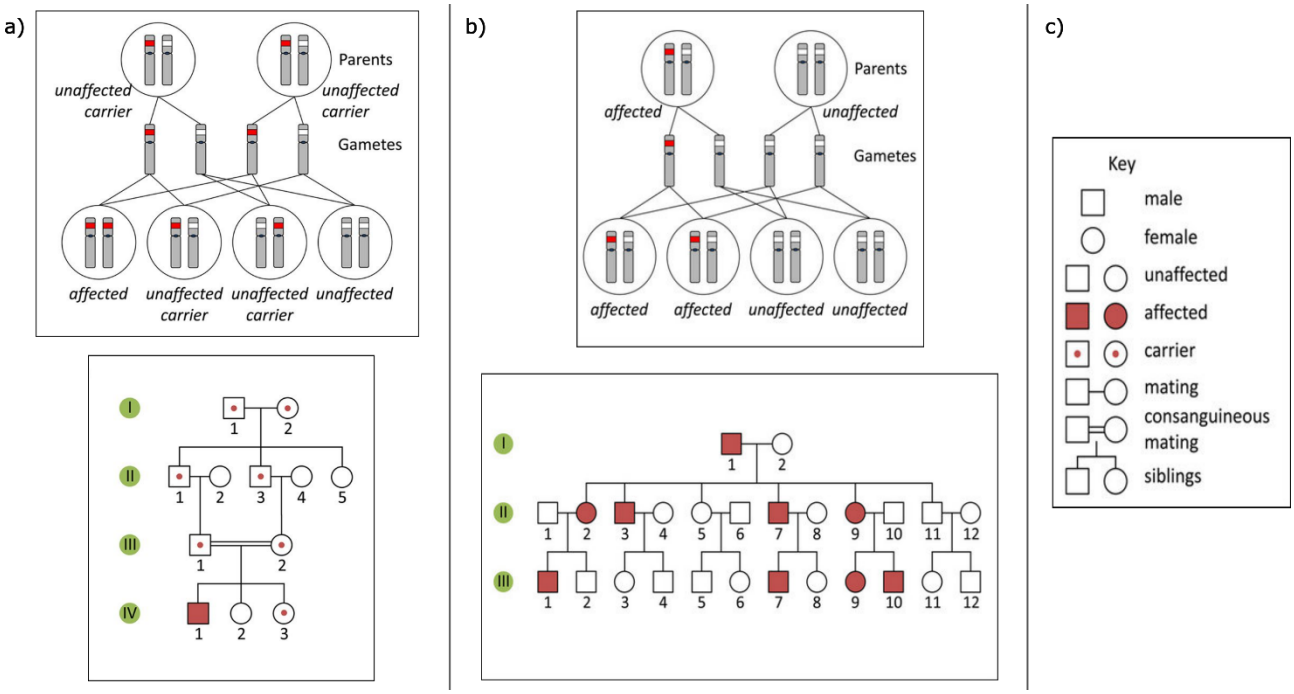


Figure 1.5 Mendelian patterns of inheritance. A pathogenic variant occurring on a single allele within the parental chromosome is passed to offspring in different inheritance patterns. Afflicted individuals are affected within generational lineage at various frequencies in the lower images of **(a)** autosomal recessive, and **(b)** autosomal dominant conditions. Interpretation of the models is within **(c)** Key of image descriptors. These images are modified from Jackson *et al*. (2018).

1.3.4 Patient classification of MODY for precision medicine

The Mohn Center 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 SUMOlysation (small ubiquitin-like modifier proteins responsible for targeted protein expression) (Aukrust *et al*., 2013) have been uncovered as a result of these registries. These records have proven necessary to identify genomic patterns for research and is 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 workflow provided in **Figure 1.6** is a system created for processing suspected MODY cases that has been referred to the MCDPM. The criteria for referrals to be considered includes confirmation of diabetes with supporting clinical diagnostics, supporting family history of diabetes (minimum two generations), 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 workflow filters out NDM subjects and MODY is confirmed for the remaining cases. Curiously, a substantial symptomatic cohort tests negative for MODY which are 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.

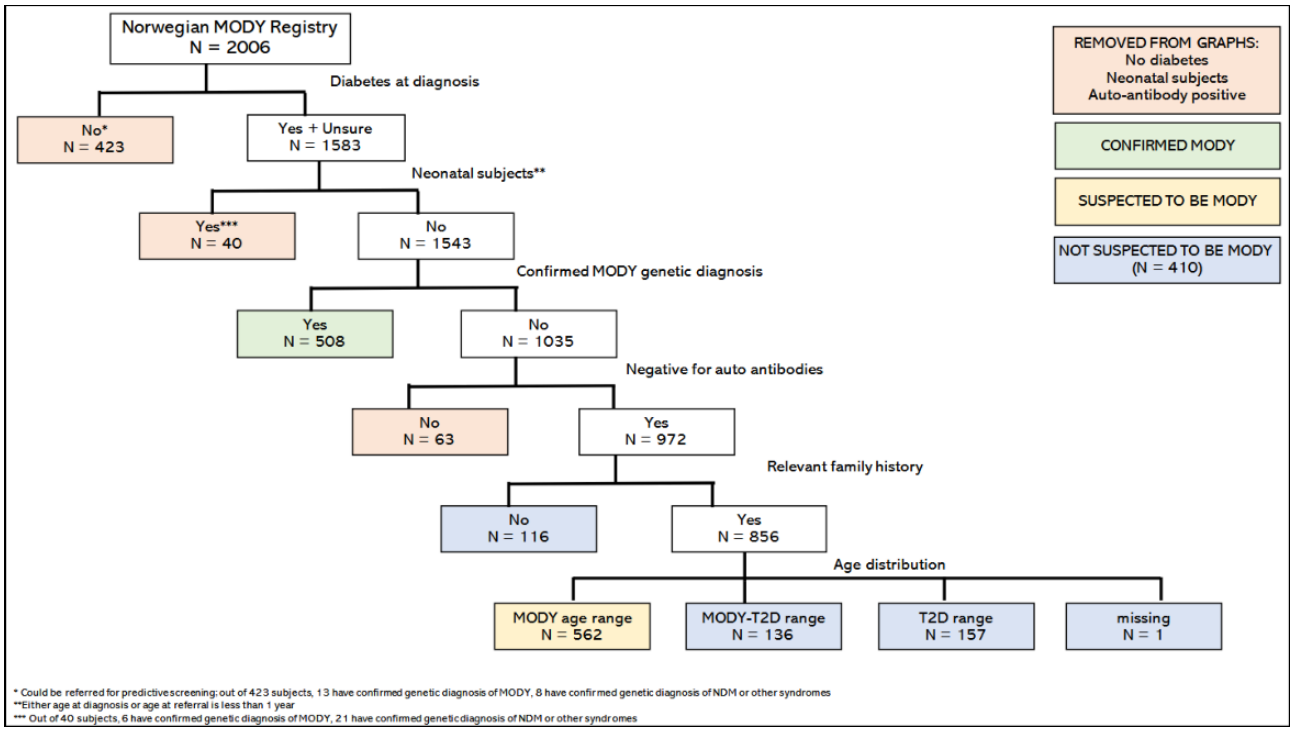


Figure 1.6 MODY patient classification pipeline. A cohort of 2006 subjects suspected of MODY are processed for diagnostics. The colour coded boxes on the upper right corner describe the events of the pipeline. This pipeline is used to visualize the underdetermined cohort of MODY cases (yellow box) after 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-MODY is most 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-MODY has been largely identified however over 50% of cases are due to unidentified variants. A recent study in South Africa (Matsha *et al*., 2020) has revealed HNF1A-MODY as the main contributor to MODY cases in a small cohort of 1643 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 on-going-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 bioprocesses and molecular functioning that requires personalized care. Chronic insulin intake, developed by Banting *et al*. (1922), is the first line of treatment for most diabetic cases. Insulin is regularly administered via injection into the body during 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 sulfonylureas (Sagen *et al.*, 2004), 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 the quality of life can potentially be improved with alternative treatment plans for certain MODY cases.

To be able to provide personalized treatment options, patients are first required to be correctly diagnosed. Monogenic diabetes presents as mild-hyperglycaemia in young people and are 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 to identify pathogenic variants causing MODY to correctly classify undiagnosed individuals and provide personalized care. The nature of this work questions if it possible to identify these variant protein products using existing *in silico* mechanics 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 bioactivity in multicellular organisms and reflect DNA modifications occurring within its complementary gene. However, not all genes are expressed into protein products. Therefore, analysing the proteins in the blood/plasma is confirmation certain genes are processed during specific conditions (Ferkingstad *et al*., 2021). Proteome composition and disease susceptibility can be evaluated, for example, under hyperglycaemic conditions caused by a variant MODY gene (Malikova *et al*., 2020). Through the integration of proteomics and genomics the field of proteogenomics is an effective strategy to comprehensively understand the relationship between genetic variations and their functional protein products.

Shotgun proteomics is a standard technique for identification and quantification of 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 retuned as the end 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). Commonly used 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 sequence that the experimental spectra are aligned with to infer peptide identification (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 which evaluate the similarity between sequences. The alignment with the best or highest scores are 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 on the assumption that each of them 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 results 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 algorithms that are uninformative individually, and there is no clear distinction between correct and incorrect matches. Therefore, to this end, these raw scores require downstream processing to be better interpreted. Additionally, discriminating between variant and canonical peptides remains challenging due to similarities between the sequences. Currently there is no means to improve resolution of variant PSMs during downstream processing. To identify correct PSMs, statistical inference is employed to estimate error rates i.e. the likelihood a spectrum-match is incorrect. The metrics to accomplish this is introduced next in context of standard PSM processing. Then in the methods of this work, these techniques will be explored in context of variant PSMs.

1.5 How to determine reliability of PSMs

1.5.1 The target decoy approach

To establish the reliability of PSMs various strategies, broadly categorized into two approaches, have been integrated into the search engine annotation process as additional components to increase accuracy of true identifications. The first approach involves filtering proteomic data through a set of 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), with established tools such as DTASelect (Tabb *et al*., 2002) and CHOMPER (Eddes *et al*., 2002). Several probability-based approaches such as hidden Markov model (Wu *et al*., 2007), Gaussian distribution (López-Ferrer *et al*., 2004), and Bayesian inference in the well-known PSM processing tool PeptideProphet (Keller *et al*., 2002) are included in the second approach. Although each method is not applicable to every hypothesis, identifying false positives is fundamentally a statistical resolution especially for larger datasets (Cargile *et al*., 2004). The second approach is an effective method introduced by Elias and Gygi (2007) 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 randomness of decoy PSMs (Colinge *et al*., 2003), however 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 true experimental spectra be incorrectly identified as random. The TDA can estimate the frequency of these occurrences to statistically interpret PSM quality.

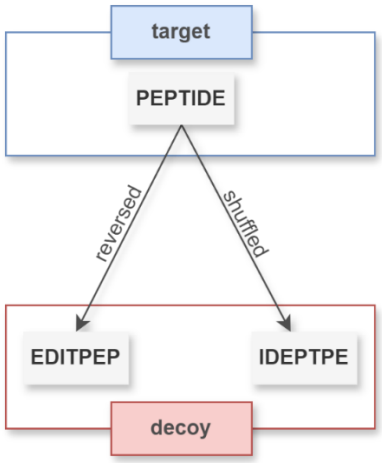


Figure 1.7 Target decoy approach method. An example target sequence 'PEPTIDE' is either reversed or shuffled to create a fake decoy sequence (Bern and Kil, 2011)

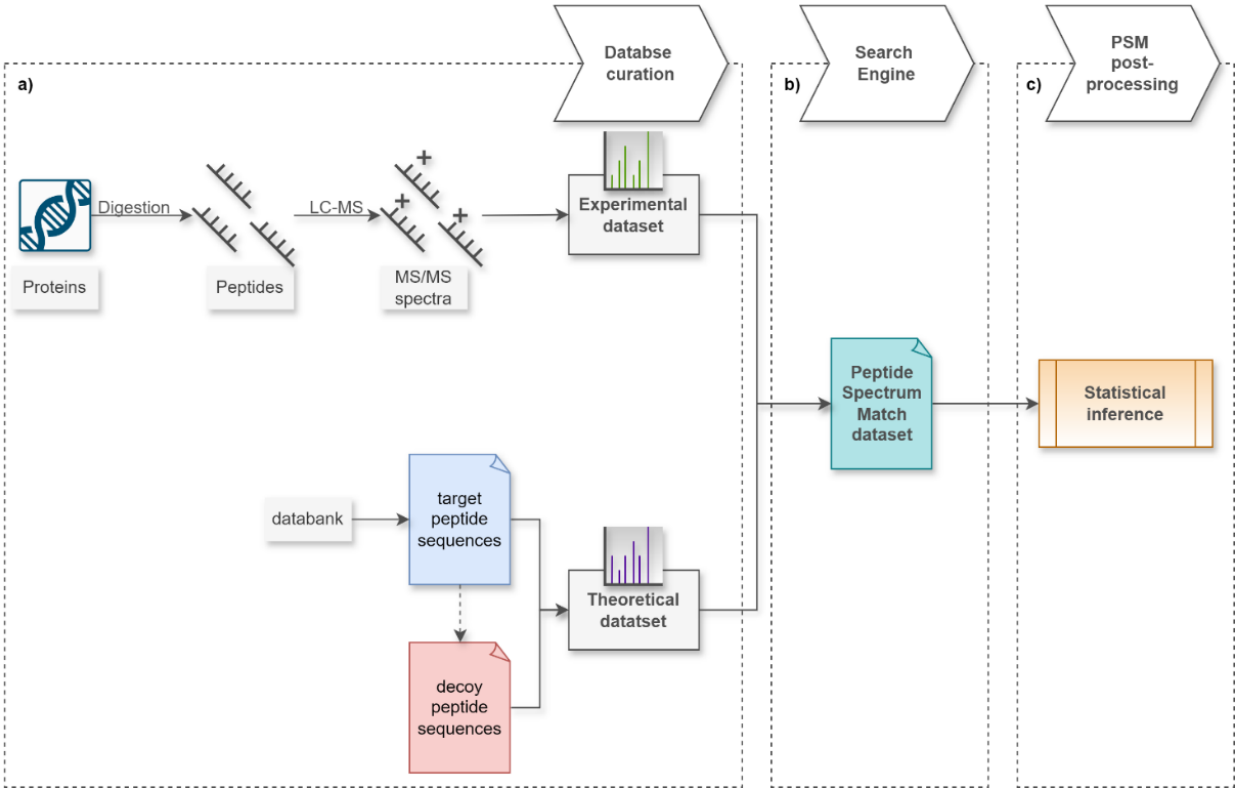


Figure 1.8 Proteogenomic pipeline with target-decoy approach. (a) Database curation: Proteins are fragmented to generate a MS/MS peptide spectrum sequence dataset. Target peptide sequences are harvested from databanks and are combined with decoy sequences to create a theoretical dataset. **(b) Search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

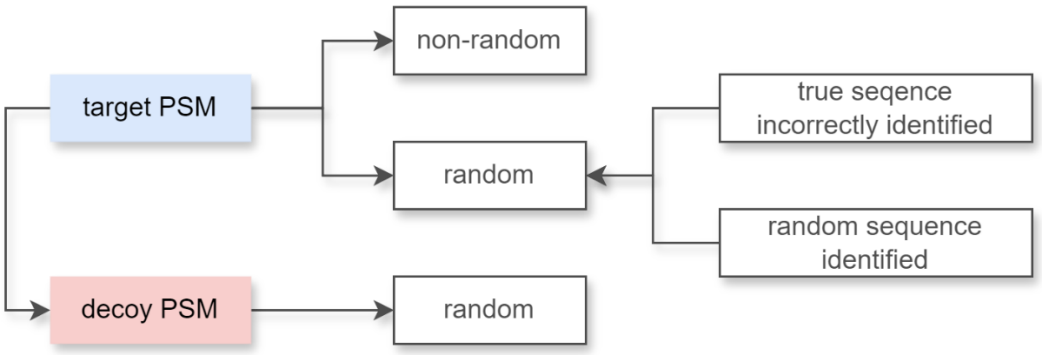


Figure 1.9 Interpretation of target and decoy PSMs. Target PSMs have two possible outcomes and decoy PSMs have one. Random target PSMs is a mixture of two outcomes.

In a separated search, 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 parameters of the search engines 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.

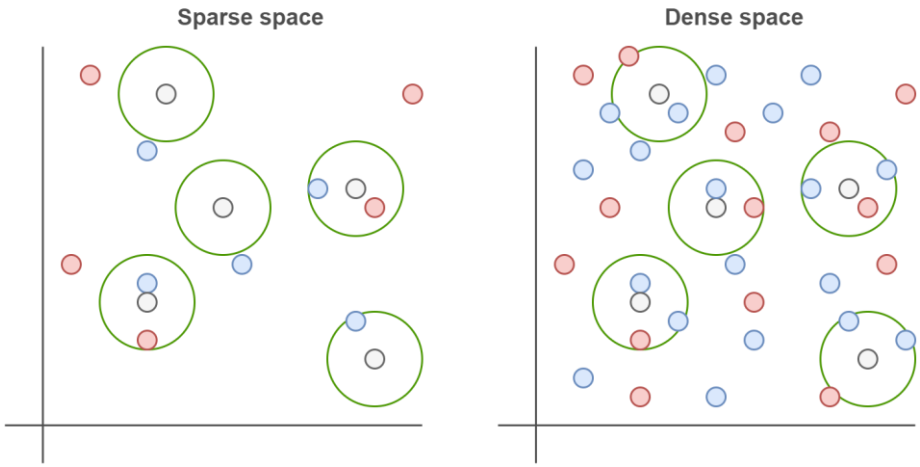


Figure 1.10 Difference in sparse and dense search space environments. The peptide sequences of the observed experimental spectra (grey circles) are identified by its similarity to either theoretical target (blue circle) or decoy (red circle) spectra. The green circle represents the threshold of proximity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

The goal of processing PSMs is not to confirm a spectrum as either a correct or incorrect match. Instead, statistical strategies are used to infer a probability score which indicates the degree of correctness or incorrectness of a PSM. Confidence of PSMs with a high likelihood of being a true match are 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 of random decoy spectra. Statistical inference is applied using the target and decoy PSMs to estimate error rate metrics.

1.5.2 Statistical inference: the null hypothesis

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 the observed spectrum is a correct prediction. These scores issued by a post-processing classifier and are irrelevant as stand-alone scores when reporting the quality of a PSM for identification (Anderson *et al*., 2000). To effectively interpret statistical inference, hypothesis testing is used to provide evidence the probability score is a reliable identification.

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). The null hypothesis ($H0$) (Millikin *et al*., 2020) refers to a statement in which no effect is observed for a given research question or hypothesis. If the statement is true, then $H0$ is accepted. The p-value is a hypothesis (Biau *et al*., 2010), for example, an observed spectrum is an incorrect match if the assigned score is below a threshold (e.g. $a = 0.5$). Therefore, the spectrum is a correct match if the H0 is rejected, which would summarize as:

$$H0: p > 0.5$$

The alternative hypothesis (Ha) (Millikin *et al.*, 2020) is thus accepted as the match is incorrect if the hypothesis is observed:

$$Ha: p \leq 0.5$$

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) (or multiple testing corrections), the likelihood of observing false positive or Type I errors, increases. In remedy, this has been compensated for by use of the Bonferroni correction (Dunn, 1961; Armstrong, 2014). Developed as 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 is 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}), by readjusting it (a), for example, with (n) spectra would be ($a_{FWER} = \frac{a}{n}$), globally, for all PSMs (Storey, 2002). This method is known to be stringent with a disproportionate level of low false positives 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 of the FWER. Within the context of proteomics, it is the goal to discriminate between correct and incorrect spectra matches. In this FDR approach, the error rate is a measurement of its proportion to the 'global' spectra family redacted by a threshold to include only significant values (Käll *et al*., 2008). An FDR threshold of $a = 0.05$ of a spectrum, for example, would imply that it has a 95% chance of being a correct match, while accepting a 5% chance of it being incorrect (Elias and Gygi, 2007). The null hypothesis can be re-written as:

$$H0: FDR \leq 0.05$$

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 true match. In the original TDA method, it was stipulated that the number of incorrect PSMs are proportionate for both the target and decoy datasets (Elias and Gygi, 2007). The FDR was hypothesized 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 would appear as the formulae:

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

The threshold of the FDR includes the statistical inference of processing, for example, using p-values allows for the spectra to be ranked in order of significance. Then for each spectrum the FDR can be estimated based on its rank. For example, a spectrum at position 10 includes 8 target and 2 decoy PSMs, then using the formulae above would equate as:

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

An increase of 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 this example above, an FDR threshold of 1% would reject this spectrum as being a correct match, whereas 5% would not.

Since its inception in 2007, there has been various adaptations of the above formulae to explore alternative false positive hypotheses concerning the proportion of decoy spectra especially in different research strategies. The work of Levitsky *et al*. (2017) suggested to increase decoy

representation by scaling the target-decoy ratio to reduce bias when estimating the FDR for the q-value test statistic. Scaling was likewise suggested in the work of Kim *et al*. (2019) regarding disproportionate small decoy datasets for FDR estimation. 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 become a standardized next to Elias and Gygi(2007), due to it being user friendly and has 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 1700's an essay published by entrustee Richard Price on behalf of the late Reverend Thomas Bayes, conceptualized 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

In the previous chapter, the FDR method was described 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. Therefore, regarding peptide identification tasks, it is necessary to estimate the likelihood of it specifically being a true match. Complementary to the FDR method estimating 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 then implemented the target-decoy strategy (Choi and Nesvizhskii, 2008). The tool Percolator (Käll *et al*., 2007) was then introduced as a standard method to estimate FDR and PEP values to process target and decoy spectra. In these tools, the FDR and PEP are estimated using a null hypothesis {$H0$} modelled by decoy spectra. The interchangeable relationship between FDR and PEP is illustrated in **Figure 1.11**, in which spectra are ordered by a test statistic, then for a spectrum (x) the metrics are estimated in relation to its 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 in relation to the sum of decoy and target spectra with the same test statistic score.

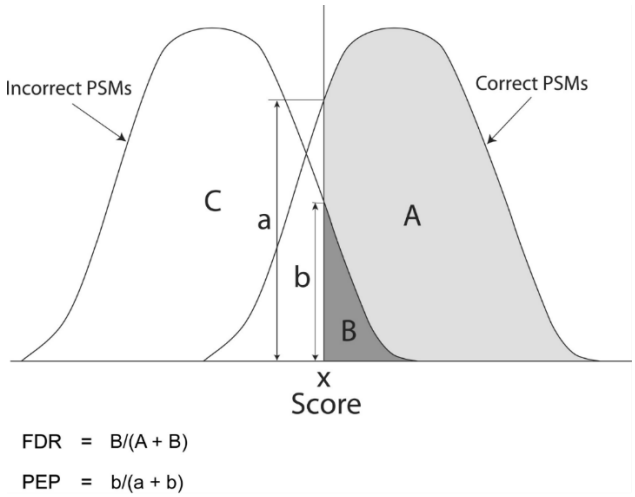


Figure 1.11 Relationship between FDR and PEP. Spectra are ranked by a test statistic. The FDR is the ratio of the number of incorrect and correct spectra ranked above a specific score (x) of a spectrum. The PEP is the ratio of all the incorrect and correct spectra with the same score (x) of a spectrum. The formulae to estimate the FDR and PEP are described. This unmodified image 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 same scoring spectra $\{X\}$, generating the null hypothesis $\{H0\}$ as (Käll *et al.*, 2008):

$$P(H0|X = x) = \frac{f_0 x}{f_1 x}$$

Similar to the FDR, a threshold of significance is applied to the spectra, however the criteria for PEP estimates are more stringent. While a range of 1% - 5% for FDR is acceptable, generally a strict PEP threshold of 1% is recommended for reporting peptide identifications (Käll *et al.*, 2008). The task of identifying mismatched variants in peptides is difficult, however, due to their heavy similarity and limited frequency compared to their canonical counterparts. The PEP estimates require decoy spectra to emulate incorrect matches, which are created by reversing/shuffling the target spectra sequences. Owing to this, it is possible variant peptides bearing similarity to decoy spectra are likely to be unaccounted. More concerningly, variant PSMs are compounded with the canonical PSMs as target spectra and are likely to obtain similar test statistics. 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

Search engines annotate peptides by assigning multiple numerical functions to determine the degree of similarity between observed and theoretical spectra e.g. X!Tandem applies Hyperscore, E-value and Delta score (Craig and Beavis, 2004). PSMs are produced as a function of these scores that are independently reviewed. Multiple scoring functions is beneficial to provide a comprehensive overview of the peptide identification accuracy. Machine learning offers a means to process these scores as a collective and produce a single test statistic inferring an overall measurement of significance. This allows for the PSMs to be interpreted legibly which is especially useful when search engines are combined adjacently to increase resolution (Kwon *et al.*, 2011; Shteynberg *et al.*, 2013).

Machine Learning (ML) is a branch of artificial intelligence harnessing computational technology to execute statistical algorithms programmed to perform large-scale data analysis (Yoo *et al.*, 2014). It is applicable across any research field such as robotics (Soori *et al.*, 2023) and plant biology (Soltis *et al.*, 2020), and here the attributes of ML will be described as pertaining to PSM post-processing. The process of ML entails a learning algorithm or base-

learner, also known as a model, to be *trained* i.e. learn patterns and trends of a dataset to then infer predictions on new *testing* data. The scoring functions of the PSMs are defined as feature vectors (Yoo *et al.*, 2014) or variables (x), and they are labelled (y) representing either a target or decoy class type. A standard ML learning 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 a feature variable characterizing each PSMs class label is used to train the algorithm designed as function e.g. (f): $y = f(x)$. Then testing PSMs with no class labels are classified by the trained model as either a target or decoy using only the feature variables of unseen PSM data.

This is an example of a linear learning algorithm built to perform a classification task i.e. categorizes a data type. Regression tasks involve predicting future values to aid decision making (Timm *et al*., 2008). There are many types of algorithm architectures with different functional strategies that may be better suited to optimally process certain tasks, challenges and datasets. There are three different learning styles (Oladipupo, 2010) and in **Table 1.2**, their characteristics and functionality are briefly described along with a few ML architectures appropriate for the task. Supervised and semi-supervised learning styles can perform classification and regression tasks using deep learning (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. This approach involves classifying unlabelled datapoints from the dataset by iteratively training a model with the labelled data points (Tanha *et al*., 2015). The model is retrained using confidently predicted labelled data points until class labels are assigned. The concept of self-training is synonymous to post-processing target and decoy PSMs due to the nature of target spectra being a composite of non-random and random matches (see Chapter 1.5 and **Figure 1.9**). Random peptide matches are technically unlabelled as they are unknown and through processing they are identified through statistical inference. PSM datasets can range anywhere from the hundreds to millions of peptide entries that require classification, and certain architectures can be optimal to compute these large files.

Table 1.2 Different Machine Learning styles.

	Supervised	**Unsupervised** (Lin *et al.*, 2012)	**Semi-supervised**
Dataset characteristics	Datapoints are labelled and all feature variables are provided	Feature variables only	Some datapoints 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 Classifier, Neural Networks	Clustering, K-means	Linear Regression, Random Forest Classifier, Neural Networks, self-training

The linear function $\{y = f(x)\}$ could be supervised or semi-supervised and is built for a single feature variable (x) thus creating a singular-dimensional search space environment. To include more feature variables (i.e. more scoring functions) would create a higher-dimensional environment (Clarke *et al.*, 2008). With more dimensions, the model can efficiently separate datapoints by their measurement weights of the feature variables. The concept of the illustration **Figure 1.12** is applied to PSM-processing and is used to visualise a learning algorithm performing binary classification on target and decoy spectra. The model can then draw a distinction between the class types to then infer identification on new unseen PSMs.

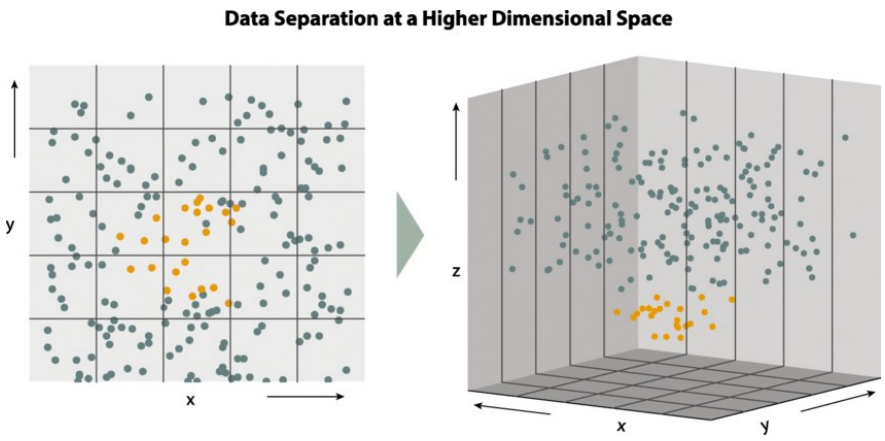


Figure 1.12 Data point separation in different search space dimensions. Data points identified as yellow, and olive, are separated into two-dimension spaces. The lower dimension (**left**) consisting of x and y shows a simple separation. The higher dimension (**right**) consists of x, y, and z increases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

Non-linear models can separate data points with more complexity in high-dimensional environments. There is no limit to the number of dimensions, however an overly complex model can reduce likeness of data points, while insufficient complexity can cause unfair discrimination. Dimensionality needs to be controlled to optimise performance of a learning algorithm. In the next chapter the issue of dimensionality is continued with methods to resolve as pertaining to the performance of a classification algorithm.

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 to separate datapoints by their labelled class types. Then the patterns are inferred onto unlabelled datapoint that are categorized by only their feature vectors. The performance of the model's ability to discriminate the class types is determined by the frequency of errors when separating data points (Anderson *et al*., 2003; Sampson *et al*., 2011). While it can seem that a good model is one that can confidently identify each class type, a state of high accuracy can result in overoptimistic classifications. To explain this, the illustrations in **Figure 1.13** visually describes the performance of binary classification as three possible outcomes: underfitting, overfitting, or a good fit.

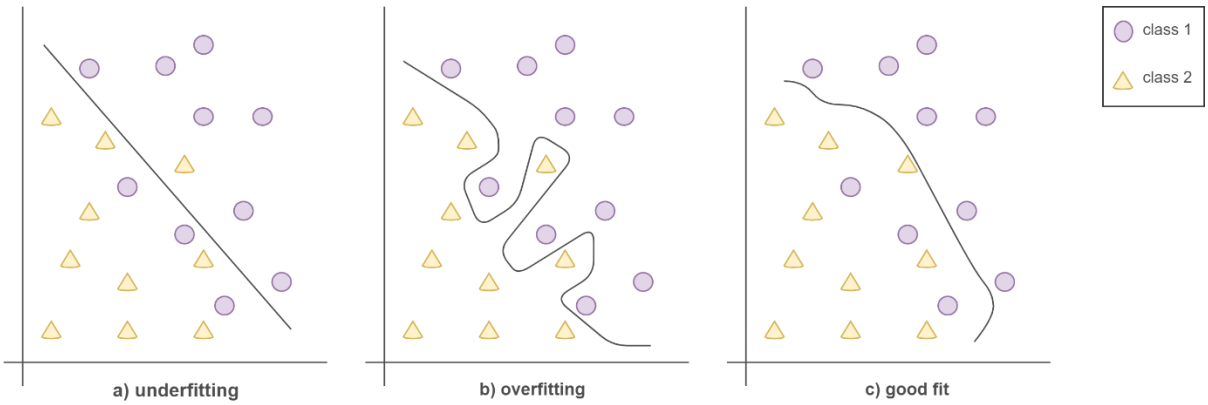


Figure 1.13 Three types of classification performance outcomes. Two class labels are separated by a learning algorithm performing binary classification. The line drawn through each illustration indicates the model's decision-making process when performing binary classification. **(a) underfitting:** the model oversimplifies. **(b) overfitting:** too many redundancies are influencing the classification. **(c) good-fit:** the model generalizes well.

An algorithm can fail to pick up the details of a datasets patterns and oversimplify the learning process. This can occur if the dataset has a poor representation of the data points (Dincer *et al*., 2022) causing a model to under-fit (**Figure 1.13[a]**) and create an overly simplistic distinction between class types. A bias classification is observed due to the algorithm learning no variance from the dataset's patterns. The trade-off between a bias and variance for the learning (training) and classification (testing) process is visualised in **Figure 1.14**. High-bias classification is observed from an unskilled or low complex model, producing high-volumes of errors during both the training and testing process (Pothuganti, 2018).

On the contrary, an algorithm which overly learns the complexities of a dataset causes it to over-fit as illustrated in **Figure 1.13[b]**. This can be caused by a PSM dataset with high amounts of contaminates or poorly fragmented peptides creating noisy random spectra (Anderson *et al*., 2003) and large-scale datasets such as that of MS/MS proteomic data (Frank, 2009). An overly skilled or high-complex model (**Figure 1.14**) is sensitive to these fluctuations, causing high-variance unrelated to the data points. Note as the model becomes more complex the training process appears to generalize well, yet the testing process produces little to no errors. Classification of a PSM dataset would be unreliable due to the expectation of irregularities therefore little to no errors are unrealistic.

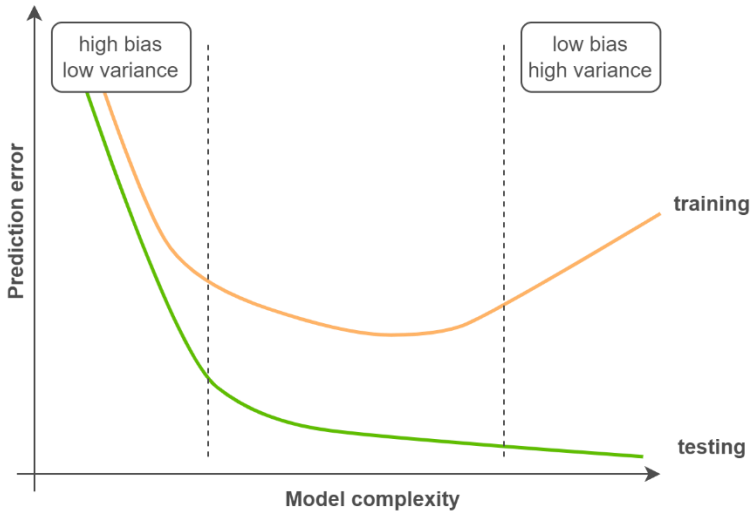


Figure 1.14 Prediction error vs. model complexity. The degree of biasness and variance is observed in the training and testing process of a classification model. A low complex model is left of the first dashed line. A high complex model is right of the second dashed line.

Classified datapoints are more reliable from an algorithm displaying some ambiguity (**Figure 1.13[c]**) to confidently discern between data class types (Anderson *et al*., 2003). A model that is efficiently trained can generalize well with a median amount of complexity (**Figure 1.14**) and produce a low volume of errors in the testing process.

1.6.2 Creating a balanced learning model

1.6.2.1 Hyperparameter tuning

A models classification ability is influence by various factors occurring during the training process. Configuring certain attributes pertaining to a ML architecture can potentially resolve the issue of overfitting and underfitting. Multiple feature variables increase the dimensionality of the search space to provide a higher-level classification. Similarities between data points are measured by their distance to each other in the search space. Data points unsimilar such as two class labels (**Figure 1.12**) with different identifying patterns would be further apart whereas those with the same label will group together (Ghaddar and Naoum-Sawaya, 2017). Including more feature variables during the training process increases the search space. This can negatively impact classification of data points that are likely to be 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 impact prediction values significantly and their importance is graded by the ML architecture. To reduce overfitting, feature variables that poorly contribute to a models learning algorithm can either be removed or their contribution limited to the classification process by penalizing their significance (Deng and Runger, 2012; Blanco *et al*., 2018). This is achieved by regularization which is a method to control a model's complexity. Configuration of regularization is not simplistic as minor influencing feature variables can in fact improve classification separation of data points. There are metrics available to control this such as Lasso and Ridge (Tibshirani, 1996; Muthukrishnan and Rohini, 2016), however the statistics covering this are not of interest in this work.

Alternatively, model complexity can be controlled with hyperparameter tuning. ML architectures are designed using internal values known as parameters that define how the algorithm will process information. Parameters are unalterable functions that decipher the patterns of feature variables characterizing class labels and update the algorithms learning function automatically during the training process (Breiman, 2001). Hyperparameters are external values that are configured before the training process to control the robustness of the learning algorithm (Sipper, 2022). There are numerous types of hyperparameters, and each architecture consists of a unique selection pertaining 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 *A*, *B*, *C* and *D* are listed with a set of values specific to their mode of operation. During hyperparameter tuning, the training process repeats for every combination of variables. This process is computationally expensive, and more so when large-scale databases require processing (Ali *et al*., 2023). The parameter '*C*' for example has a range of 10.0 – 500.0 therefore every integer within this interval will be tested for every other variable of the other hyperparameters. The combination producing the highest accuracy can optimally balance bias and variance and prevent overfitting and underfitting. Tuning too many hyperparameters and variables can limit the learning potential of the algorithm and inadvertently cause overfitting and underfitting (Zimmermann *et al*., 2023). To resolve this and the computational cost concern, a smaller selection of hyperparameters can be configured with reduced integer ranges, then increase as needed for performance optimization.

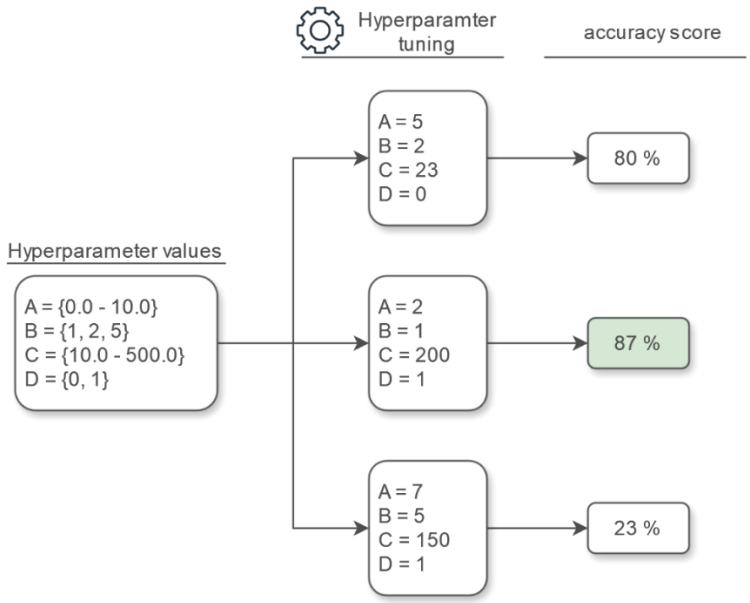


Figure 1.15 Example of hyperparameter tuning method. Hyperparameters A, B, C and D are listed with several variables that adjust the robustness of a learning algorithm. All possible combinations of variables are screened for accuracy. The best performing combination of variables for each hyperparameter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

1.6.2.2 Stratified nested k-fold cross-validation

A typical supervised learning algorithm would require one dataset to train the learning algorithm and another for classification. An issue with using different PSM datasets for training and classification, is that individually they are not representative of each other. Experimental sample preparation differs across laboratories and certain variables such as enzyme type, modified peptides and mass-spectrometry calibration is difficult to be accounted 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, and the qualities of one PSM dataset are not transferrable to classify another. This is also highly likely to cause overfitting. An alternative solution to bypass this as proposed in the works of Percolator (Granholm *et al.*, 2012), is to use *cross-validation*.

The term cross-validation was coined by Stone (1974) who identified the evolution of Monte Carlo predictive subsampling 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 a training, validation and testing set (Bai *et al.*,

2021). A learning model is trained to learn the underlying patterns of the dataset using the training sample, then the validation portion assess the quality of its classification performance before the testing sample is finally classified. Typically, the proportion of testing subset is 10 – 30% of the dataset (Berrar, 2018). This approach is not feasible in processing PSMs as it would lead to a direct loss of the unclassified spectra held out for the training and validation samples. A simple solution is to use the modern account of cross-validation which has been redeveloped for regression analysis models (Picard and Cook, 1984). Classification tasks primary adopt the *k-fold cross-validation* method (Xu and Goodacre, 2018), and **Figure 1.16** is provided to explain 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 removed to be used for testing and the remaining folds $\{k - 1\}$, are used to train a learning algorithm. This process is repeated for k iterations, until all folds are classified.

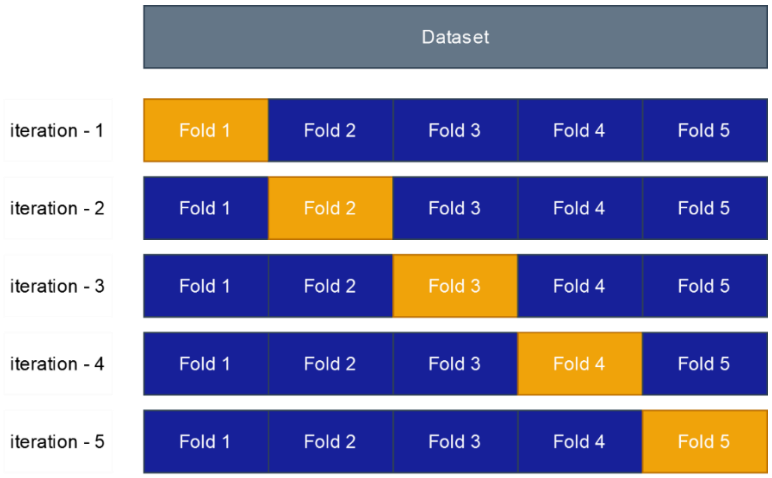


Figure 1.16 Cross-validation method. A dataset undergoes 5-fold cross-validation and is divided into five folds. The process repeats for five iterations. In each iteration the blue folds are used for training a learning algorithm and the orange folds are held out for classification.

With this approach, the learning model becomes a self-trainer (see Chapter 1.5, **Table 1.2**) and the dataset is classified by its own family of data points. The number of k-folds can range from 2 to n, where n is the 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). Generally, 10-fold (Kohavi, 1995) or leave-one-out (n – 1) (Gronau and Wagenmakers, 2019) is suggested in this case. A lesser number of 3 – 5 folds are sufficient for larger datasets e.g. PSM datasets with over a million spectra. 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. there would be 2 data points per fold for a dataset consisting of 10 data points split into 5-folds. In the case of PSM datasets comprised of target and decoy spectra with uneven frequencies, it is necessary to ensure that each fold receives a proportionate distribution to avoid bias classification. To achieve this, data points with different class types undergo *stratification*. The process of the method is illustrated in **Figure 1.17** in which nine data points are split into three folds, with three data points each in a 3-fold cross-validation setup. The two different class types with varying frequencies are divided equally into each fold.

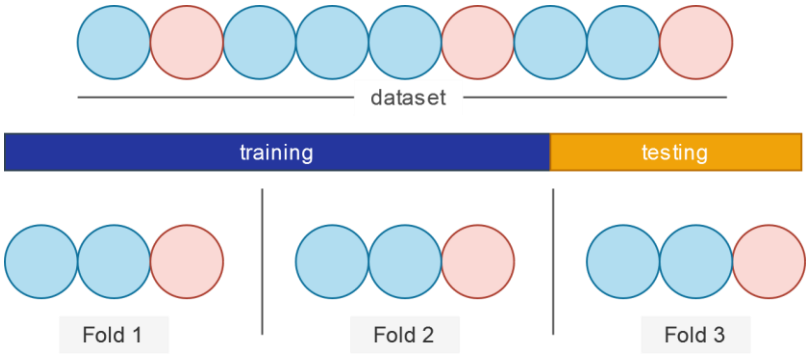


Figure 1.17 Stratification in k-fold cross-validation. An imbalanced dataset consisting of nine data point class types are stratified in a 3-fold cross-validation setup. Each fold receives one of class type 1 (red) and two of class type 2 (blue) to create an equal divide.

The best combination of hyperparameters (see Chapter 1.6.2.1) influences a model to improve its learning accuracy, however, it is acknowledged that this is an overly optimistic estimation. Errors 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 instead conduct hyperparameter tuning on a smaller sample of the dataset in a method known as *nested k-fold cross-validation* (Rakhshani *et al.*, 2015). The method of 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 process is directed to the training folds $\{k-1\}$ of the first level cross-validation. The number of folds for either level of cross-validation does not have to correspond to each other. In this example a *nested 5-fold cross-validation* scheme is created simply for uniformity.

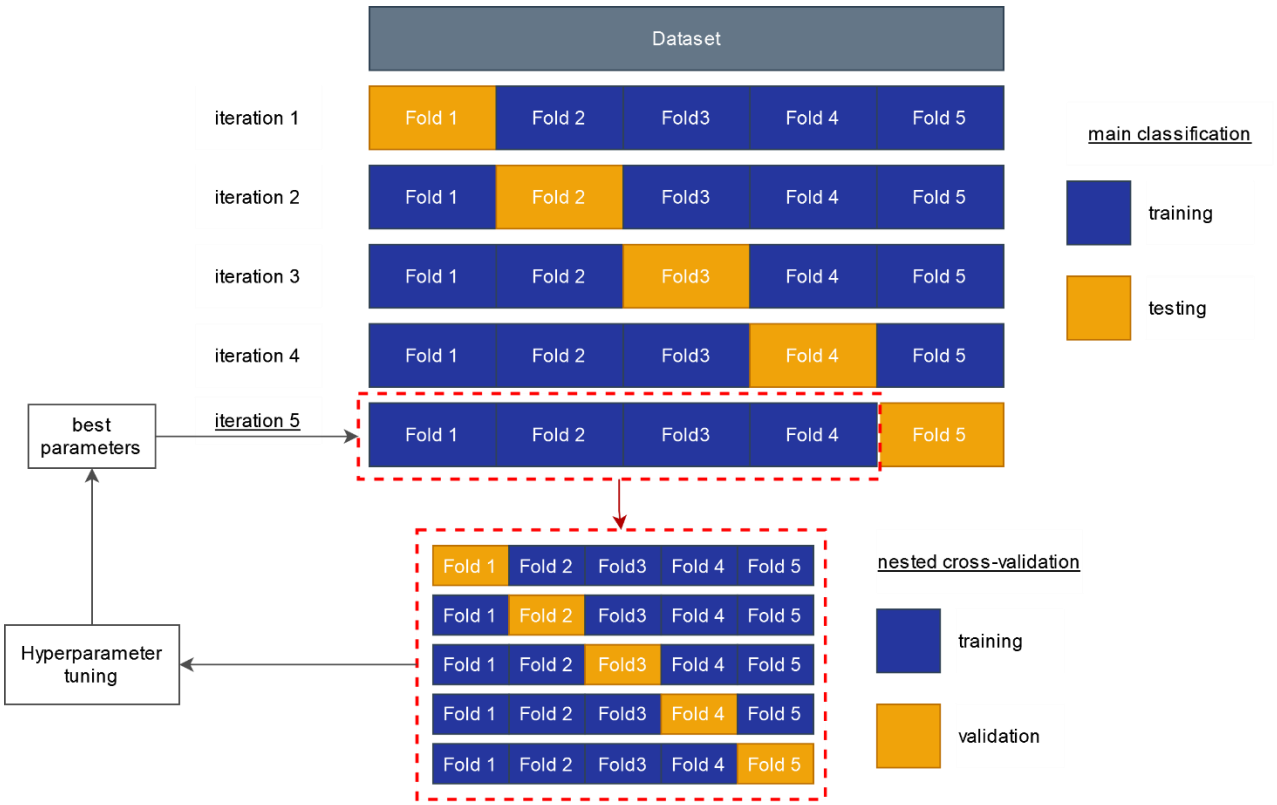


Figure 1.18 Nested k-fold cross-validation method. Two layers of cross-validation is created. The second layer is created using the training folds of the first layer. Hyperparameter tuning is performed in the second layer cross-validation. The best parameters used to train the learning model of the first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

Hyperparameter tuning is conducted for each iteration of the nested cross-validation. The training folds are divided into 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 levels training folds where the main classification and performance evaluation takes place. Nested cross-validation repeats for each iteration of the first cross-validation. The best combination of hyperparameters therefore corresponds only to the training folds of each iteration i.e. the hyperparameter tuning of iteration-1 will not carry over to iteration-2.

Although hyperparameter tuning can aid a model to be a good-fit, accuracy scores alone are not sufficient 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 is likely to be a class type. If 8 out of 10 data points were correctly predicted as a class type, the overall accuracy score would be 80%. Another model classifying the same data points can obtain a score of 85% and so on. Whether individual data points receive a 51% or 95% probability score of being a certain class types, by majority (i.e. a decision threshold over 50%)

they would both be assigned the predicted class label during classification. They are, hence, uninformative and do not provide context to the likelihood data points are correct or incorrect.

Consider if one data point receives a probability of 51% in one model and 76% in another. A global accuracy score cannot justify, nor confidently ensure which classifier produces the 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 in classification tasks. The TDA (see Chapter 1.5.1) can effectively be used to infer error rate metrics onto 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 inspect 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 learning model's predicted labels of the classified dataset. This can be specifically tailored towards PSM processing. The goal of PSM processing is to identify random spectra from the target PSMs (see Chapter 1.5). The predicted labels therefore are not of interest and hereon the process will be described using the target and decoy spectra.

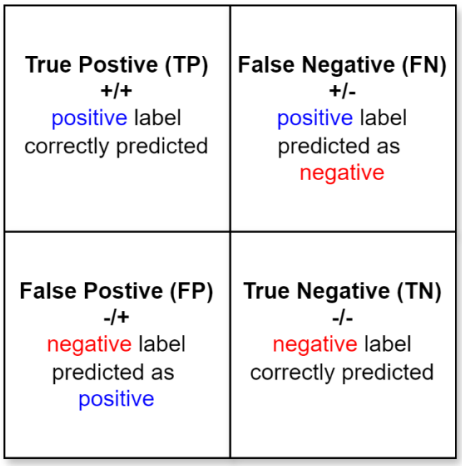


Figure 1.19 Confusion matrix. Binary class labels positive and negative are distributed into four statistical outcomes following classification.

Similar to the FDR and PEP, confusion matrix estimates factor in the benefit of ranking PSMs (**Figure 1.11**) by a probability of fitness. The confusion matrix definitions of **Figure 1.19** can be translated as follows: the target spectra are the *positive* class type, and the decoy spectra are the *negative* class type. The test statistic assigned to the classified spectra by a learning algorithm is used to rank the PSMs. This is demonstrated with probability scores in **Figure 1.20**. Then instead of using predicted outcomes, the frequency of target and decoy spectra occurring at the rank of each spectrum is tallied. The TP, FP, TN and FN rates are computed using the appropriate estimation strategies. The confusion matrix rates are generally 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 a note here is that the calculations visualised in **Figure 1.20** are designed for the current project and will be referenced later in the following Chapter 2.8.2. After the spectra have been assigned the four confusion matrix rates, further metrics prescribed for model performance evaluation can then be computed.

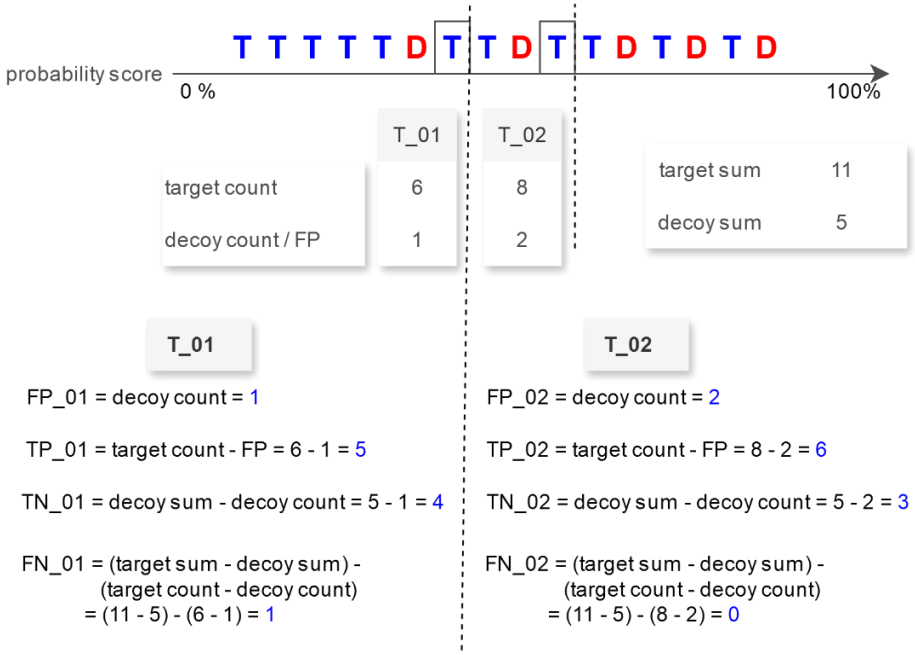


Figure 1.20 Confusion matrix concept simplified.A classified PSM dataset is ranked by probability scores. The confusion matrix rates are computed for two target spectra (blue 'T's in the outlined boxes) at different rank locations. The target count and decoy count are the number of spectra occurring up until the spectrums rank position indicated by the dashed line. The target sum and decoy sum are the total number of class types for the complete dataset. The false positive, true positive, true negative and false negative rates are estimated by inserting the target and decoy measurements using the equations.

1.6.3.2 Model performance metrics

The error rates and trends of the distribution of data points can be visualised through graphical representation to empirically observe a learning algorithms performance. There are several methods to achieve this, two of which are the: area under the receiver operator characteristic curve (AUROC) and the precision-recall curve (PR). The receiver operating characteristic (ROC) curve is the standard graphical illustration applied in statistics to analytically evaluate the performance of binary classification. Originally created for war fare to discern real enemy fire and false alarms (D. M. Green, 1970; P. E. Green, 1970), this method has quickly found its way into medical diagnostic research since the 1970s (Lusted, 1971). The ROC curve compares the trade-off of the true positive rate (TPR) and false positive rate (FPR) (Huang and Ling, 2005). The TPR, also known as *sensitivity* and *recall*, measures the positive instances (TPs) against incorrectly classified negative (FNs). The FPR, also known as *1 - specificity*, measures the positive instances estimated as a negative class (FP) and correct estimations of the negative instances (TNs) (Van Stralen *et al*., 2009). Here, the method is described for PSM processing, and the positive and negative instances are interpreted as target PSMs and decoy PSMs, respectively.

This evaluation estimated for each spectrum can be comprehensively visualised using the ROC curve (**Figure 1.21**). A performance curve close to the random classifier line, would indicate the model has poor ability to separate class types. The area under the curve (AUC) is an accuracy measurement of the ROC. The closer a curve is to the upper left coordinates $(0, 1)$ the higher the AUC score, indicating excellent performance (Nahm, 2022). The AUC score is perceived as being better than the accuracy score issued by a learning algorithm (**Figure 1.15**) due to the ranking process considering the likelihood of spectra being a true match (Ling *et al*., 2003). The reliability of true matches estimated by a model can be ensured by a good performing model measuring the error rates in the same manner. Combining AUC and ROC, the AUROC curve can be implemented to compare performance of numerous models.

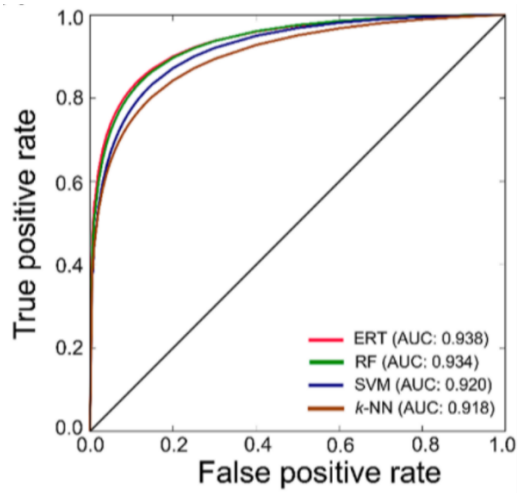


Figure 1.21 Comparison of architectures using the AUROC curve. The ROC curve is the trade-off of True positive rate and False positive rate. Four ROC curves for different Machine Learning architectures are represented by identifying colours. Their performance is compared using their AUC scores. The black diagonal line represents a random classifier. This unedited illustration is from Manavalan *et al*. (2018).

The AUROC curve is prone to overlook class imbalances, as its focus is on the learning model's ability to separate positive and negative class types, regardless of the proportion of their frequencies (Saito and Rehmsmeier, 2015). Consider for example, an FP target spectrum at rank position 8 with 7 TN decoy spectra ranked above incurring a low FPR. Negative classes are expected to be at lower ranked for any test statistic, and this poor performance would not be considered by the AUROC. This is possible to occur when processing large-scale PSM dataset with an expected increase of negative cases (Saito and Rehmsmeier, 2015). To ensure the outcome of an AUROC, it is coupled with a PR curve. The PR curve similarly measures *sensitivity* renamed *recall*, against a precision score that is a measurement of the positive instances (TPs) against the likelihood it is a FP (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 is a mirror image of the ROC, meaning, the higher the curve is to the top right (top-left for an ROC plot) at (x, y) coordinate $(1, 1)$ the better the performance of the model. In this example, the two algorithms appear to perform well in the ROC curve, yet both have poor precision in the PR curve. Both metrics are computed for classified ranked PSMs, therefore the correlation between precision and accuracy can be directly inferred. Generally, an accuracy score similar to the AUC, is accompanied with the PR curve (Sofaer *et al*., 2019), however, to establish model performance observation of trends is sufficient.

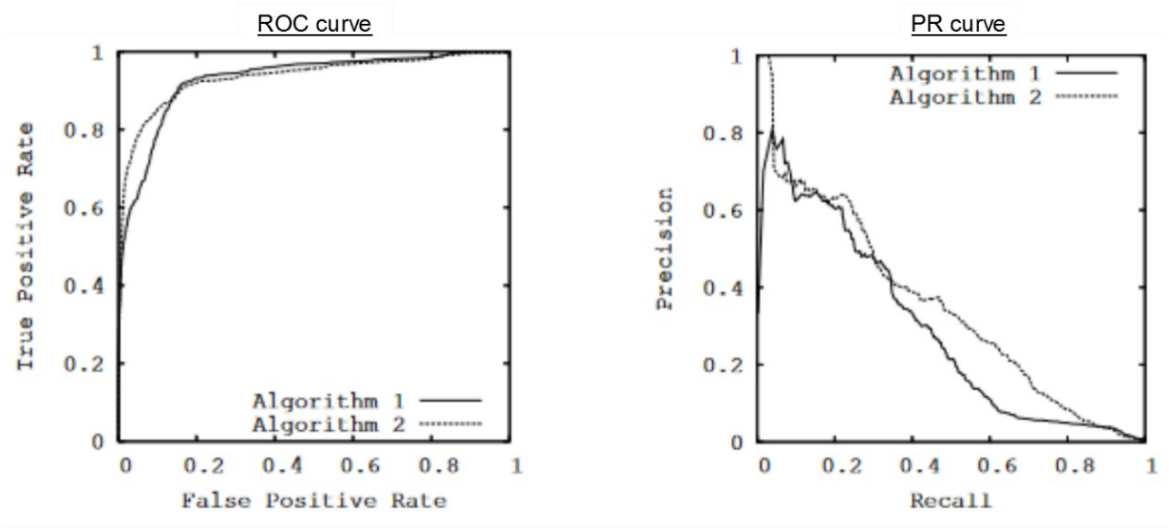


Figure 1.22 Relationship between a ROC and PR curve. The performance of two algorithms is observed in each graph. The PR curve displays poor precision as a means to compare to their good performing ROC curves. This unedited illustration is from Davis and Goadrich (2006).

The AUROC and PR curve has been applied widely in biological studies whose methodologies include statistical inference. A study by De Smet *et al*. (2004) identified an increase in FNs in the case of large-scale biopsies concerning leukaemia research and explored balancing error rates using ROC curves. Computational identification of rare anti-coronavirus peptides from imbalances datasets using AUROC and PR metrics for classification performance was examined by Pang *et al*. (2021). Regarding PSMs ranked by a probability metric indicating fitness, the ROC and PR curve are indicative of spectra behaviour. It is expected that a higher frequency of correct target PSMs would incur strong probability scores with a high TPR and low FPR. An increasing FPR would be synonymous with target PSMs having a strong likelihood of being an incorrect match. The curve of the ROC should then follow the same appearance as a well-performing model (**Figure 1.21**) which would be validated by the PR curve. In this chapter, the method of processing PSMs using ML was introduced. In the next chapter, existing tools and methods will be discussed.

1.7 Machine learning in practice as a post-processing tool

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 database size (see Chapter 1.4). Current methods have been created to filter out poor quality spectra at either the PSM-level or peptide level. In **Figure 1.23**, a PSM dataset consisting of target and decoy spectra are used to annotate MS/MS peptide sequences.

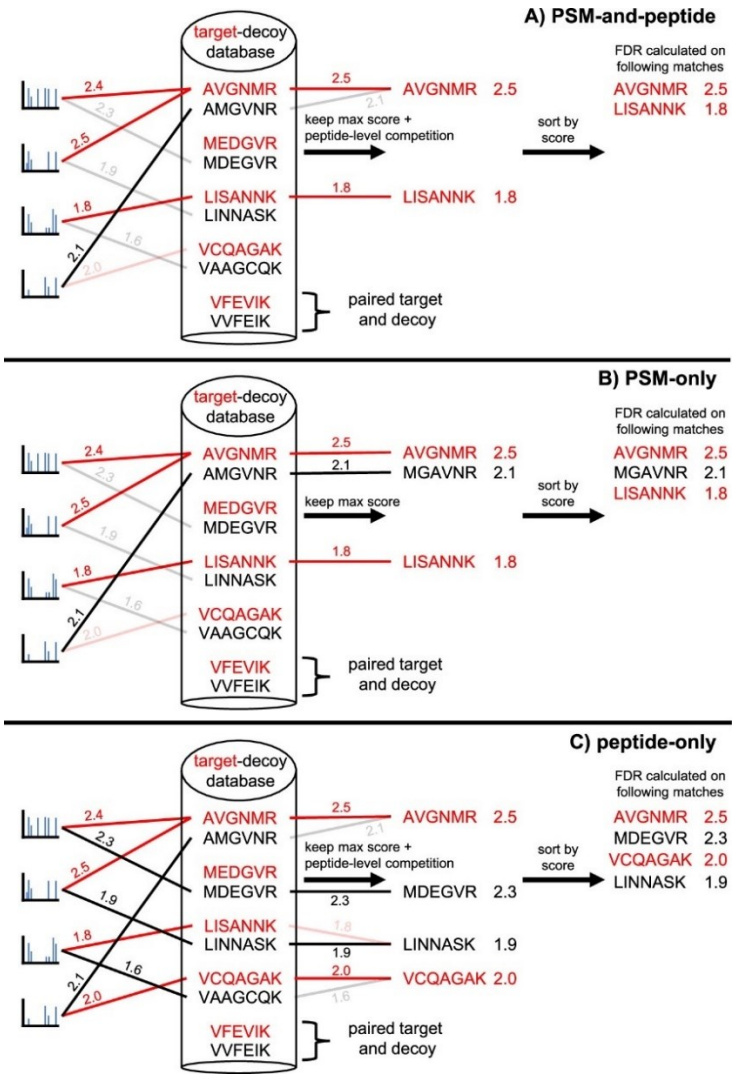


Figure 1.23. Difference between PSM level and peptide level processing. Mass spectrometry spectra can undergo error rate performance evaluation at three stages. **(A)** PSM and peptide **(B)** PSM only and **(C)** Peptide only. This unedited illustration is from Lin *et al*. (2022).

The PSM-level (**Figure 1.23[B]**) obtains the overall highest scoring alignment to either the target or decoy spectra, during search engine processing. A well-known existing PSM-level tool is DTASelect (Tabb *et al*., 2002) and there are more recent ventures such as Crema (Lin *et al*., 2024) that measure 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 well-known and obtain the highest scoring target and decoy spectra alignment for each observed spectrum which then the best scoring alignments are retained. Peptide-level filtering ignores the relationship between target and decoy spectra of 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 control the nuance of implying significance for peptide identification tasks. Alternatively, it has been proclaimed that a combination of PSM and peptide processing (**Figure 1.23[A]**) is the most effective approach to produce significant amounts of identified peptides. Research on improving FDR application on *E.coli* peptides from the work of Lin *et al*. (2022) reported a performance increase over 40% in the jointed method over the PSM only approach. Existing PSM processing tools such as Scaffold (Searle, 2010) and IDPicker (Ma *et al*., 2009) explore the potential of extending analysis from PSM level to protein validation.

Although these tools are well-established, ML explores the tactic of combining the multiple scoring functions of search engines into a single interpretable test statistic. Early ML tools such as PeptideProphet (Keller *et al*., 2002) built a statistical model using Bayesian inference to rank annotated peptides and assess their validity. An SVM learning architecture was adapted by Anderson *et al*. (2002), to discriminate between incorrect and correctly identified peptides matches derived from SEQUEST. The incorrect peptides sequences were created experimentally and selected from low scoring annotations. Later, decoy PSMs were created 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 built to discriminate target and decoy spectra. In its methodology, the annotated target and decoy spectra are first refined using 20 scoring functions (**Figure 1.24**) collected from SEQUEST (Anderson *et al.*, 2002) and other statistical measurements. The workflow of Percolator (**Figure 1.25**) begins with it ranking the combined target and decoy PSMs by 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 which is trained to discriminate between the selected target spectra and the full-set decoy PSMs. Nested 3-fold cross-validation hyperparameter (see Chapter 1.6.2.2) tuning is conducted during the learning process. The testing folds (**Figure 1.16**) are normalized and merged by the SVMs 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 unedited image of the twenty scoring functions is supplied in the supplementary documents of Percolator (Käll *et al.*, 2007).

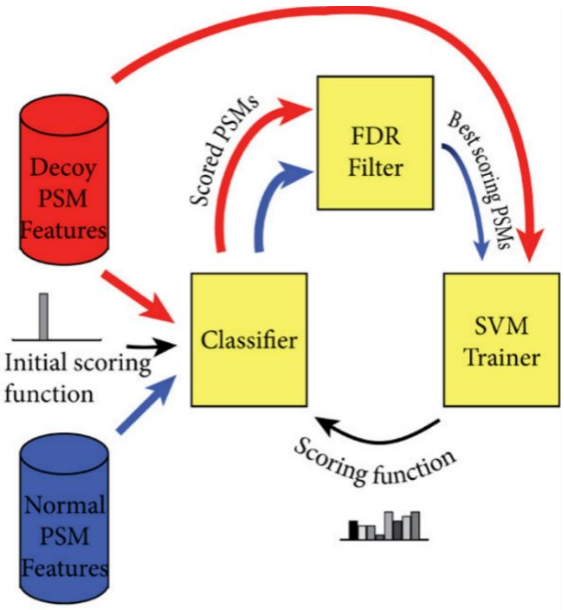


Figure 1.25 Learning algorithm of Percolator. The workflow of Percolator consists of an iterative loop between three key elements. The process starts with a PSM dataset of target and decoys being ranked by an initial scoring function in the *Classifier*. An *FDR filter* is used to extract target spectra for the *SVM trainer*. This image is modified from Halloran *et al*. (2019).

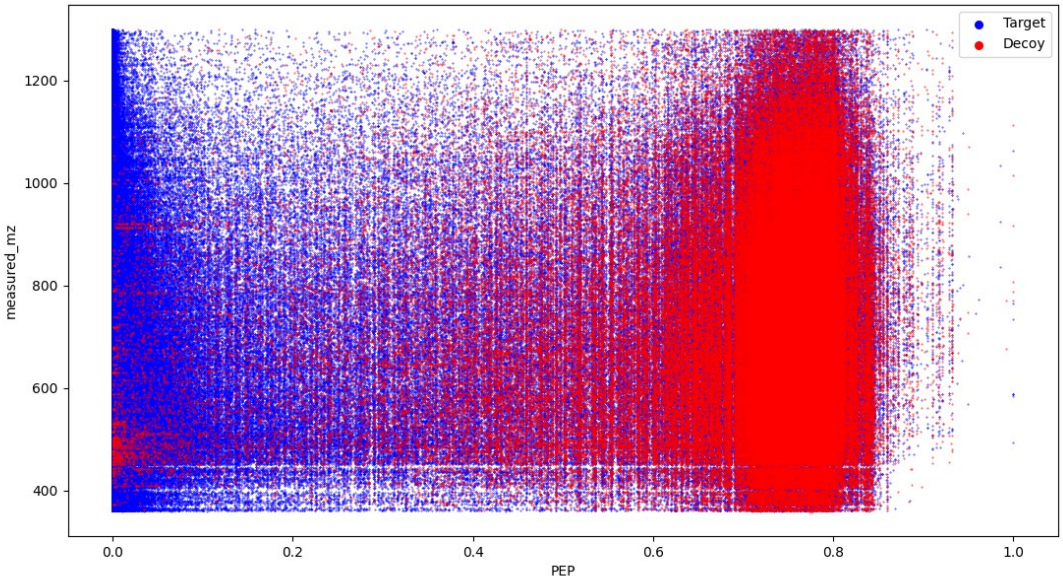


Figure 1.26 Performance evaluation of Percolator. Target and decoy spectra were classified using Percolator. Two of the scoring functions *measure_mz* and *PEP* were used to observe the model's ability to separate the spectra.

This process repeats until there is no change in the ranking. The final product is the target and decoy spectra classified with p-score estimates and further metrics such the PEP and q-values are included for downstream validation. A preliminary execution of Percolator was conducted during this research to demonstrate its classification performance . This figure observes an accumulation of target spectra with significant PEP scores with a low frequency of decoy spectra amongst its population. Although there is a clear separation of target and decoy spectra, it is questioned whether there is room for classification improvement. This work will not make a direct comparison to Percolator or other existing methods and instead aim to explore the nature of PSM processing to improve variant PSM classification resolution.

1.7.2 Machine Learning for variant PSMs

While the existing PSM processing tools are well-established they are not specifically equipped to identify mismatches in variant peptides. Variant peptides are highly similar to their canonical counterparts and occur at a lower frequency, making 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 post-translational markers of variant peptides during MS/MS research. Generally, these methods specifically generate variant peptides sequences through experimentation then annotate with variant peptide genome databases to improve identification.

Here in this work, alternative machine learning architectures will be explored in their ability to separate target and decoy PSMs of variant peptides at the PSM level. Specifically, the performance of modern decision-tree ensemble machine learning architectures is investigated in their ability to process PSMs which then the better performing model 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 core idea of ensemble models is the 'wisdom-of-the-crowd' ideology, meaning a group of individually trained learning algorithms are more likely to produce an accurate prediction (Alizadeh *et al*., 2015; Elliott and Anderson, 2023). An elementary illustration of this is provided in **Figure 1.27**. The learning algorithm workflow (**Figure**

1.27[a]) is first trained to learn the patterns and features of labelled data points. Unlabelled data points are classified using the ensemble model (**Figure 1.27[b]**). In this diagram, data points are classified by three individual base-learners that produce different predictions. The ensemble model collects these decisions to confirm the majority prediction and produce the final classification of the data point, completing the workflow of **Figure 1.27[a]**.

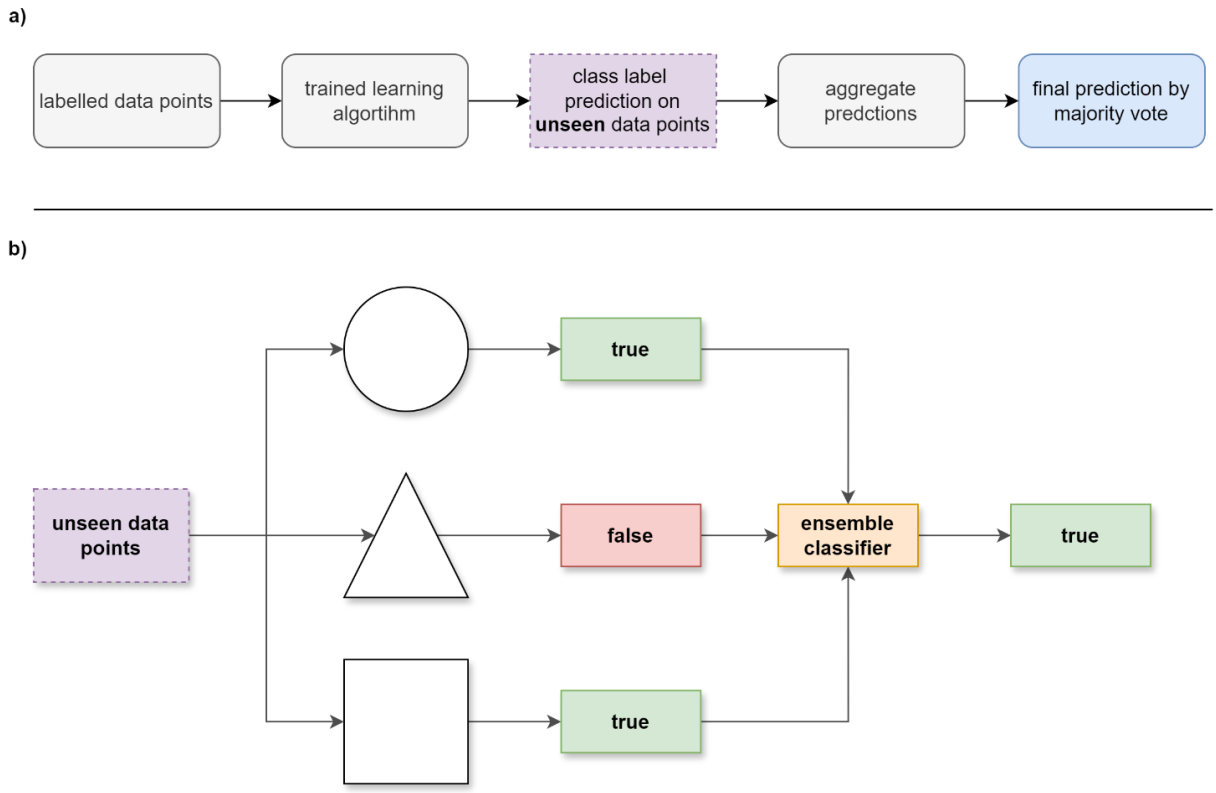


Figure 1.27 Ensemble learning algorithm methodology. (a) Stages of the ensemble classification workflow.**(b)** An unlabelled data point is classified by individual three individual base learners represented by the different shapes. Each learner produces a prediction. The ensemble classifier aggregates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

1.8.1.1 Improving 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 obtains an accuracy score above 0.5. If a data point receives a score of 0.51 or 0.85, for example, it would receive the positive class label. A *weak learner* produces a prediction accuracy score just above a random guess of 0.5 (Joshi *et al*., 2002). *Strong learners* produce near accurate predictions but are generally difficult to obtain and have a likelihood of being incorrect (Galton, 1907). To improve the

performance of ensemble models the *Boosting* method, established by Kearns (1994) is employed. This tactic converts learners from weak to strong by improving their significance and 'boosting' their contribution to the overall base-learners prediction accuracy.

One other method to improve performance is the Bootstrapping and AGGregratING or *Bagging* method introduced by Breiman (1996). The goal of *Bagging* is to increase the independence of the multiple base-learners from the ensemble model. Prediction accuracy is more likely to increase with base-learners producing independent predictions that differ from the other learners (Zhou, 2012). The tactic of *Bagging* aims to increase randomness when training learning algorithms by using bootstrapping (Efron and Tibshirani, 1993) or out-of-bag sampling techniques. Here, data points are selected at random to train the individual base-learners which are then replaced back into the dataset.

The concept of the ensemble technique can be applied to modern machine-learning architectures such as neural networks (Rosen, 1996), SVMs (Shen and Chou, 2006) or decision trees (Geurts *et al*., 2005). Ensemble models can be created using a single architecture type (homogeneity) or as a combination of architectures (heterogeneity). Heterogeneous learners are a form of combination or 'stacking' learning style. The stacking tactic creates layers of base-learners for each architecture to then combine the strong learners from each layer to make a final prediction (Zhou, 2012). In this work, homogeneous decision-tree ensemble learning models will be investigated for processing PSMs.

1.8.2 Decision trees

The decision tree (Quinlan, 1987) architecture learning method is structured as a conditional flow-diagram or *tree* consisting of a series of *nodes*, *branches* and *leaves*. The decision tree structure (**Figure 1.28[a]**) starts from a *root node* which *branches* into *internal nodes* that *branches* into *leaf* nodes. The *root* and *internal nodes* pose conditional questions, and the *leaves* represent possible decisions or predictions. Consider a PSM dataset consisting of data points with class labels (x) and feature variables (y) undergoing classification. A decision tree classifier system (**Figure 1.28[b]**) creates conditional nodes formatted to the datasets feature variables. The flow from *root* to *leaf node* represents the *decision path* for a particular data point, guiding it through a series of choices until a prediction is reached. PSM data points are classified as either a target or decoy spectrum based on the *decision-path*. In this example, two question *nodes* are created, however, *tree* depth can be extensive to increase verbosity of the decision-making process. Attributed 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.

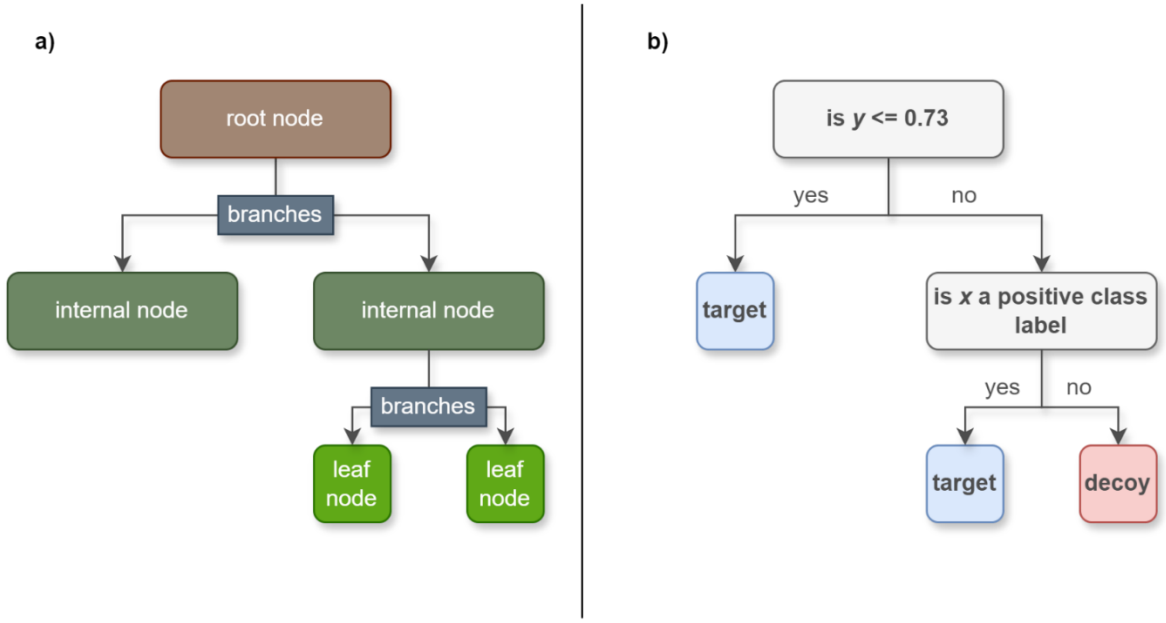


Figure 1.28 Decision tree classification method. (a) Structure of a decision tree. A decision-making path starts from the root node and follows a route of nodes until a prediction is made in a leaf node (Zhou, 2012). **(b) A decision tree processing PSMs.** Spectra are classified as either a target or decoy class types following a decision path of nodes conditionally describing scoring functions.

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), ExtraTrees (Guerts *et al*., 2006) and XGBoost (Chen and Guestrin, 2016). These architectures will be compared in this work for their ability to process PSMs. Each of these architectures can perform supervised and semi-supervised learning tasks concerning binary classification. In several works such as Sankari and Manimegalai (2017), Ahmad *et al*. (2021) and Krawczyk *et al*. (2014), decision tree ensemble models were determined as a practical approach in handling imbalanced datasets. In the case of PSM datasets with rare variant peptides with low frequency compared to their canonical counterparts, this may prove to be beneficial. A brief description of each architecture will be introduced next.

1.8.2.1 Random Forest and ExtraTrees

Feature importance is a metric measured by a learning algorithm which evaluates a dataset's feature variables contribution to the prediction making process. The Random Forest tree structures randomly select feature variables (Ho, 1995). Feature importance is then measured based on the frequency and effectiveness each feature creates *branches* to determine the *best-split* (i.e. decision-making path) (Breiman, 2001) whilst reducing prediction errors. An example of three decision trees in **Figure 1.29** is displayed with varying levels of *tree* growth. The decision trees of Random Forest are created independently from each other to ensure they are unrelated or *de-correlated* (Hastie *et al.*, 2008), which in effect reduces the likelihood of overfitting. ExtraTrees, also known as extremely randomized trees, is a newer ML addition that is an extension and aggressive form of Random Forest (Guerts *et al.*, 2006). Unlike the Random Forest method, ExtraTrees incorporates the entire dataset to build decision trees instead of randomly selecting feature variables (bootstrapping) (Efron and Tibshirani, 1993). Rather than computing the *best-split*, ExtraTrees sets thresholds to create *branches* when randomly selecting feature variables.

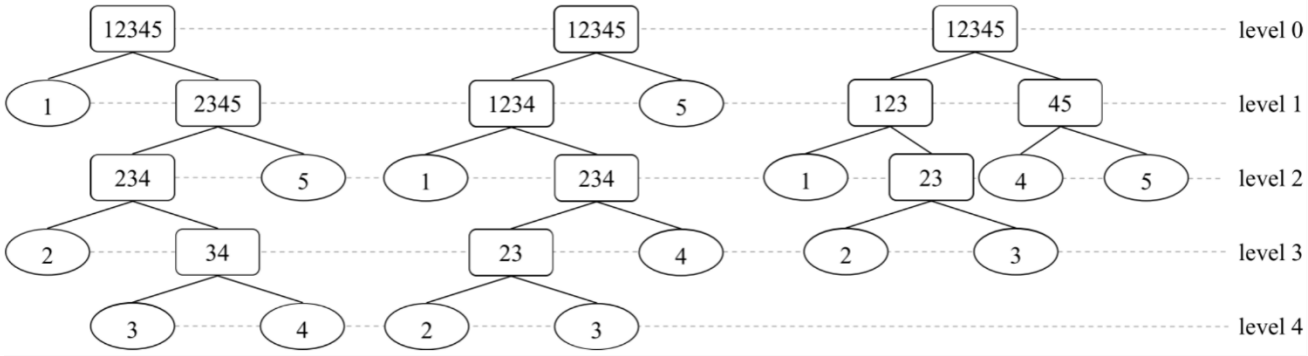


Figure 1.29 Random Forest algorithm method. Five data points are classified in three individual random decision trees. Each tree has a root and internal nodes (rounded squares) that creates layers of splits. The data points are classified when they reach a decision leaf node (oval). This unedited illustration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

Random Forest and ExtraTrees are *Bagging* (Breiman, 1996) ensemble models that create fully-grown decision trees. This means the decision trees are permitted to develop with no restrictions, imposing as many question nodes and decision-paths as it needs to produce the best prediction. Furthermore, by imposing feature selection, the decision trees do not require *pruning* (Qi, 2012), meaning that *branches* imparting limited significance are not removed so to purposefully induce randomness and fully capture patterns of the training data points.

1.8.2.2 Gradient Boosting, Histogram Gradient Boosting and XGBoost

The Gradient Boosting architecture creates an ensemble of decision trees by aggregating *weak learners* and sequentially corrects errors of base learners (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 first base learning model does not efficiently capture the patterns of the training data points. 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 metrics to reduce the errors from the previous learner. The models are combined to evaluate the overall prediction outcome. This process repeats to consecutively build decision trees with gradual improvements to the prediction accuracy. Histogram Gradient Boosting and XGBoost are variants of the Gradient Boosting method.

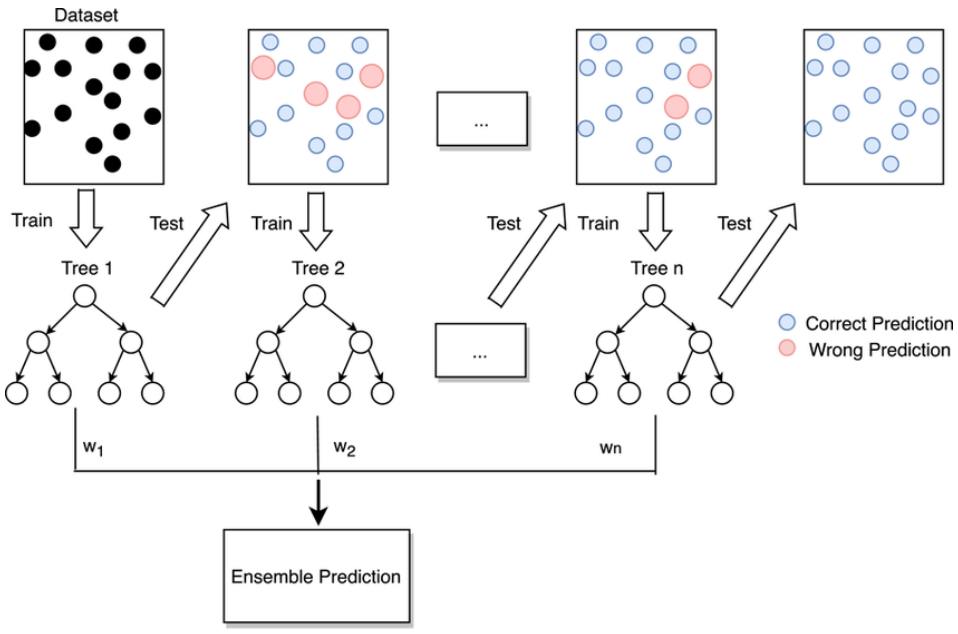


Figure 1.30 Gradient Boosting algorithm method. The process begins with a single decision tree producing a high frequency of incorrect predictions (upper left box with black circles). New decision trees are created until incorrect predictions (red circles) are reduced and correct predictions (blue circles)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

The Histogram Gradient Boosting architecture (Guolin *et al.*, 2017) compartmentalises feature variables into intervals or *bins* before training the learning algorithm. Data points are then assigned to a *bin* based on their feature value (Hossain and Deb, 2021). Decision tree *branches* are created using the *bins* to determine the *best-split* and estimate the *residuals* (errors). The Histogram Gradient Boosting architecture was developed by Microsoft as an element of their Light Gradient Boosting Machine library (Guolin *et al.*, 2017), which then gained attraction due to its ability to reduce computation time and classify larger amounts of data points.

Extreme Gradient Boosting or XGBoost is an advanced implementation of the Gradient Boosting method (Chen and Guestrin, 2016). XGBoost prevents overfitting by using regularization in which overly complex models are penalized or *pruned*. There are two forms of regularization L1 (Lasso) and L2 (Ridge) (Chen and Guestrin, 2016; Cortes *et al*., 2019; Ahmad *et al*., 2021). The Lasso metric prunes *leaf nodes* by penalizing weights with little importance. The effectiveness of irrelevant feature variables is reduced to simplify the model. The Ridge method discourages large weights to encourage *weak learners* and prevent any individual tree from overly influencing the prediction outcome.

1.9 Project scope

Detecting variant and non-variant proteins can be an effective strategy to provide definition to the heterogeneity of diabetes by evaluating protein expression and stability. Here it is proposed that investigating the performance of modern machine-learning architectures to confidently predict canonical peptides can further be 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 to then conduct statistical inference and graphically visualise 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 peptide matches. The *predictor function* and *counting function* are the main functional features serving to accomplish this objective. The classifier accepts a PSM dataset refined with multiple scoring functions and ranks them by a single function to estimate the FDR. The learning algorithms of the predictor function are trained with target spectra threshold with a 1% FDR and decoy spectra from the dataset. The main body which houses the ML architecture is the *predictor function*. Here, the learning algorithm operates as a self-trainer system by utilizing *k*-fold cross-validation to classify the full-set PSM dataset.

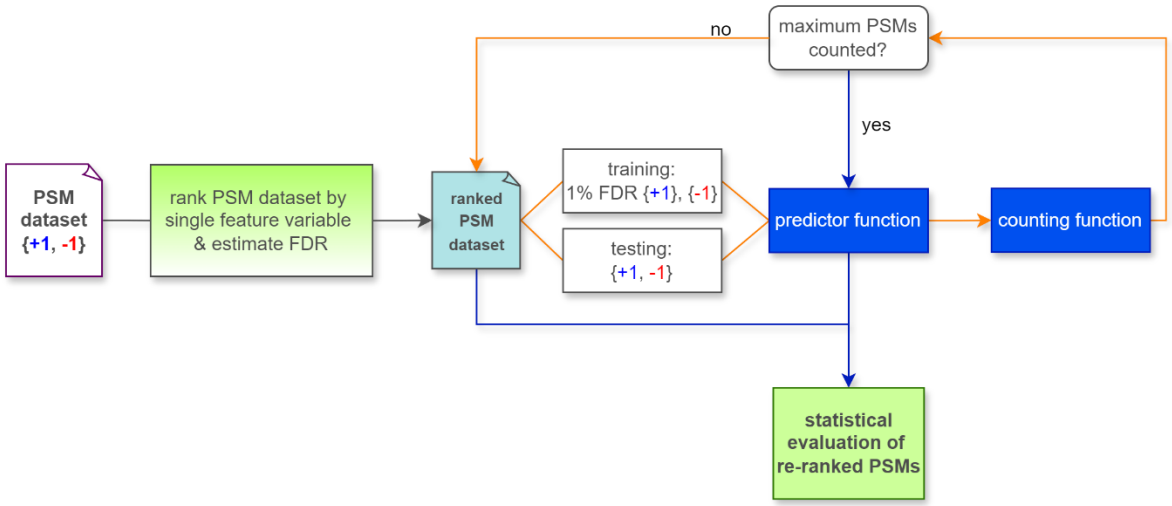


Figure 1.31 General schematic of the Nagilums Tree pipeline. A PSM dataset composed of target {+1} and decoy {-1} labelled PSMs, with feature variables is processed. The spectra are ranked in ascending order by one of these features and classified in the predictor function. The orange path line indicates the conditional loop imposed by the counting function. The blue path line is applied when the condition is met, and performance of the model is investigated for the classified dataset.

The counting function tracks the number of confidently predicted target PSMs and enacts a conditional loop between itself and the *predictor function* until the maximum number of classified spectra are observed. With each iteration the PSM dataset is re-ranked, and a new FDR is estimated to improve classification. The final ranking is 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 here the loop serves to collect a high amount of confidently predicted target PSMs.

The pipeline of NT is used to implement a workflow of three tasks (**Figure 1.32**) and statistical inference and graphical visualisation is specifically setup for each individual 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, ExtraTrees and XGBoost. The classification objective of this task follows standard PSM processing using the TDA where the target PSMs are a composite of canonical and variant peptide spectra. The better performing architecture is selected for task two.

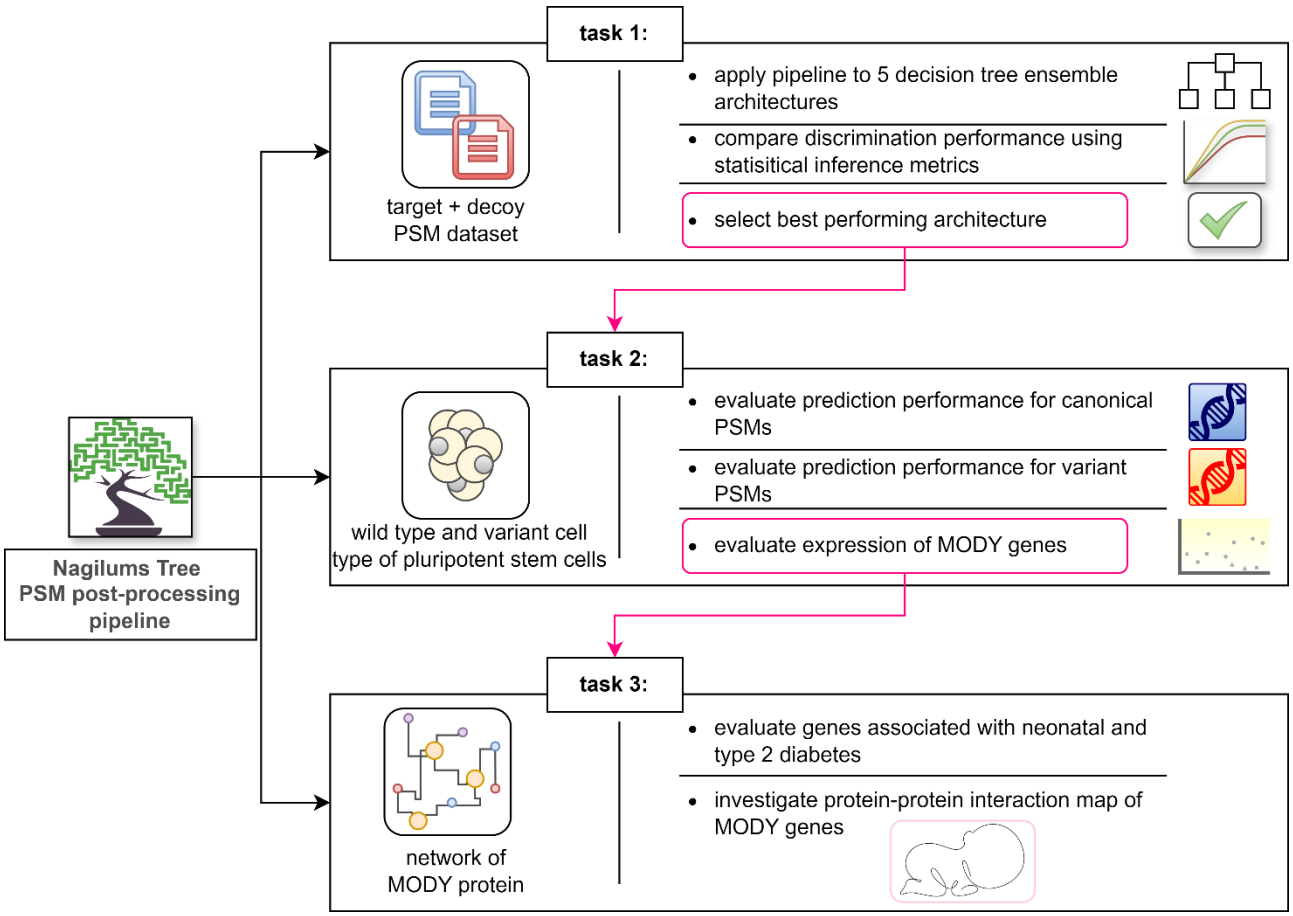


Figure 1.32 Project pipeline. The Nagilums Tree pipeline executes three consecutive tasks. Task 1 compares the discrimination performance of machine learning architectures using the target-decoy approach. Task 2 explores two search strategies to improve variant and canonical PSM classification of MODY 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 with separated class types of the canonical and variant spectra which are used individually to create *decoy canonical* PSMs and *decoy variant* PSMs. Two search strategies are investigated using either decoy PSM type against the canonical PSMs and variant PSMs labelled as target class types. This task serves to investigate the potential of using decoy variants as a null model to improve resolution of confidently predicted variant PSMs. Here, experimental files are real examples of pluripotent stem cells induced with insulin resistance on the HNF1A (MODY3) gene. The NT pipeline is adapted to account for these alterations and conduct a protein expression comparison of each decoy search strategy.

Lastly, MODY gene expression are analysed using error metrics of these two search strategies to evaluate their effectiveness. Task 3 considers evaluating the protein-protein interaction network of significant peptides for their inferred association with MODY and other diabetic conditions. Due to the extent of the current study, task 3 is marked as future works and will unfortunately not be explored here, although the step is developed. Instead, a minor evaluation of the MODY expression will be included to contribute to the discussion about potential implementation of the concepts explored in this work.

The NT pipeline was written in Python script, and all files are available on GitHub (link: Nagilums Tree pipeline). Connecting GitHub links are provided throughout the next chapters for specific Python file references. Detail of the tools and methods crafted for NT is explained in the next chapter. Descriptions of the PSM datasets, workflows, statistical tactics and performance evaluation methods are further described.

2: Materials and Methods

2.1 Flowchart of classification system

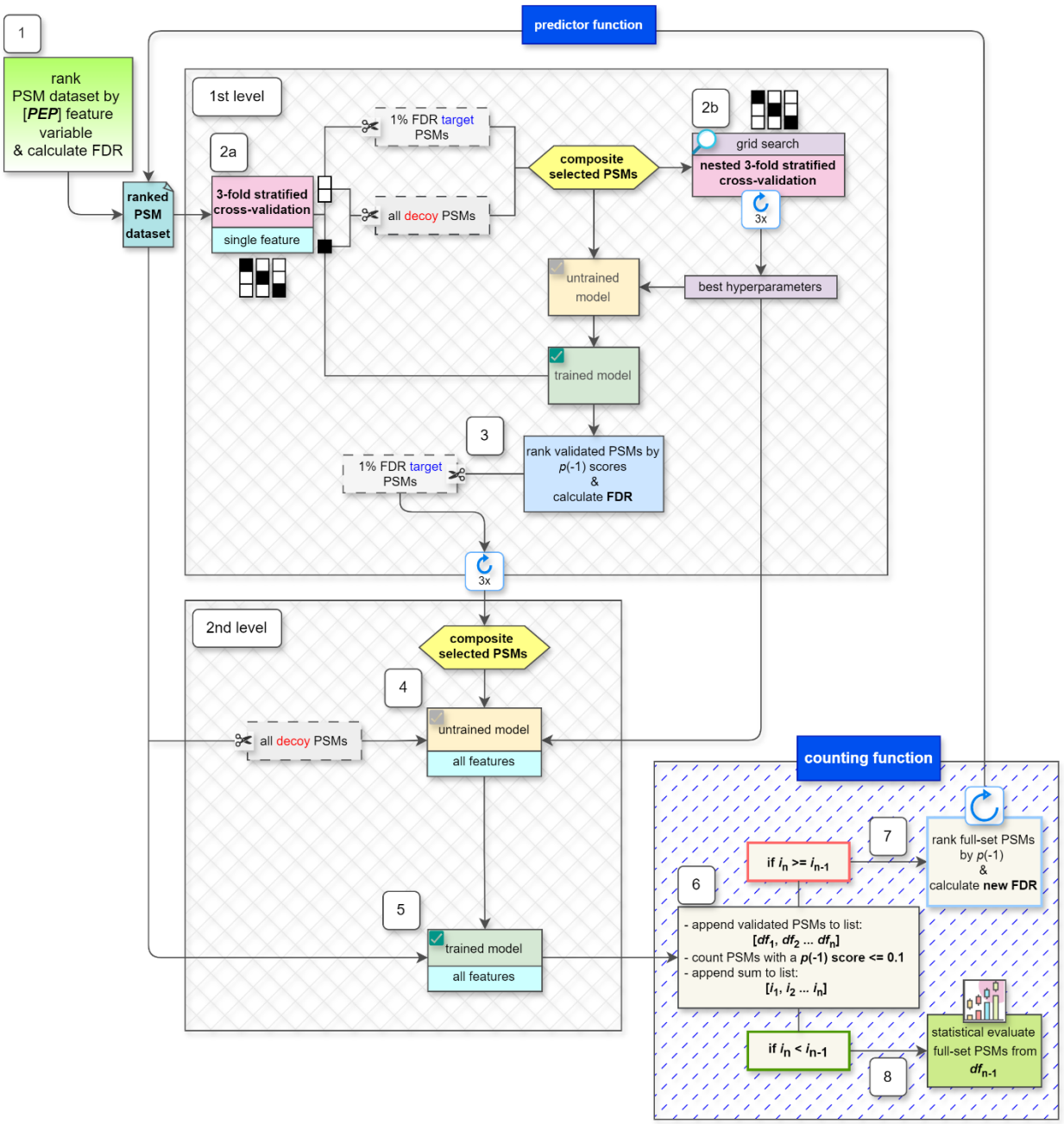


Figure 2.1 Schematic of Nagilums Tree pipeline. The workflow is divided into 8 steps to classify a PSM dataset. The *predictor function* is separated into 2 levels. The 1st level hosts the nested *k*-fold cross-validation process. The white blocks are the training folds, and the black block is the testing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

The workflow of NTs 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 classified 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 it being significant 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 levels classification process. Target spectra with a 1% FDR 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 using 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 optimization. The trained algorithm then classifies the testing folds PSMs of cross-validation from Step 2a.

Step 3: The classified PSMs of the 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. Target spectra threshold at a 1% FDR are retained. The 1st level repeats three times for each iteration of the 3-fold cross-validation process during which a new learning model is trained. After each 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. The complete set of feature variables available from the PSM dataset are selected for this training step. The best combination of hyperparameters from step 2b are 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 for the classified PSM dataset and another for 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 added to the archives. A conditional setup is issued by the counting function to track the target PSM count archive.

Step 7: If there is an increase count of the target PSMs from the previous repeat, the classified PSM dataset is ranked by its p(-1) test statistic and a new FDR is estimated. The newly ranked dataset is passed back to the predictor function and steps 2 – 6 are conducted. With each repeat of the predictor function a new p(-1) test statistic is issued by the learning algorithm, and the new FDR is expected to improve classification.

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

This pipeline is constructed to produce a PSM dataset ranked by the learning algorithm with a high frequency of confidently predicted target PSMs. The base learner created for this pipeline is used for both task 1 and task 2 (see Chapter 1.9, **Figure 1.32**). The construct 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 Python scripting language using the IDE, PyCharm version 3.10. Python is an attractive programming language for biological computing due to its ease-of-use, versatility of its vast ecosystem of third-party library packages and user-friendly nature. It is an ideal tool for data analytics and visualization. All PyCharm libraries were used at their most current version and are listed for reference in **Table 2.1**.

Table 2.1 Python libraries used in Nagilums Tree.

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 offering a diverse range of classical machine-learning architectures. After its release in 2011, Scikit-learn has quickly become a state-of-art-tool across disciplines and is actively being further developed with additional updates and improvements. Python programming is popularized due to its highly functional environment creating a user-friendly interface. These qualities are effectively utilized by Scikit-learn which is explicitly created to avoid framework code and is built 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 and creatively extend its use for general purpose. Current research such as AlphaPept (Strauss *et al*., 2024) and PeptideMind (Handler and Haynes, 2021) have recently been exploring the use of Scikit-learn in biological computing within the context of proteomics. Application of the Scikit-learn methods used in this project's pipeline will be described in this chapter. 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 cohesively integrated and operating procedures created according to the user's specification (Pedregosa *et al*., 2011). This feature is essential to the completion of task 1, in which the 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-learns 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 block code in **Figure 2.2**. Here the base estimator is introduced in context of PSM dataset processing. The basic setup 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_titles = [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_titles]          : create label and feature samples
8:  training_labels = labels = np.array(training_sample['Label'])
9:
10: testing_features = testing_sample[feature_titles]
11: testing_labels = labels = np.array(testing_sample['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-learns 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 setup up for processing. 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 an unseen sample of the dataset being classified by a fitted model using the feature weights only from which class labels are inferred based on its prediction. Separate datasets can be used for the training and testing process or as visualised here in **Figure 2.2**; the single dataset can be portioned into individual samples. The feature variables and labels are then 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**) for flexibility when processing 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 testing procedure is a means to conduct an unbiased evaluation to the performance of the model as it was not used in the training procedure. The trained model is tested with the *testing_features* (**Figure 2.2,** line15) by using the *.predict* function.

Beyond classification, the learning models confidence regarding classification of the testing sample can be assessed. The *.predict_proba* (**Figure 2.2,** line16) method in Scikit-learn produces the probability estimates of the class type for each classified data point. Two scores are returned 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) or 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 or spectra sums up to 1.0 (100 %) and is a distribution scaled from 0.0 – 1.0, representing a low to high degree of confidence respectively.

Table 2.2 An example of 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

To infer significance to classified spectra, the p(-1) scores are solely used in this work. The reasoning of including decoy PSMs is to infer the null model of statistical hypothesis testing (see Chapter 1.5.2) concerning 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. The classification process is not intended to be biased (see Chapter 1.6.1). Decoy spectra are reasonably expected to occur throughout the probability range although at higher frequencies toward a 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. 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 class labels of the PSM dataset corresponds as: target {+1} and decoy {-1}. The rows of spectra are organised by columns describing the peptide sequence, label, 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*, respectively, from the *pandas* library (**Table 2.1**). The dataset is ranked by using a Python method called *.sort_values()*. To rank spectra by the *pep* scoring function would appear as so: *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 Nagilums Tree workflow

2.5.1 Stratified k-fold cross-validation

The base learner of the predictor function is constructed as a self-training algorithm. The central ideology of self-training (see Chapter 1.6) concerns reusing a dataset for the training and testing process of a base learner. This is typically accomplished 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 peptide matches, 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.

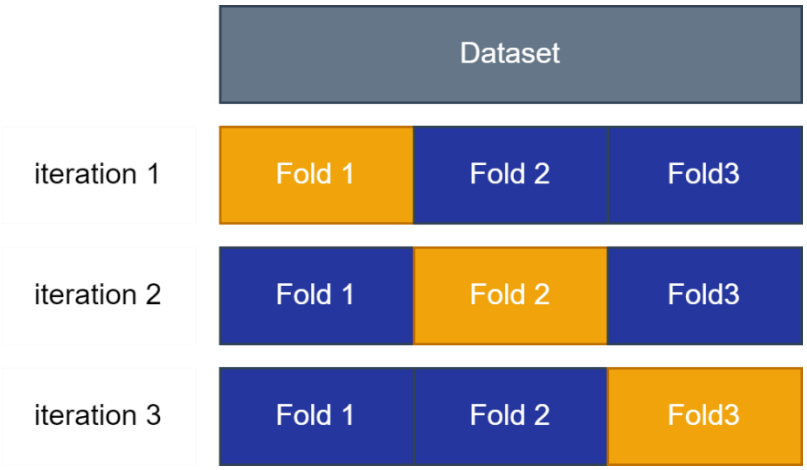


Figure 2.3 Method of 3-fold cross-validation. A dataset is divided into three equal subsets known as folds. The two blue folds are used for training a learning model. The single orange fold is classified. This process iterates three times to classify the complete dataset.

The block code of the stratified cross-validation process created for the predictor function is provided in **Figure 2.4[a]**. The *StratifiedKFold* method derived from the Scikit-learn library (**Table 2.1**) is the configuration setup for the: number of splits (*folds*) for cross-validation, robustness of random data point selection and whether or not to shuffle the dataset. Here, *n_splits* is set to 3 corresponding with 3-fold cross-validation and the spectra are shuffled to be randomly sorted into *folds*. The *pep* scoring function from the PSM dataset is selected as the single feature (**Figure 2.1**) variable for cross-validation. Individual variables are created to refine the PSM sequence identifications by the *pep* variable and the class labels (**Table 2.3**).

a)

```
StratifiedKFold(n_splits=3, random_state=1, shuffle=True)                    : setup
input_data = PSM input file
single feature = 'pep'
dataset_feature = input_date[single_feature]                                 : separate input file into feature and labels
dataset_labels = input_data['Label']

for fold, (training_index, validation_index) in \                            : stratified cross-validation loop
    enumerate(stratifiedKFold, split(dataset_features, dataset_labels):

    training_folds_01 = input_data[training_index]                           : input file separate into training and validation folds
    validation_fold_01 = input_data[validation_index]

    training_feature_01 = training_folds_01[single_feature]                  : base estimator setup
    training_labels_01 = np.array(training_folds_01['Label'])
    testing_feature_01 = validation_fold_01[single_feature]
    testing_labels_01 = np.array(validation_fold_01['Label'])

    psms_within_fdr_threshold = training_folds[training_folds['FDR'] <= 0.01]
    target_psms_within_fdr_threshold =
                    psms_within_fdr_threshold[psms_within_fdr_threshold['Label'] == 1]   : composite 1% FDR target PSMs from training folds
    decoy_psms = psm_input_file[psm_input_file['Label'] == -1]                           : and decoy PSMs from dataset
    composite_target_decoy_spectra =
                    pd.concat([target_psms_within_fdr_threshold, decoy_psms])

    training_features_02 = composite_target_decoy_spectra[single_feature]
    training_labels_02  = composite_target_decoy_spectra[labels]

    for fold, (training_index,_02 validation_index_02) in \                  : nested stratified cross-validation loop
        enumerate(stratifiedKFold, split(training_features_02, training_labels_02):

        training_fold_02 = training_fold_01[training_index]                  : training folds separated into training and validation folds
        validation_fold_02 = training_fold_01[validation_index]
        training_features_02 = training_folds_02[single_feature]             : hyperparameter tuning with training fold
        training_labels_02 = np.array(training_folds_02['Label'])

        base estimator continues ...                                         : GridSearch base estimator
```

b)

```
        grid_parameters = {'learning_rate': =[1,5, 10], 'n_estimators'=[100, 200], \   : dictionary of parameters of interest
                           'max_depth'=[3, 5]}
        hyperparameter_tuning = GridSearchCV(estimator=ml_model(warm_start=False), \   : GridSearch setup
                                             param_grid=grid_parameters)
        hyperparameter_tuning.fit(training_features_02, training_labels_02)            : base estimator
        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_combination_overall = ['learning_rate'=1, 'n_estimators'=200, 'max_depth'=3]
```

Figure 2.4 Block code of predictor function self-trainer structure. The layout of the scripting environment for k-fold cross-validation method is described (left-align) with their basic usage descriptions (right align). **(a) 3-fold cross-validation**. Cross-validation is setup

using the *StratifiedKFold* method. Two levels of cross-validation are created to process a PSM dataset for hyperparameter tuning. The process to threshold target PSMs and collect decoy PMS for training a learning algorithm is described. **(b) Hyperparameter tuning**. The *GridSearchCV* method is used to repeatedly retrain a learning model with all possible combinations of parameter variables. *warm_start=False* enables a new model to be fit with each repeat. The best combination of parameters is return 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.

To enact cross-validation, a *for loop* is created in Python script. The three iterations required is tracked by the *enumerate* function which incorporates the *StratifiedKFold* setup application to the class labels (**Figure 2.4**, line 6). In each iteration, the dataset is separated into the training $\{k-1\}$ and testing $\{k-(k-1)\}$ folds automatically using the *StratifiedKFold* method. The divided spectra are recorded by their index values (**Table 2.3**). Within the loop, the indices are then used to extract the appropriate spectra directly from the dataset (**Figure 2.4[a],** line 9 and 10) to create the *training_fold_01* and *validation_fold_01* variables. Note *validation_fold* refers to the testing process; here is it used interchangeably to track the cross-validation levels.

In lines 12 – 15 of **Figure 2.4[a]**, the *training_folds_01* and *validation_fold_01* are prepared for a base estimator by diving them into their appropriate feature and label variables. The learning algorithm of the base estimator can then be *fit* with the dataset sample of *training_fold_01* with the selected combination of hyperparameters (**Figure 2.2**, line 13). The *validation_fold_01* is then classified. After each iteration of cross-validation, the classified testing folds are ranked by their p(-1) scores to collect 1% FDR target spectra (**Figure 2.1**, Step 3) using the same process as lines 17 – 19 in **Figure 2.4[a]**. The full-set decoy PSMs are selected using the same process as line 20 in **Figure 2.4[a]** and are combined with the 1% FDR target PSMs to be prepared for a new base estimator. The main classification is initiated by using this composite of target spectra with the full-set decoy PSMs to train a new learning algorithm (**Figure 2.1**, Step 4). The full-set PSM dataset is then classified accordingly (**Figure 2.1**, Step 5).

The *nested 3-fold cross-validation* method performed next to conduct hyperparameter tuning (**Figure 2.1**, Step 2b). In the NT pipeline the training process is conducted with,1% FDR target PSMs from the training folds and the full-set decoy PSMs from the dataset (**Figure 2.4[a],** lines 17 – 22). The target PSMs are extracted by their class label {+1} and an FDR of 0.01 (1%). These two subset dataframes are combined using *pd.concat*. Individual variables are created for the

composite spectra refined to single feature and labels (**Figure 2.4[a],** lines 21 and 22). The composite spectra then undergo hyperparameter tuning using the same *StratifiedKFold* setup and the training_labels_02 are stratified (**Figure 2.4[a],** line 24 and 25). A second level of variables for the, *training_fold_02*, and *validation_fold_02* are created. Again, individual variables, *training_features_02* and *training_labels_02*, are created for the *training_fold_02*.

2.5.1.1 Hyperparameter tuning

The decision-tree ensemble architectures provided by Scikit-learn are built with internal parameters that are unchangeable and specific to the learning algorithm. Hyperparameter tuning configures certain characteristics attributed of decision-tree structures (see Chapter 1.8.2, **Figure 1.28**) such as number of *nodes*, *branches*, *leaves* and *trees* to control robustness of the classification assignment. In Scikit-learn, parameters are set at default generalized 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 have 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 selected are a mixture of the defaults and alternatives for each parameter. The descriptions of the parameters are directly from 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. It is possible to increase the variable range in future works.

The *stratified k-fold cross-validation* method is continued in **Figure 2.4[b]** to describe hyperparameter tuning. A variable is created for the dictionary list of hyperparameters and variables concerning the decision-tree ensemble architecture (**Figure 2.4[b]**, line 23). Hyperparameter tuning (**Figure 2.1**, Step 2b) is implemented using Scikit-learns *GridSearchCV* method. A variable, *hyperparameter_tuning* is created to setup the configuration arguments by entering the ML architecture employed under *estimator* and the hyperparameter dictionary as *param_grid*. To ensure a new learning algorithm is created for each combination of

hyperparameters the *warm_start* argument is set to *False*. Then the *GridSearchCV* model is executed by following the base estimator (**Figure 2.2**) method.

The best combination of hyperparameter variables is isolated using the *best_params* function (**Figure 2.4**, line 28). 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 parameter is reflective of the training subsets learning algorithm, a generalised approach is used in NT to proceed with the most common occurring variables from the combination collection. Therefore, one out of the three combinations for *testing_fold_02*, that will be used to complete training process of cross-validation (**Figure 2.4[c], Figure 2.1** Step 2a). One out of the nine combinations for *testing_fold_01* are selected for the main classification (**Figure 2.1**, Step 4).

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 growth of tree, branching nodes. More depth can increase overestimation**
B. **Maximum number of nodes per tree. Limits complexity and reduce overfitting**
C. **Minimum number of data points per leaf node.**
D. **Number of feature variables considered for best split. Controls randomness**
E. **Controls the contribution of each tree. Lower values increase robustness, requires 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**

2.6 FDR implementation method

To estimate the FDR for a classified testing dataset, the method derived from the original works of Percolator (see Chapter 1.5.3) is used in this work. This method is suited for the target-decoy approach and assumes that the number of decoy PSMs is equal to the number of false positives of the target PSMs within an FDR threshold. Although this FDR equation was intentionally created for p-score evaluation in Percolator, here **Figure 2.5** demonstrates how its application is suited for *prediction probability p(-1)* scores. Classified spectra are ranked in ascending order {0.0 – 1.0} by their p(-1) scores. Then for each spectrum, the frequency of target and decoy spectra occurring till its rank position is used to estimate the FDR. To estimate the FDR, the p(-1) scores are rounded to 6 decimal places.

In the argument of significant digits, Cole (2015) stipulates test statistics require no more than 2, 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 lesser significant digits produced biased performance due the 1% FDR threshold. Therefore, to retain variability, 6 decimal places were the lowest number of digits observed to successfully complete classification for each ML architecture.

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

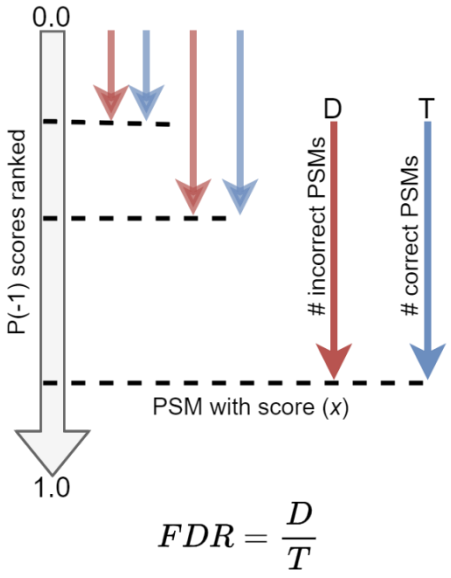


Figure 2.5. FDR estimation method. Classified spectra are sorted in ascending order by their *prediction probability* p(-1) scores and going down the list, the correct PSMs (T for target PSMs) and incorrect PSMs (D for decoy PSMs) for each spectrum threshold are counted. The accumulated counts are used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

2.7 The counting function

The main objective of NT is to classify a PSM dataset and to obtain the maximum number of confidently predicted canonical peptide sequences. The PSM dataset is classified in the *predictor function* and is ranked by the *prediction probability* p(-1) scores. Target spectra obtaining a p(-1) score between 0.0 – 0.1 signify high prediction confidence. The block code of the *counting function* is provided in **Figure 2.6** to explain the process.

Outside of the function in line 24, the main setup for the NT pipeline is configured. Archives (**Figure 2.1**, step 6) for the target spectra count and dataset are created. The PSM dataset is ranked by the single feature variable '*pep*' using the *.sort_values(by=['pep'])* method. The FDR is computed by executing the *FDR_calculation* (see Chapter 2.6). The NT workflow begins by activating the *counting function* with prepared inputs. The *predictor function* is activated within the *counting function* and classifies the ranked PSM dataset. Target spectra denoted by a class label of {+1} and a probability distribution of $\{p \leq 0.1\}$ are isolated as

confidently_predicted_target_psms in line 4 of **Figure 2.6**. The spectra count is measured by length of the refined dataset using the *len* method. Note 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 ranked 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 = [0]
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 counting function. The workflow of the scripting environment for the counting function is described (left-align) with their basic usage descriptions (right align). The main setup process for the PSM dataset input file includes creation of the *dataset_archive* (line 25) and *target_spectra_count_archive* (line 24). These parameters activate the NT pipeline through the counting function initiating setup (line 29). The predictor function is activated (line 2) with the setup parameters passed from the counting function. The conditional loop (line 11) is setup to assess the classified PSM dataset for the number of confidently predicted target spectra. The confusion matrix is activated (line 21) for the final classification of the dataset.

The ranked PSM dataset and the confident target spectra count are archived (**Figure 2.1**, step 6). An example of the first entry into these archives would appear as follows:

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

The counting function enacts a conditional assessment (**Figure 2.6** , line 11) to ascertain if the total count has increased or decreased from that of a previous classification. Archives for the target spectra count and dataset 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 activated again. With each repeat the spectra are issued new $p(-1)$ scores. The dataset is re-ranked by these scores and enters the counting function to be assessed. If there is an increase again, the FDR will be estimated on the re-ranked dataset and this loop process (**Figure 2.1**, step 7) continues as such:

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

The ensemble learning models are trained with target spectra threshold at a 1% FDR and with each repeat of the predictor function, classification is expected to improve prediction of true matches. The loop between the predictor and counting function ends (**Figure 2.6**, line 16) when there is a drop in the next target spectra count. The ranked dataset producing the most confidently predicted target spectra from the previous entry (indicated by the * in the example below) is selected for performance evaluation:

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

The *prediction probability* scores are required for assessment and for this reason the ranked datasets are archived so that statistical evaluation (**Figure 2.1**, step 8) can be performed on the specific PSM ranking producing the highest target spectra count. The ranked PSM dataset to be evaluated is prepared by removing redundant columns that would interfere with metric estimation for downstream analysis.

In this script the *confusion_matrix* is activated for the final ranked dataset (**Figure 2.6**, line 21) however this is only necessary for implementation task 1 which concerns evaluating ML model performance. This is removed for task 2 as only the PSM ranking is required for analysis, however it can easily be added if necessary for future works. The strategies of task 1 and task 2 are discussed next.

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, ExtraTrees, 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 in explained in reference to this figure. All Python files for task 1 are available on GitHub at: Task 1: Comparison of ML architectures.

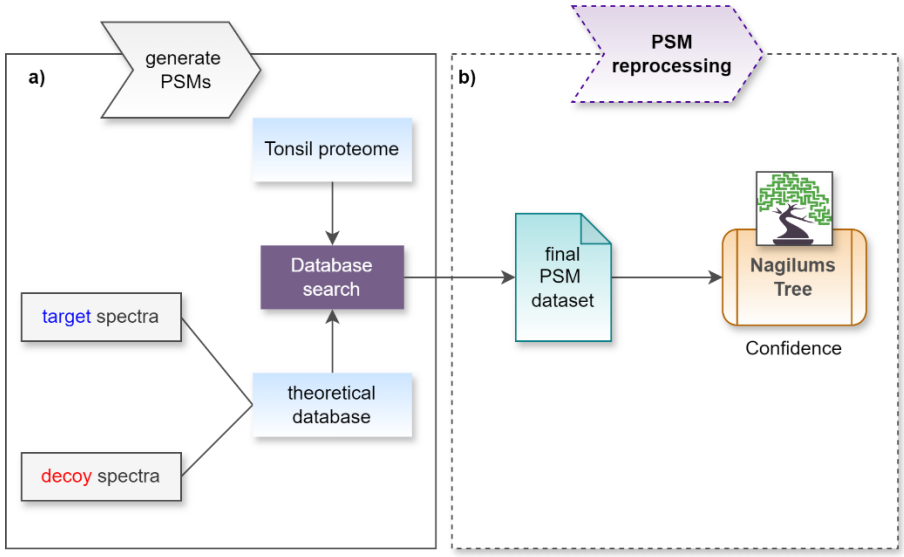


Figure 2.7 Proteogenomic pipeline of task 1. (a) The tonsil proteome dataset is annotated with a theoretical dataset composed of target and decoy spectra. PSMs are produced after a database search. **(b)** The final PSM dataset is reprocessed using Nagilums Tree.

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) was 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, were obtained from Ensembl (Harrison *et al*., 2024). Their respective decoy sequences were 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 like 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 are intended to undergo a final reprocessing using the NT pipeline (**Figure 2.7[b]**) to evaluate the discriminative performance of the decision-tree architectures. Each architecture individually processed the PSM dataset using the NT pipeline (**Figure 2.1**) and produced 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. To evaluate the discriminative power of the architectures, statistical inference is conducted using the confusion matrix.

2.8.1.2 Confusion matrix method

The final step of NT is statistical inference on the processed PSMs ranked by their *prediction probability p*(-1) scores. In this task, these ranked spectra are investigated using the confusion matrix which 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 of the PSM dataset 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 the FPs are the same as the decoy count.

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 are included in **Table 2.6**. The script is setup as a function called *confusion_matrix* and is available on GitHub at: Calculations. The function is imported into the Python file of each architectures 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 created to contain all the necessary graphical functions and visualise the performance metrics. The GitHub link: Graphical visualisation is a Python file setup to import the final *.csv* files of the classified PSM dataset of each decision-tree ensemble model and enacts the *plots_and_figures* script.

When computing the FDR, the *prediction probability* $p(-1)$ scores are adjusted to 6 decimal places (see Chapter 2.6). It is unnecessary to adjust the calculated metrics in this task due to the graphical displays used to monitor performance. The plots created follow the model performance techniques of Chapter 1.6.3.2, with additional figures to display spectrum distribution using histograms and evaluate the FDR with cumulative count plots. However, to improve visualisation resolution, the *prediction probability* $p(-1)$ and FDR estimates are rounded down to 3 significant digits in the *graphical_visualisation* file. All figures are included in the results chapter of this work.

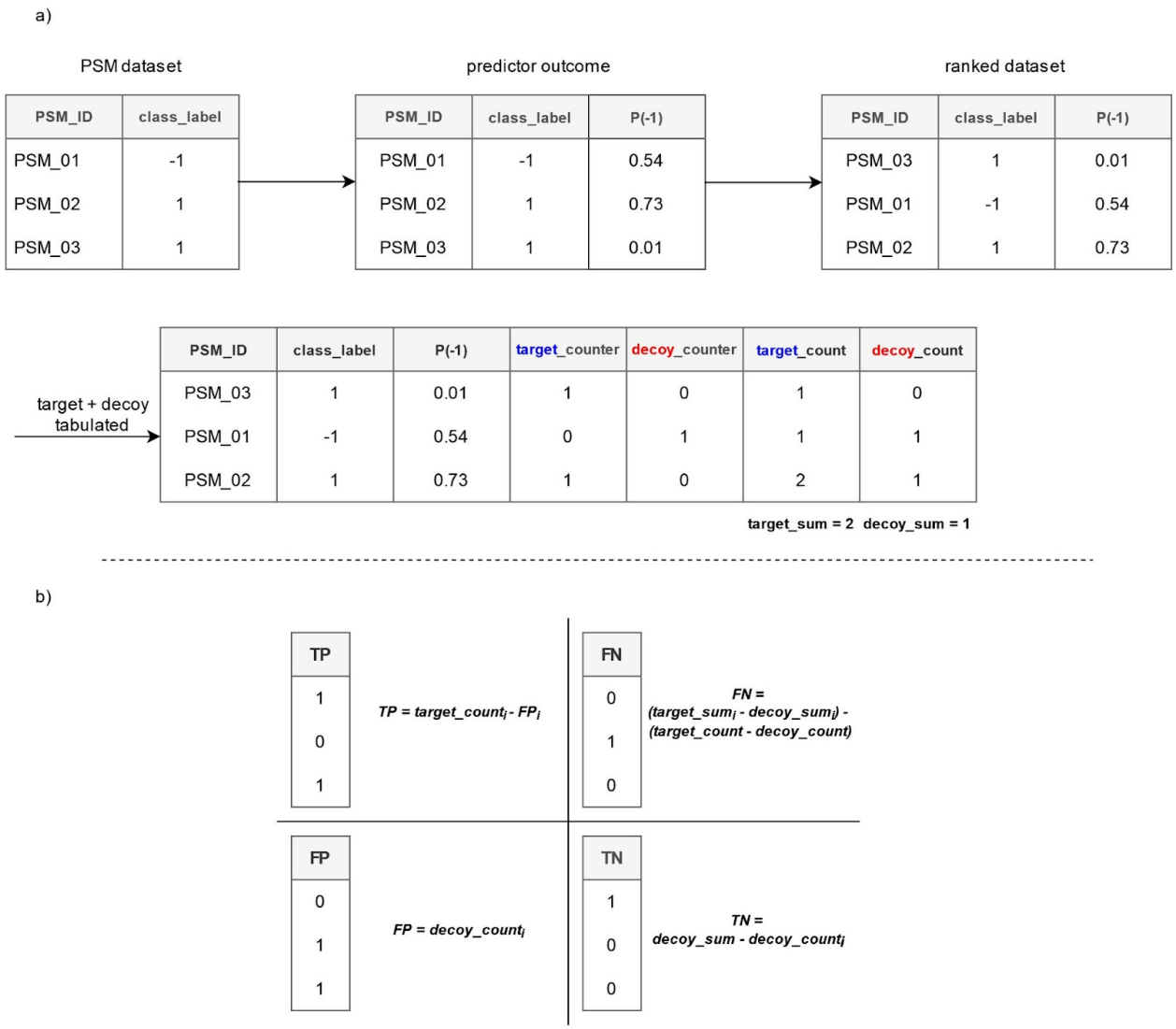


Figure 2.8 Creating the confusion matrix. An example dataset consisting of two target PSMs and one decoy PSM is processed. The *target_counter* and *decoy_counter* record the class type of each spectrum. The *target_count* and *decoy_count* tally the frequency of each class type. The *target_sum* and *decoy_sum* are the number of target and decoy spectra within the dataset. **(a) Tabulating class types.** A stepwise example on how the classified PSM dataset is ranked and how it applies to tabulating the frequency of the target PSMs and decoy PSMs of each spectrum. **(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 using these formulas for each spectrum.

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 ability or inability to detect 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 ability to avoid false positive	A high specificity indicates a low rate of false positives.
Precision	$\frac{TP}{TP + FP}$	Evaluates 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 means 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 and improve prediction confidence of variant PSMs. The basis for this task is made possible by the setup of the PSM dataset, which is processed differently to the standard TDA as in the methods of task 1 from the previous chapter. Instead of 2 PSM class types, here 4 are created: *canonical*, *decoy sequence*, *variant* and *decoy variant*. Processing of these class types requires for the NT classification process to be altered slightly 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.

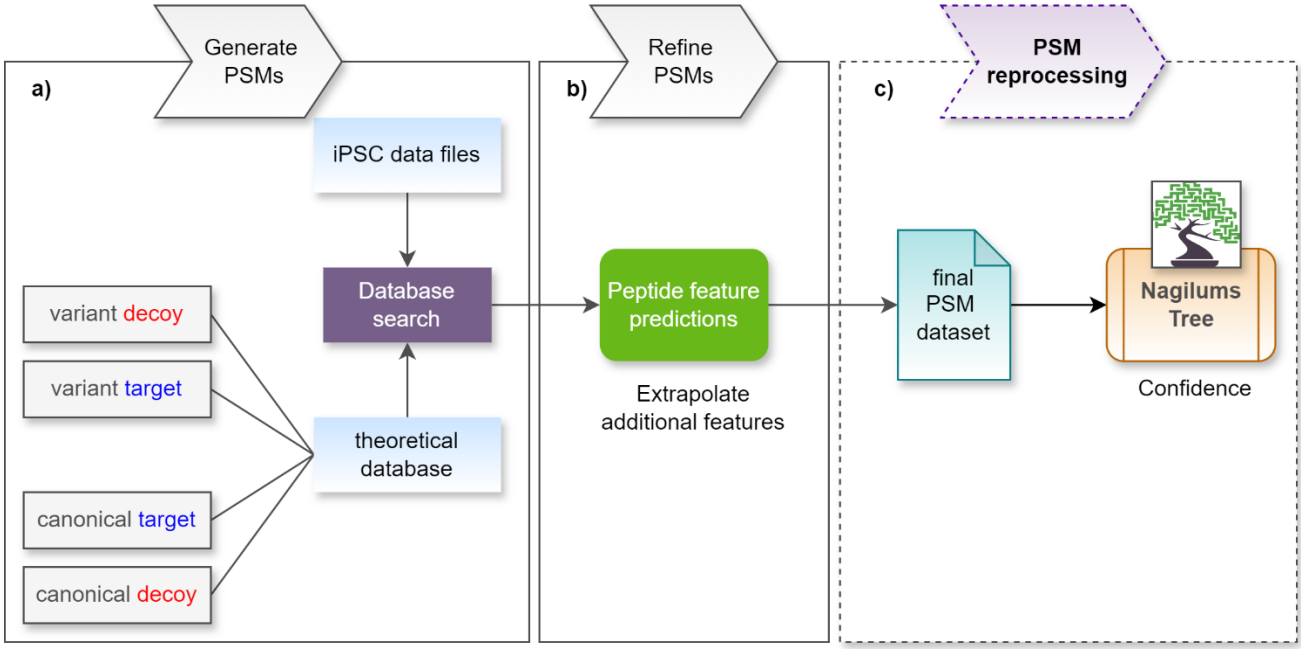


Figure 2.9 Proteogenomic pipeline of task 2. (a) Generate PSMs. PSM dataset files are created from induced pluripotent stem cell (iPSC) experimental proteome sequences and a theoretical database setup with 4 PSM class types. **(b) Refine PSMs.** Additional scoring functions are extracted and are included in the annotated iPSC PSM files. **(c) PSM reprocessing.** 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 the nature questions the likelihood of a point mutation occurring on a known variant peptide (**Figure 2.10**).

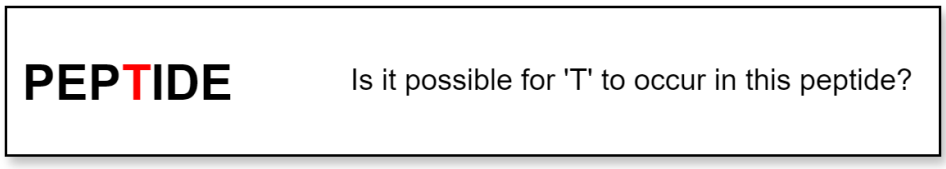


Figure 2.10 Explanation of decoy variant search strategy. The word PEPTIDE is used as an example of a peptide sequence in which the individual letters represent amino acids. The red 'T' serves to indicate a variant in the sequence.

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 to experimentational studies which has made them attractive 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 derived from either embryonic cells or induced 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 with 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 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). There were three cell-lines 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 the Dr. M. Wierer Proteomic Research Infrastructure from 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]**). The list of the 41 scoring features used in this task are listed in **Table 2.7**. The 18 scoring features from X!Tandem are listed, and the remaining were extrapolated based on peptide feature predictors in the experimental research of Dafni Skiadopoulou for the improvement of PSM classification resolution. 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 other 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 were reprocessed using NT that had been adapted to include these scoring features and conduct two search strategies visualised in **Figure 2.11**. The PSM class types of the datasets 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 referring to decoy canonical PSMs). In both strategies the *canonical* and *target_seq_target_var* are given a positive class label as the target PSMs. The method of the first strategy omits the *target_seq_decoy_var* and the learning model discriminates between the target PSMs and the negative labelled *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.

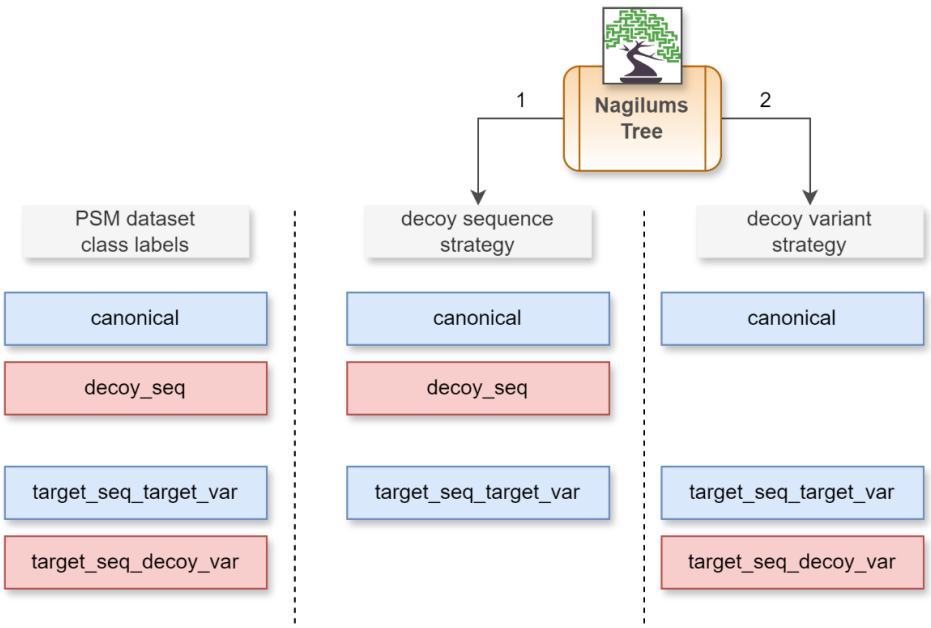


Figure 2.11 Description of the two search strategies of task 2. The blue boxes indicate the PSM type labelled as a target PSM. The red boxes indicate the PSM type labelled as a decoy PSM. The *decoy sequence* strategy removes the *target_seq_decoy_var* class type. The *decoy variant* strategy removes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

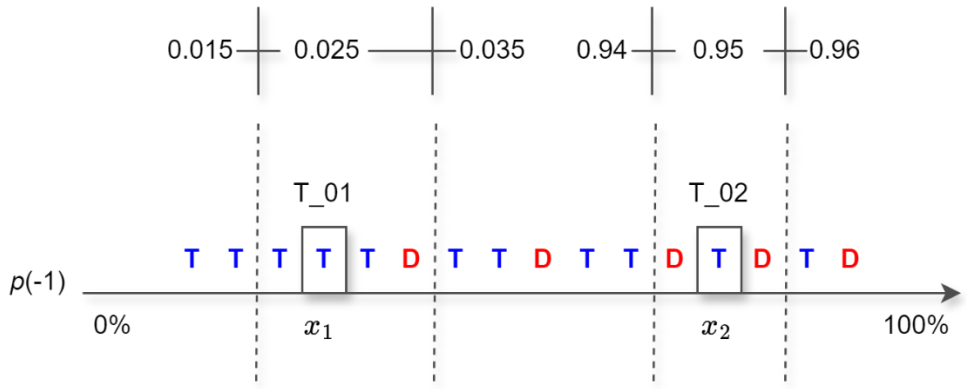
These two strategies evaluate the performance of the *decoy sequences* PSMs and *decoy variant* PSMs as a null model The Calculations Python file, contains the FDR function and the confusion matrix which is included as a formality but will not be used for evaluation. Instead, the distribution of canonical PSMs and variant PSMs are compared for each search strategy using histograms. The Python file on GitHub: Plots_and_figures contains the necessary functions to obtain all figures.

2.8.2.5 Posterior error probability estimation

To evaluate the distribution of confident peptide matches after classification the PEP metric is inferred (see Chapter 1.5.4). Machine learning techniques estimate PEP values particular to the learning models parameters and training data. In this study the PEP values will not be extracted with Scikit-learn and instead a novel technique is explored as visualised in **Figure 2.12**. The classified iPSC PSM datasets are initially ranked by their *prediction probability* $p(-1)$ scores. Then for the $p(-1)$ score of each target spectrum (x_m), the number of target and decoy spectra with a score (x_m) ranging above $(x + 0.01)$ and below $(x - 0.01)$ are tallied. The formula recommended to estimate the PEP is proposed in the works of Percolator which originally measures the frequency of PSMs with the same test statistic (see Chapter 1.5.4.2, **Figure 1.11**) and instead here it is used for this 'container' method.

Identification of mismatched variants is difficult due to their similarity to their canonical counterparts but more so because of their rarity and a stringent threshold of 1% is applied to PEP to deem a PSM as significant. Although generally multiple PSMs would obtain a similar *prediction probability* p(-1) score, this collective method is expected to include spectra that would otherwise be marginally excluded and boost their significance through grouping. The p(-1) scores are considered as a secondary threshold measure which 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 are 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 is annotated for each file. The figures created are to display the PEP against the p(-1) distribution for each search strategy as well as cumulative count comparisons.



$$0.01 < x_m < +0.01$$

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

T_01: PEP = $\frac{1}{3}$ = 0.66 T_02: PEP = $\frac{2}{1}$ = 2

Figure 2.12 PEP calculation method. Classified spectra are ranked in ascending order by the p(-1) test statistic. An example of the method is observed for two target PSMs (T_01 and T_02) at different p(-1) intervals (x_1 and x_2) respectively. The distribution of target and decoy spectra within a range of 0.01 are used to estimate the PEP values for each spectrum (x_m) using the formulae prescribed in the works of Percolator (Käll *et al.*, 2008).

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, however it remains as the final proposed step of the NT pipeline and the Python scripts are available on GitHub at: Task 3: MODY expression analysis for future works. Instead, a MODY protein expression plot was created by observing the PEP score distribution of the 14 MODY genes (see Chapter 1.3.2, **Table 1.1**).

The iPSC PSM datasets were setup to include the inferred protein sequences and gene names of each peptide-match. The iPSC PSM datasets that were classified in each decoy strategy was combined with its original iPSC data file to link estimated metrics to the PSM information. A Python script was written to extract the PEP scores of the 14 MODY genes from the classified iPSC PSM datasets. Peptide sequences generally do identify with multiple genes, and for this gene names were separated into their own PSM with the same metrics. In the case that multiple MODY genes were identified from each file, an average PEP score was estimated. This was conducted for each decoy strategy. The Python script for the *decoy sequence* strategy and *decoy variant* strategy 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. In this folder there is another file containing gene name identifiers as identified from experimental research articles mentioning their possible association. This extensive list was curated by Dr. Ksenia Kustenova and would provide interesting insight for future works of protein-protein interactions concerning MODY and other diabetes conditions.

This setup is informal and not typical for protein expression. The analysis aims to provide a general overview of how the PEP expression of *decoy variant* PSM strategy differs to the *decoy sequence* strategy. Protein identification from peptide-matches is a large and concerning topic in itself, and the extent of it is not considered in 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 peptide-matches. The PEP scores are not an indicator of protein expression level nor abundance. Instead, 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, ExtraTrees and XGBoost, individually reprocessed a PSM dataset composed of proteomic data on tonsil tissues searched against a theoretical peptide sequence database. These target spectra were annotated with their contrasting decoy sequences. Each ensemble model was made to perform binary classification using the target-decoy approach in 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 architectures classification performance is evaluated. The assessments will concern the architectures' ability to discriminate between the target and decoy PSMs using histograms. An overall performance review using AUROC and PR curves using confusion matrix estimations is conducted. The target-decoy approach 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 ROC (Receiver Operating Characteristic) curve is a graphical illustration applied in statistics to display classification accuracy. This 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 models' ability to discriminate dichotomous datasets, which in this case are the spectra with target and decoy class labels. Unlike in disease diagnostics which require 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 target class type is a mixture of non-random true-matches and random incorrect matches (see Chapter 1.5.1, **Figure 1.9)**. The PSM dataset are ranked by the p(-1) test statistic issued by Scikit-learns learning algorithm, to sort spectra in order of prediction confidence to estimate the confusion matrix and infer statistical metrics. The AUROC curve for task 1 is viewed in **Figure 3.1**. *Sensitivity* is a measurement range of 0.0 – 1.0, which reflects the model's ability to predict the positive class type or target spectra. An increasing *sensitivity* displays spectra with a higher chance of being a TP or non-random target. Its companion, *1 – specificity*, is similarly a measurement range of 0.0 – 1.0, reflecting the likelihood of the spectra being a negative class

type or decoy. Spectra with a decreasing *1 – specificity*, are interpreted as having a low chance of it being a FP or incorrect match. A target spectrum with a high *sensitivity* and low *1 – specificity* indicates it is likely a correct non-random true match. An increasing *1 – specificity* with high *sensitivity* indicates the target spectra are predicted as random. Decoy spectra are expected to have a low *sensitivity* and high *1 – specificity*. Therefore, target spectra with similar metrics are likely to be non-random incorrect matches.

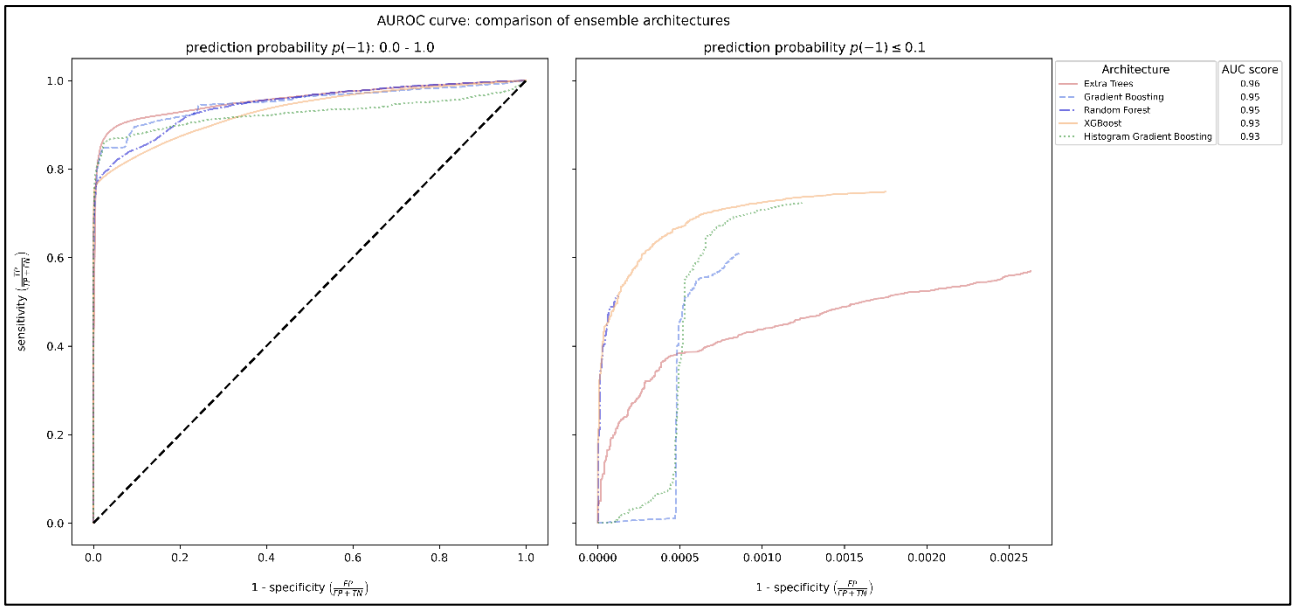


Figure 3.1 AUROC graph. The classification performance curves of five decision tree ensemble architectures are compared. The rate of sensitivity (y-axis) and 1 – specificity (x-axis) is measured for each curve. **Left plot**: The full p(-1) distribution of classified spectra. The black dotted line indicates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

The decision-tree architectures individually reprocessed the PSM dataset and produced spectra with unique p(-1) scores and different ranking positions. The combined AUROC prediction performance for the architectures is viewed in two conditions in **Figure 3.1,** the first (left) is the complete p(-1) range of 0.0 – 1.0 and the second (right) is for spectra with a p(-1) less than 0.1. In this work a threshold of p(-1) $\leq$ 0.1 is applied to spectra as being highly significant with a high chance of being true match as detailed in the methods Chapter 2.3.2. These two plots together provide an overall evaluation of the prediction accuracy 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 a perfect classification. The **Table 3.1**, provides a summary of the interpretation of different AUC intervals. An AUC score of 0.5 are interpreted as ambiguous matches i.e. the learning model estimates there is a 50% chance of a spectrum being either a positive (target) or negative (decoy) class type. This is visualised by the black dotted line in the left plot of **Figure 3.1** which depicts a random model. The conditions of underfitting and overfitting can further be assessed by the curve shape and its distance from this random line, as discussed in Chapter 1.6.1. A curve close to the random line would indicate underfitting, and an uneven curve would 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 the architectures curves are in the furthest upper-left corner of the plot at a distance from the random line and produced an AUC score above 0.9 indicating excellent classification performance. However, on closer inspection to **Figure 3.1(right)** the confidently predicted PSMs do not receive sufficiently high *sensitivity* scores. The confidently predicted spectra therefore may not be amongst the spectra with high *sensitivity* scores of **Figure 3.1(left)**. XGBoost and Histogram-Gradient Boosting architectures had the highest *sensitivity* of approximately 0.7 yet received the lower AUC score of 0.93. This is followed by Gradient Boosting and ExtraTrees, with an AUC score of 0.95 and 0.96 respectively, obtaining a *sensitivity* score of approximately 0.6. Lastly Random Forest performed the weakest with *sensitivity* of 0.5 yet had and AUC of 0.95. However, all the architectures had considerably

low 1 – *specificity* scores well-below 0.1. Therefore, while the instance of these spectra being non-random true matches is poor, they are still relatively reliable with low error-rates.

The AUROC plot is coupled with a precision recall plot to reassure reliability of the ROC curves performance. The PR plot in **Figure 3.2**, are viewed under the same two settings as the ROC curve of **Figure 3.1**. The PR graph showcases the metrics *precision* on the y-axis against the *recall* on the x-axis. The recall metric is the same as the *sensitivity* measurement of the ROC plot. The *precision* is a measurement scaled at 0.0 – 1.0, which an increasing score indicates spectra are highly likely to be true positive match with a low chance of being a false negative. Target spectra with a high *precision* and *recall* score 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.

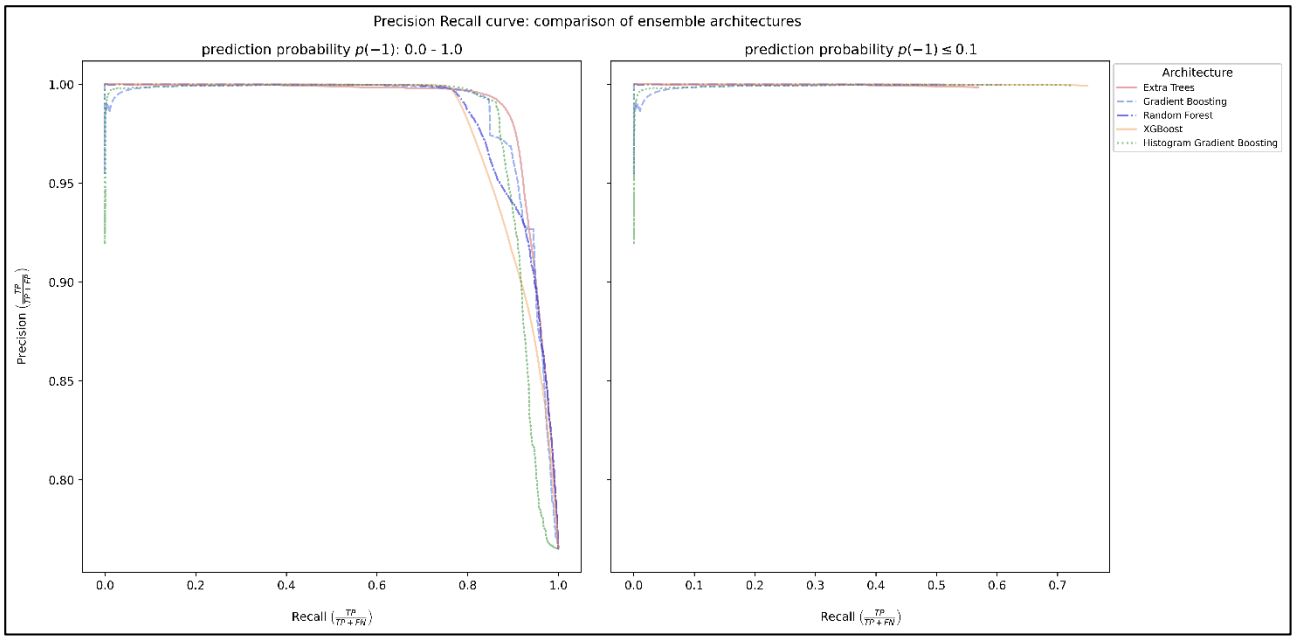


Figure 3.2 Precision-recall graph. The degree of classification accuracy of five decision tree ensemble architectures are compared. The rate of precision (y-axis) and recall (x-axis) is measured for each curve. **Left plot**: The curves are created using the full p(-1) distribution of classified spectra. **Right plot**: The performance of spectra with a p(-1) score less than 0.1 for each architecture are compared**.** The legend (upper right) colour codes the curves for each architecture.

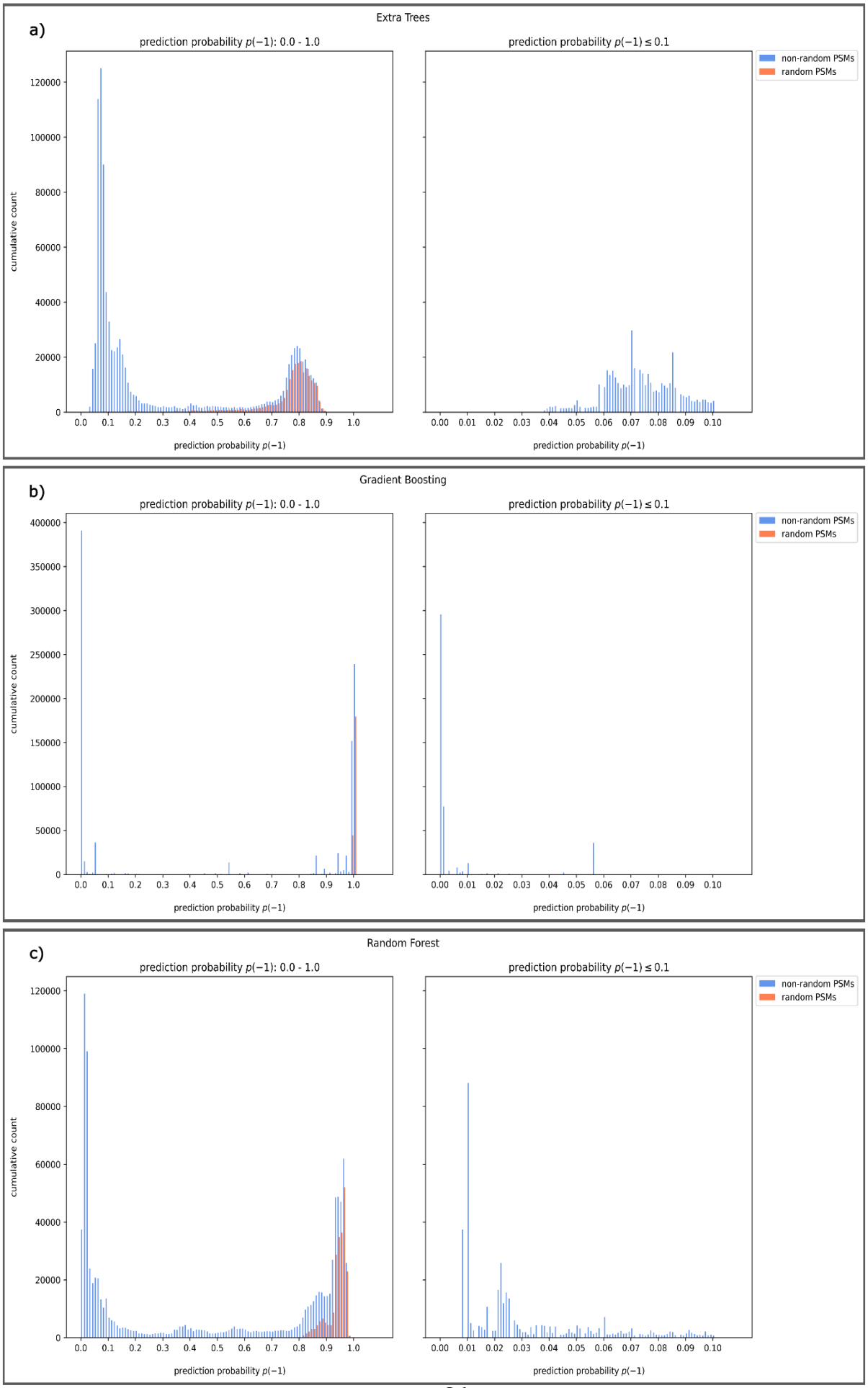
The AUROC and PR curve are directly related as they are estimated using the same classified ranked PSM dataset. In **Figure 3.2(left)** the full p(-1) range spectra are reflecting excellent precision scores considering the y-axis begins at 0.7. Histogram-Gradient Boosting and Gradient Boosting display slight variance, with a dip in *precision* at a low *recall*/*sensitivity* value of 0.0. Interestingly, excellent precision scores are observed for the confidently predicted

spectra in **Figure 3.2(right)** whilst corresponding to a low *recall*/*sensitivity* and 1 – *specificity* rate. Therefore, despite displaying an unconvincing chance of being non-random true matches, the classified spectra still hold some significance which is reinforced by their precision metric. Overall, a preliminary assessment for the absence of underfitting and overfitting at this juncture can be confirmed.

It may be questioned at this point, why the curves in either the AUROC plot or PR plot have a wide distribution across each metric. The PR curve, for example, has spectra yielding a high *precision* yet low *recall*, likely due to an increase in false negatives. A FN in this instance would be a poorly classified decoy or a random target spectrum. The PSM dataset used in this task contains 1,187,495 peptide matches and it is difficult to isolate the trends of individual spectrums using these plots at this scale. Both the target and decoy spectra were included in the AUROC and PR graphs, therefore it is further difficult to identify which class type is causing certain trends, such as an increase in a FN rate. It is usual to include further metrics such as confidence intervals to ensure validity of ROC graphs. This will be elaborated on in the discussion chapter of this work. To visualise the relationship of the classified target and decoy spectra a secondary evaluation method using histograms is next explored.

3.3 Histograms

Generally learning models built to conduct binary classification would be 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 and is then unable to classify unseen data. Underfitting occurs due to a model over simplifying patterns and failing to critically classify resulting in bias 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 for the classified ranked PSM dataset of each decision-tree ensemble model was created and can be viewed as a collection in **Figure 3.3**. The non-random and random PSMs correspond to the target and decoy class type respectively. Just as with the AUROC and PR graphs in the previous chapter, the classified PSM dataset for each architecture is viewed in two conditions, **Figure 3.3(left)** views the complete p(-1) distribution range, and **Figure 3.3(right**) is of spectra with a p(-1) range less than 0.1.


Extra Trees
a)
prediction probability $p(-1)$: 0.0 - 1.0
prediction probability $p(-1) \leq 0.1$
non-random PSMs
random PSMs
cumulative count
prediction probability $p(-1)$
Gradient Boosting
b)
Random Forest
c)

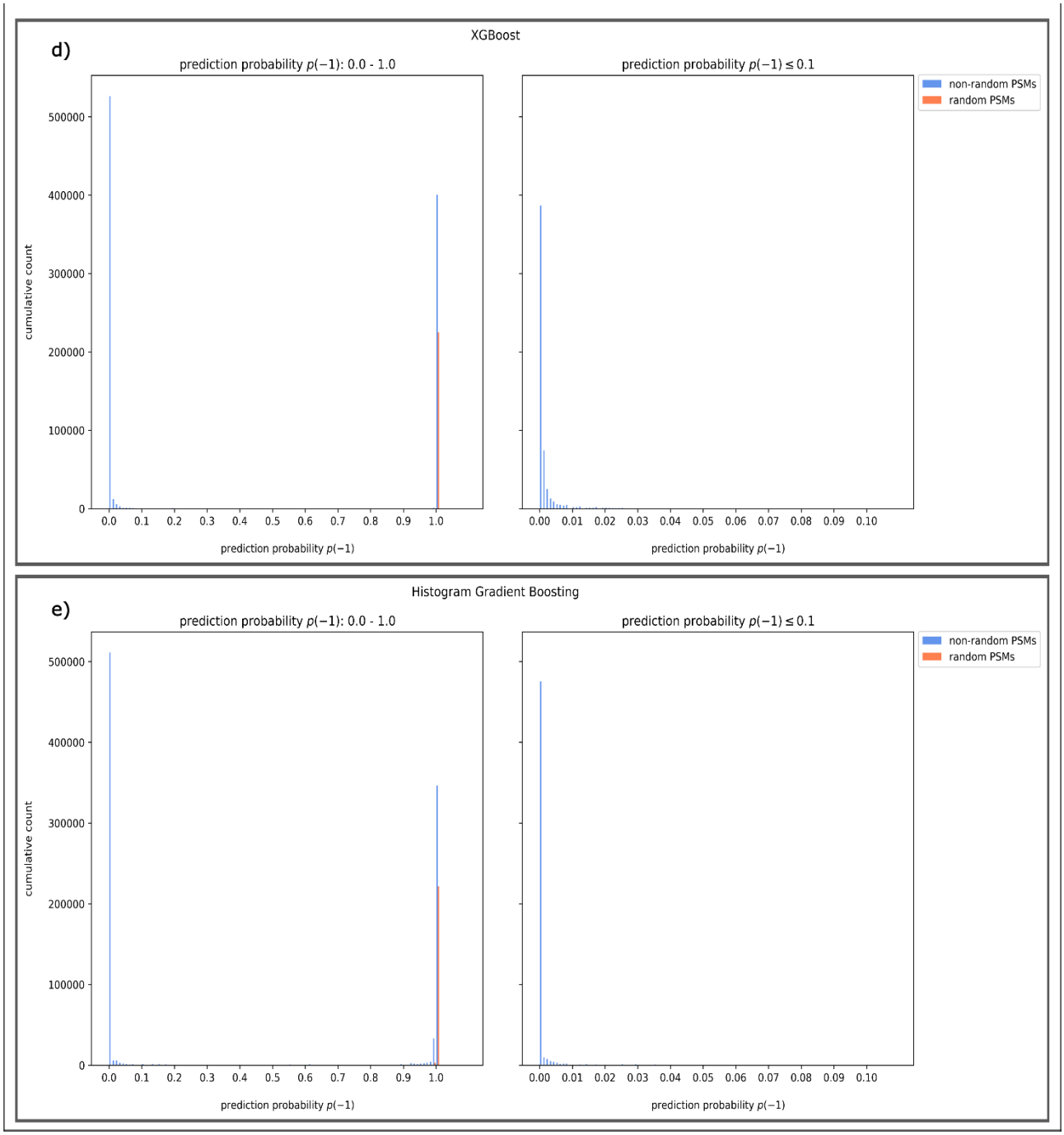


Figure 3.3 Histograms of the PSM dataset classified by decision-tree architectures. Classified non-random (blue peaks) and random (red peaks) spectra are ranked in ascending order by the *prediction probability* p(-1) test statistic. There are two histograms for each architecture. The histograms(**left**) 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 and **(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 trends of the non-random and random PSMs and evaluate model performance. A bias model would be observed under two conditions, the first being all non-random spectra are located far left at interval 0.0 – 0.1 and random spectra are far right at interval 0.9 – 1.0. This is interpreted as the model being overly confident and has failed to objectively discriminate between the class types. An initial observation is that neither of the architectures display 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 threshold at a 1% FDR and decoy PSMs, and consequently a clear split in classification is an expected occurrence although at varying degrees. *Boosting* models do increase the significance of *weak-learners* (see Chapter 1.8.2.2), which further corresponds to the decrease in spectra between the interval 0.1 - 0.9 in their histograms. The *Bagging* models (see Chapter 1.8.2.1), ExtraTrees and Random Forest (**Figure 3.3[a, c]**), display a well-balanced distribution of target and decoy spectra. These methods operate by maximising their learning strategies with the aim of learning the intricacies of the dataset. This robust approach is reflected in the wide distribution of the spectra class types in their histograms.

With regards to the p(-1) distribution range, spectra above a score of 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 of 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 for the p(-1) score range as depicted on the x-axis of the histograms. The second condition to evaluate performance bias, is if there are more random than non-random PSMs occurring at each p(-1) probability interval. This would be observed if for each interval there is a longer red peak than blue peak in either of the histograms. In each of the architecture's histograms, this is not observed beyond the p(-1) 0.5 probability distribution, however it is difficult to conclude the frequency of both class types below this range. A solution to this is described next.

3.4 Cumulative count analysis

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 of architectures histograms (**Figure 3.3(right)**). The ExtraTrees and Random Forest architectures continually exhibit a wider distribution of spectra than 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 significantly lower 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 (meaning to reduce the scale of the y-axis for peak visualisation). To address this issue, a comprehensive overview of the cumulative count of non-random and random PSMs with p(-1) below 0.1 is observed in **Figure 3.4**.

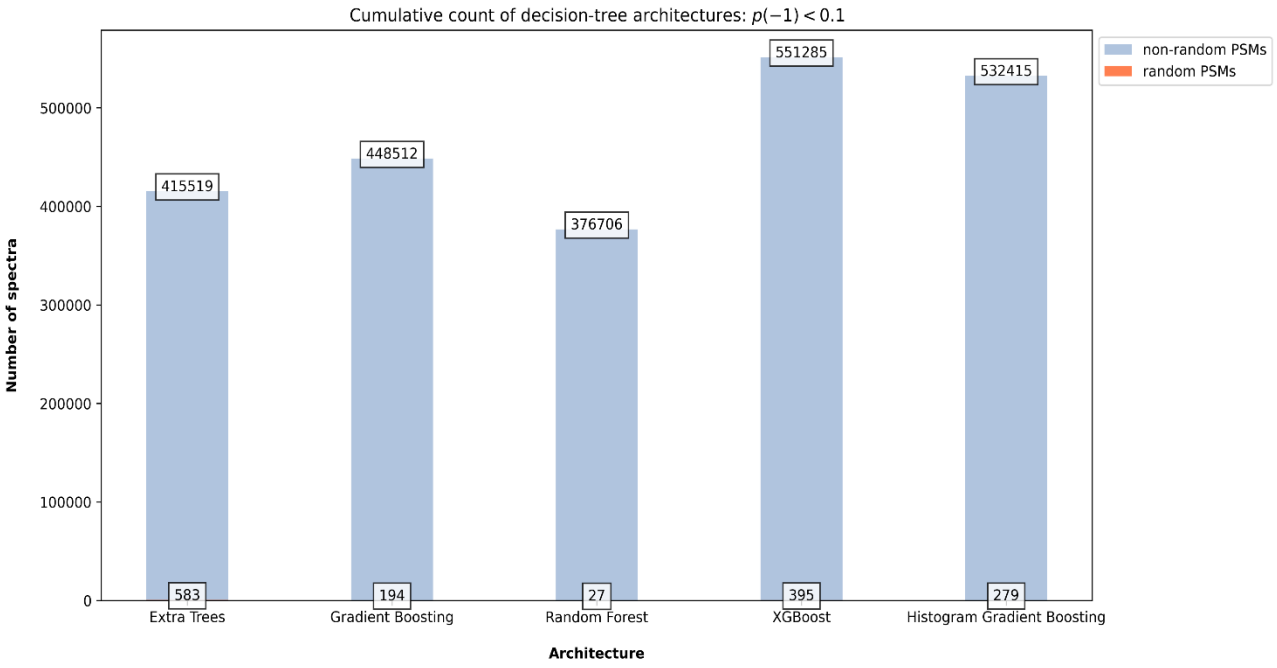


Figure 3.4 Cumulative count of decision-tree architectures: p(-1) < 0.1. The PSM dataset consisting of two PSM class types: non-random (target) and random (decoy), was reprocessed by five decision-tree ensemble architectures. The total number of classified spectra with a p(-1) score less than 0.l are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

In this display, the low frequency of false negative random (decoy) PSMs are 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 with 551 285, followed by Histogram Gradient Boosting at 532 415 and GB at 448 512. ExtraTrees classified 415 519 and Random Forest had the least at 376 706. Considering spectra above a p(-1) score of 0.5 are insignificant, PSMs below this are acceptable as moderately significant. Non-random target spectra within a p(-1) range of 0.1 – 0.5 are still representative of possible true matches. In **Figure 3.5**, the total number of spectra below a p(-1) 0.5 score, which includes those below 0.1, is observed. 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 that were classified in both intervals. Here, ExtraTrees surpasses the other architectures and has classified 70% of the target spectra of interest, which aligns with its histogram **Figure 3.3[a]**, exhibiting spectra with the widest distribution of p(-1) scores. Random Forest collectively classified 480 742 non-random spectra which is half of the target spectra from the PSM dataset. The majority of the non-random spectra from the *Boosting* models, Gradient Boosting XGBoost and Histogram Gradient Boosting, was classified below a p(-1) score of 0.1. The difference in distribution is subjective, therefore, overall correctness of spectra within this p(-1) range is evaluated.

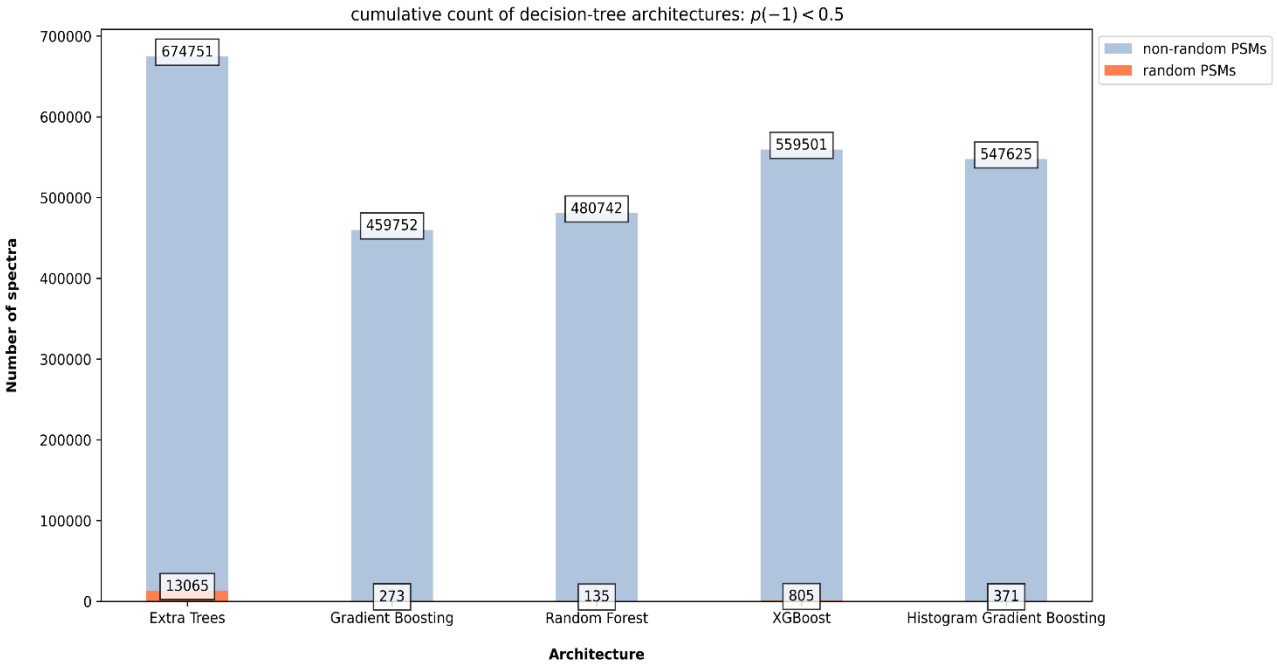


Figure 3.5 Cumulative count of decision-tree architectures: p(-1) < 0.5. The PSM dataset consisting of two PSM class types: non-random (target) and random (decoy), was reprocessed by five decision-tree ensemble architectures. The total number of classified spectra with a p(-1) score less than 0.5 are compared. The legend on the upper-right corner provides a colour key of each PSM class type.

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

Number of non-random 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 threshold at 1% FDR initially selected from the PSM dataset ranked by the *pep* scoring function (**Figure 2.1**, step 1). Thereafter the counting function estimated a new FDR in effort 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 less than 0.1 are less likely to be an incorrect match and those less than 0.5 are threshold as acceptable. These number of spectra within these two intervals are roughly similar to their cumulative counts in the previous chapter (**Table 3.2**). Classified spectra above the p(-1) 0.5 score are insignificant and here, their frequency within the 1% FDR threshold is observed.

Interestingly the *Boosting* and *Bagging* models classified false positives uniquely in contrast to their classification trends of significant spectra. Gradient Boosting contained 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. Interestingly, ExtraTrees 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 insignificant spectra, ExtraTrees is observed to be the more reliable model out of all the architectures and is thus used for implementing task 2.

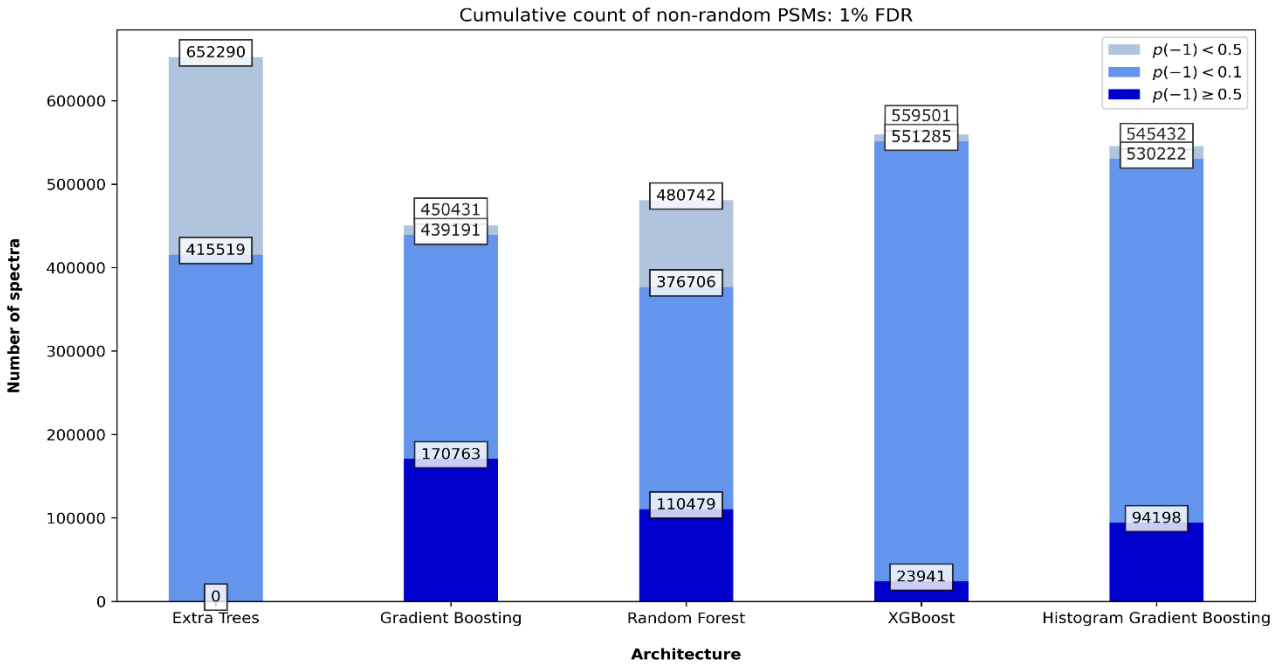


Figure 3.6 Cumulative count of non-random PSMs: 1% FDR. The number of non-random spectra portioned into three different p(-1) intervals is visualised for each decision-tree architecture. Spectra classified and threshold with an 1% FDR error rate are compared. The legend on the upper-right corner provides a colour key of 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 exist at lower frequencies despite have high penetrance in terms of impact (Bainbridge, 2020). Due to this, they are often unobserved in protein identification tasks when combined with a disproportionately larger canonical proteome during search annotation. The high degree of similarity between the variant and canonical peptide sequences makes identification even further challenging when reprocessing PSM datasets. To this end, the method of task two 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 of the canonical proteome. Here in this task, its approach is adapted to investigate the concept of *decoy variant* PSMs. Originally the canonical and haplotype variant peptide sequences are combined as the target class type. Here they are separated into two individual target class types, and each generated a set of decoy sequences. These theoretical spectra were then used to annotate iPSCs genetically modified to suppress insulin resistance by

altering the HNF1A gene. This is an effort to visualize the performance of *decoy variant* PSMs and canonical sequences in a practical example (and later infer MODY identification).

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

Pertaining to the performance evaluation of the five decision-tree ensemble architectures in task 1, ExtraTrees was selected to perform the classification for task 2. This investigation is a novel pursuit to evaluate the potential of using *decoy variant* PSMs as a null model as compared to searching canonical *decoy sequence* PSMs. To the current knowledge of study, this has not been previously reported. Here, the performance is observed using histograms to monitor PSM distribution. Then to assess statistical significance of peptide identifications in both search strategies the distribution of PEP scores in relation to *prediction probability* $p(-1)$ estimates are evaluated. The figures in this chapter along with supplementary reports are available on GitHub at: Thesis and supplementary figures.

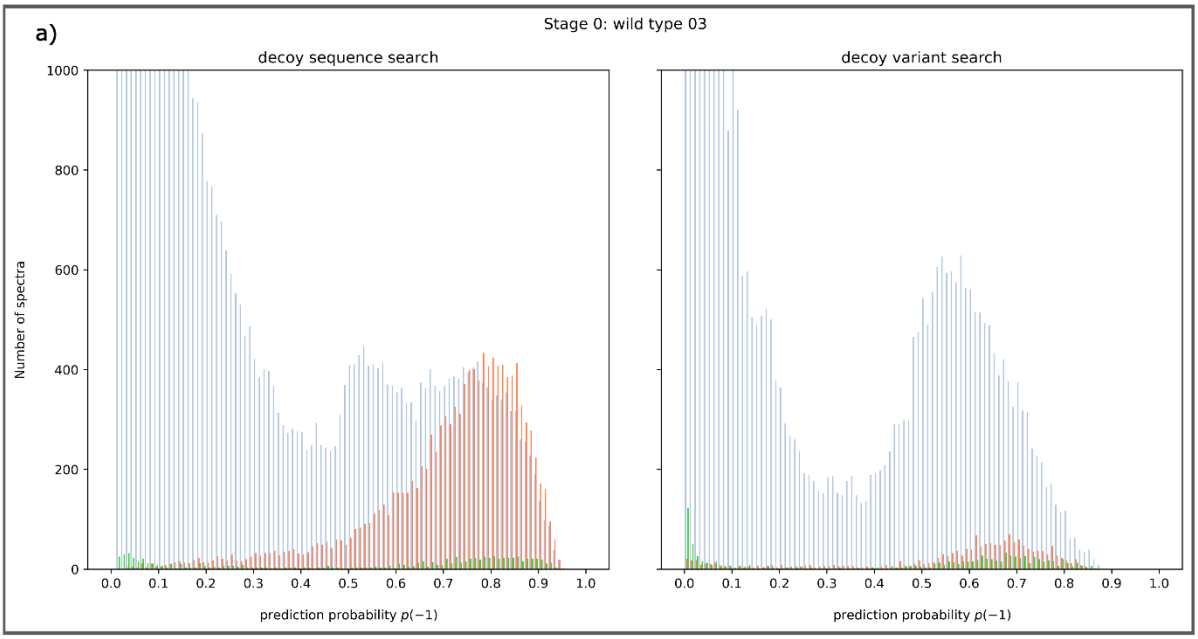
3.5.1 Histograms iPSC files

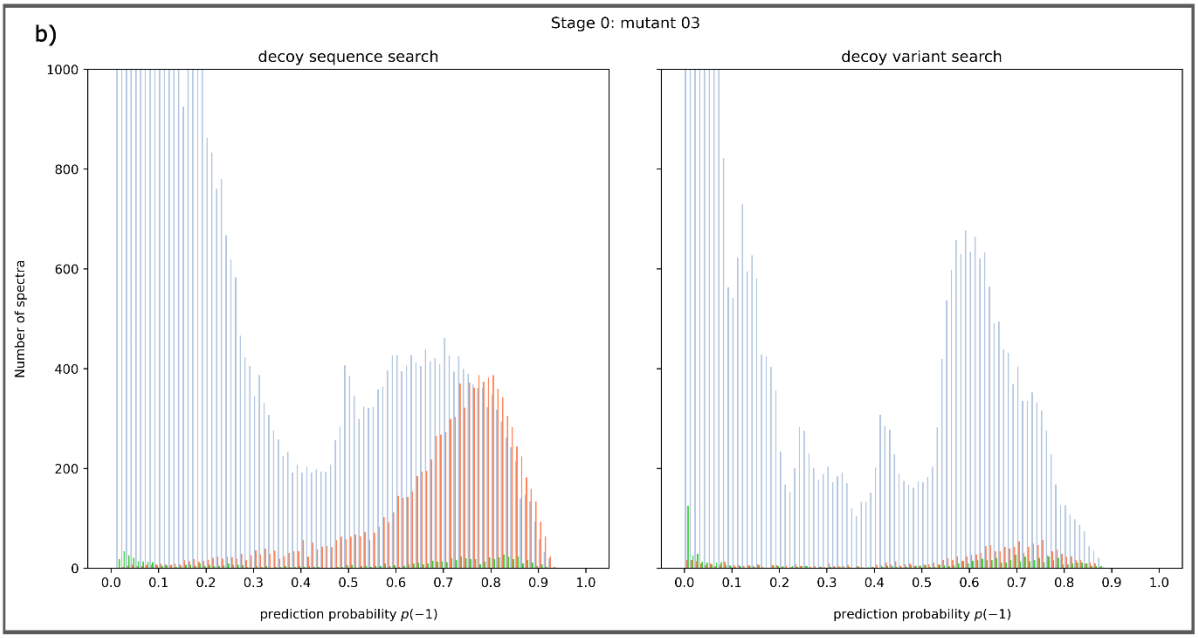
Similarly, as with task 1, here histograms are employed to investigate the distribution of classified PSMs. However, in this report, events of overfitting and underfitting are not investigated. Instead, the histograms provide the means to observe the behaviour of decoy PSMs. The twelve iPSCs files were reprocessed in two search conditions using the ExtraTrees architecture with the Nagilums Tree pipeline. In total 24 histograms were produced and here for brevity eight are provided and the remainder are provided as Supplementary. The histograms produced are labelled according to their development stage, cell type and cell line e.g. *Stage 0: wild type 01, Stage 4:mutant 02* etc. The files 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*. Note, the terminology of 'mutant' used hereon is in context of the altered iPSC and not associated with any persons. This is to not confuse with the context of 'variant' peptides concerning human health conditions mentioned in the discussion later.

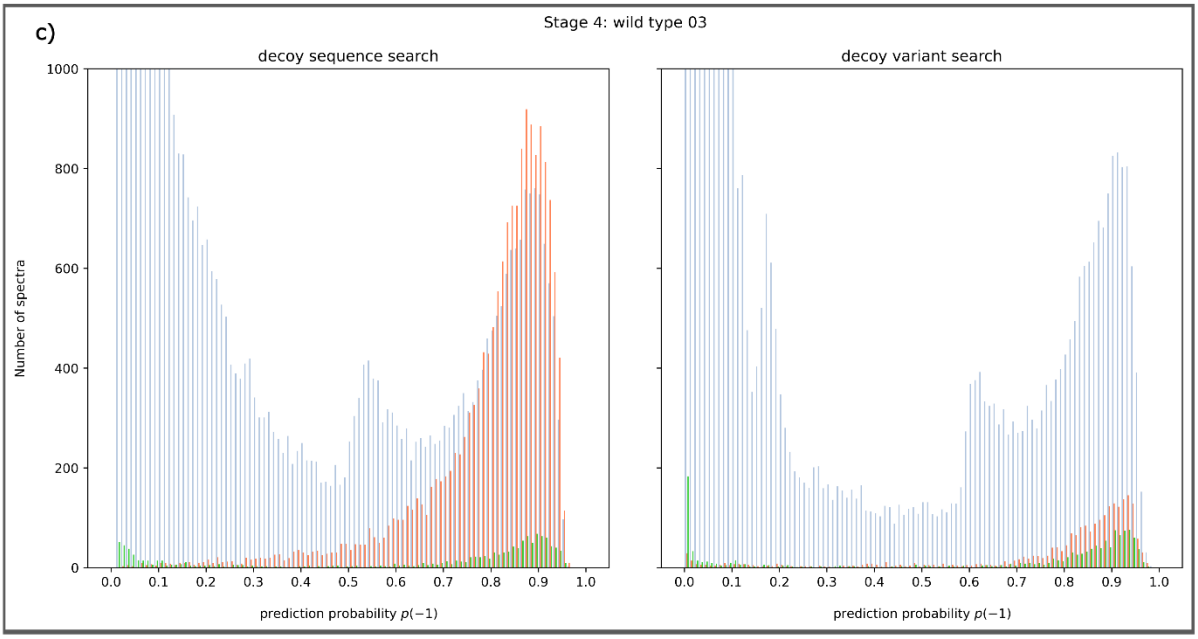
Two histograms are provided with each iPSC type, each representing the performance of a search strategy. In both strategies, the *canonical* PSMs (blue peaks) and *variant* PSMs (green peaks) were set as positive class labels to represent target PSMs. The first search condition

(**Figure 3.7(left)**) represents the *decoy sequence* strategy. In this approach the *decoy variant* PSMs were removed from the iPSC PSM datasets, and the learning algorithm was trained to discriminate between negative class labelled *decoy sequence* PSMs (red peaks) and the target PSMs. The second search condition (**Figure 3.7(right)**) represent the *decoy variant* strategy. The *decoy sequences* were instead removed in this approach, and the *decoy variants* (red peaks) were set as the negative class labelled decoy PSMs.

Observed peptides are different with each cell line and various levels of peaks are to be expected. Each PSM class type existed in varying frequencies dependent on their availability in the iPSC peptide datasets. The PSM class type frequency for the iPSC dataset *Stage 0: wild type 03*, for example, contained: 88 331 *canonical* PSMs, 9 854 *decoy sequence* PSMs, 864 *variant* PSMs and 1 344 *decoy variants* PSMs. The other iPSC PSM datasets contained class type ratios at similar portions. The canonical PSMs are largely more frequent and evidently yield higher volumes of classified spectra. Considering this task concerns the distribution of decoy PSMs, the y-axis is scaled for their lesser frequencies to improved analysis. The fewer instances of *variant* PSMs are then able to be evaluated as well. If it is questioned at this point as to why there are high portions of peptide matches 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 undifferentiated pancreatic β-cells and are expected to produce protein products. Note these PSM values and distribution in the histograms are not an indicator of protein expression levels, and simply suggest the peptides observed after reprocessing.


a)
Stage 0: wild type 03
decoy sequence search
decoy variant search
Number of spectra
prediction probability p(−1)
prediction probability p(−1)


b)
Stage 0: mutant 03
decoy sequence search
decoy variant search
Number of spectra
prediction probability p(−1)
prediction probability p(−1)


c)
Stage 4: wild type 03
decoy sequence search
decoy variant search
Number of spectra
prediction probability p(−1)
prediction probability p(−1)

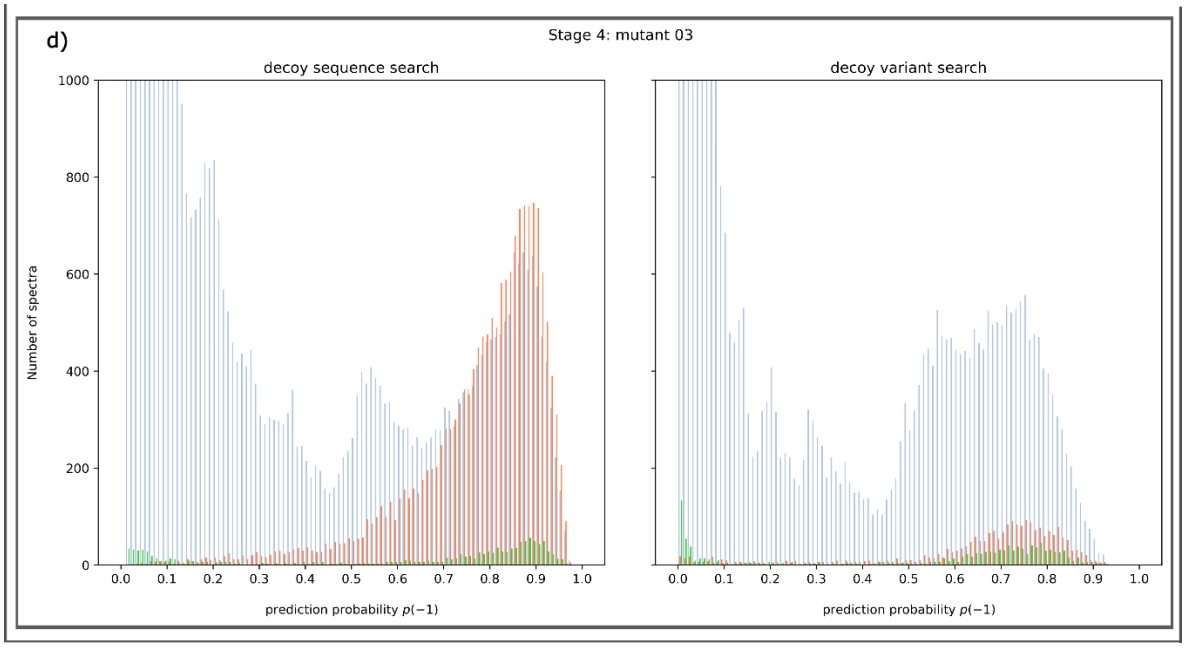


Figure 3.7. Histograms of four reprocessed iPSC PSM datasets. The PSMs for two development stages (stage 0 and stage 4) of wild-type and mutant iPSCs are ranked in ascending order by the prediction probability p(-1) test statistic. Cell line 03 is observed in these histograms: **(a)** Stage 0: wild type 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 reprocessing; **Left plot**: decoy sequence, and **right plot**: decoy variant The spectra are dispersed into 100 bins for each histogram.

The distribution of classified iPSC PSMs are observed using the *prediction probability* p(-1) scores applied by the learning algorithm under Scikit-learns framework (Pedregosa *et al*., 2011). The scores are a range of 0.0 – 1.0 where spectra with a p(-1) value closer to 0.0 indicate there is a minimal chance the peptide matches are an incorrect match or false positive. The decoy PSMs, in both strategies, represent the null hypothesis. The expected observation is that the positive class type (blue peaks) would skew more to the left (closer to 0.0) and the negative class type (red peaks) skew more to the right (closer to 1.0). Spectra are threshold at $p(-1) < 0.5$ for statistical significance, and any classified above are identified as false positives.

Three main observations, concerning the trends in the iPSC histograms collectively (**Figure 3.7**), are made for the *canonical* PSMs, *variant* PSMS and both decoy strategy PSMs with the latter two being of more importance. Firstly, the *canonical* PSMs are clearly skewing left despite the y-axis limited scale. Secondly, 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 discussed in discussions chapter of this work (see Chapter 4.3).

Lastly, the *variant* PSMs in either search strategy appears 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) 0.5 score in either of the histograms. More importantly, clear spikes in *variant* PSMs are observed at a p(-1) 0.0 value within **Figure 3.7(right)** for the *decoy variant* search *strategy*. This is particularly interesting in the *wild type* iPSCs **Figure 3.7[a, c]** in which no gene alteration occurred. This is another promotional point concerning the use of *decoy variants* as a null model.

The false positive *variant* PSMs also appear to follow the trend of the *decoy variant* PSMs. The *mutant* iPSCs experienced a single gene alteration in experimentation and the feedback of protein expression is undetermined at this point. The histograms of **Figure 3.7 [b, d]** however suggest more *variant* PSMs were classified as significant than the *wild type* iPSCs (**Figure 3.7[a, c**). To establish the frequency of statistically significant peptide matches the posterior error probability was employed.

3.5.2 Posterior error probability analysis

In the histograms above (**Figure 3.7**), it was observed that the *canonical* PSMs and *variant* PSM class types were exceptionally well discriminated against the *decoy sequence* PSM and *decoy variant* PSM search strategies. Spectra achieving a *prediction probability* $p(-1)$ score < 0.5 are threshold as acceptably significant and it appeared there is an influx in *canonical* PSMs and *variant* PSMs within this range. To determine viability of these spectra, the PEP was estimated on the iPSC PSMs during reprocessing using the novel 'container' method. The PEP is a probability statistic with a range of 0.0 – 1.0. Target spectra (*canonical* PSMs and *variant* PSMs) with a PEP value closer to 0.0 are estimated as having a low chance of being a false positive amongst negative class type spectra (decoy spectra) with similar $p(-1)$ estimates. 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 represents the distribution of the PEP against the *prediction probability* $p(-1)$ values of *canonical* PSMs and *variant* PSMs, in combination, for the twelve 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 tests' statistics. The PEP distribution in **Figure 3.8[a]** is observed to extend beyond 1.0 for spectra with an increasing $p(-1)$ value. These spikes are indicative of under sampling, as in there were more decoy than target spectra beyond this interval. Interestingly, in **Figure 3.8[b]**, spectra 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 as there were generally a larger portion of target spectra than either of the decoy class type PSMs.

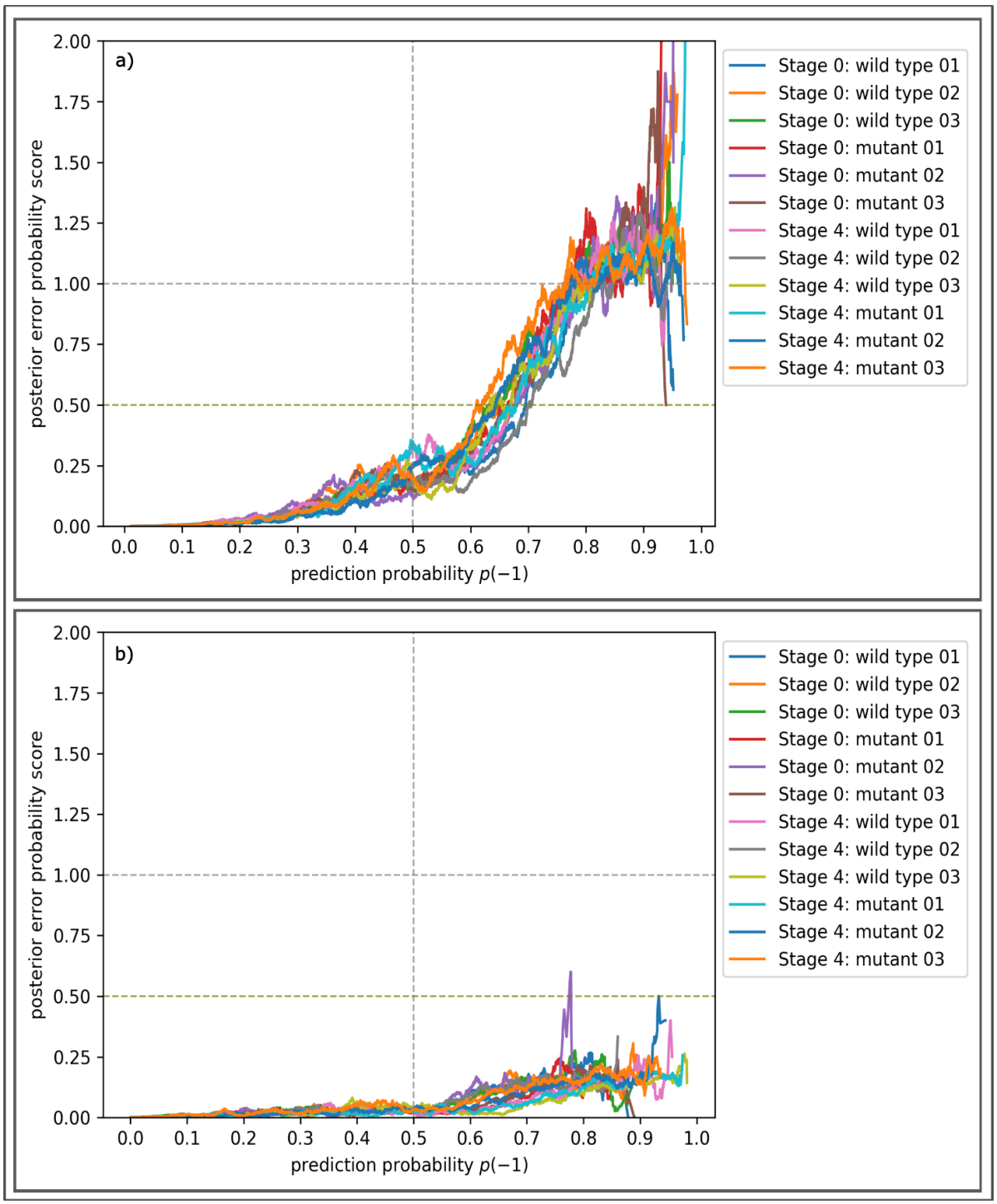


Figure 3.8 Distribution of significant spectra for two decoy search strategies. The posterior error probability of twelve reprocessed iPSC PSM datasets are plotted against the *prediction probability* $p(-1)$ estimates. A threshold of $p(-1) < 0.5$ (grey vertical dotted line) and PEP < 0.5 (olive horizontal 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, PEP is threshold at $\{< 0.1\}$, especially regarding 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, spectra threshold of $p(-1) < 0.5$ are accepted as significant as they are classified below the possibility of being a false positive. Any spectra beyond this are deemed insignificant regardless of if 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 that of the *decoy sequence* strategy. To evaluate this performance, a cumulative count of the *canonical* and *variant* PSMs in both search strategies is provided.

In **Figure 3.9**, the general scale of the total number of spectra threshold at $p(-1) < 0.5$ and *PEP* < 0.5 are evaluated. Interestingly, the *decoy variant* search strategy **Figure 3.9[b]** produced more *canonical* and *variant* PSMs than that of the *decoy sequence search* **Figure 3.9[a]**. In certain cases, the *decoy sequence* strategy produced more spectra in the *Stage 0: mutant 01* and *Stage 4: wild type 01* datasets. The *decoy sequence* and *decoy variant* strategy produced more spectra for the *Stage 4: mutant 01* dataset. The pancreatic development Stage 0 represents early developed cells and express a wide diversity of proteins. As iPSCs mature they produce proteins associated to the intended specialized human cell. This is demonstrated in the fewer spectra returned by the S*tage 0 mutant* iPSC datasets compared the *wild type* iPSCs for both the *canonical* PSMs (**Figure 3.9[a]**) and *variant* PSMs (**Figure 3.9[b]**).

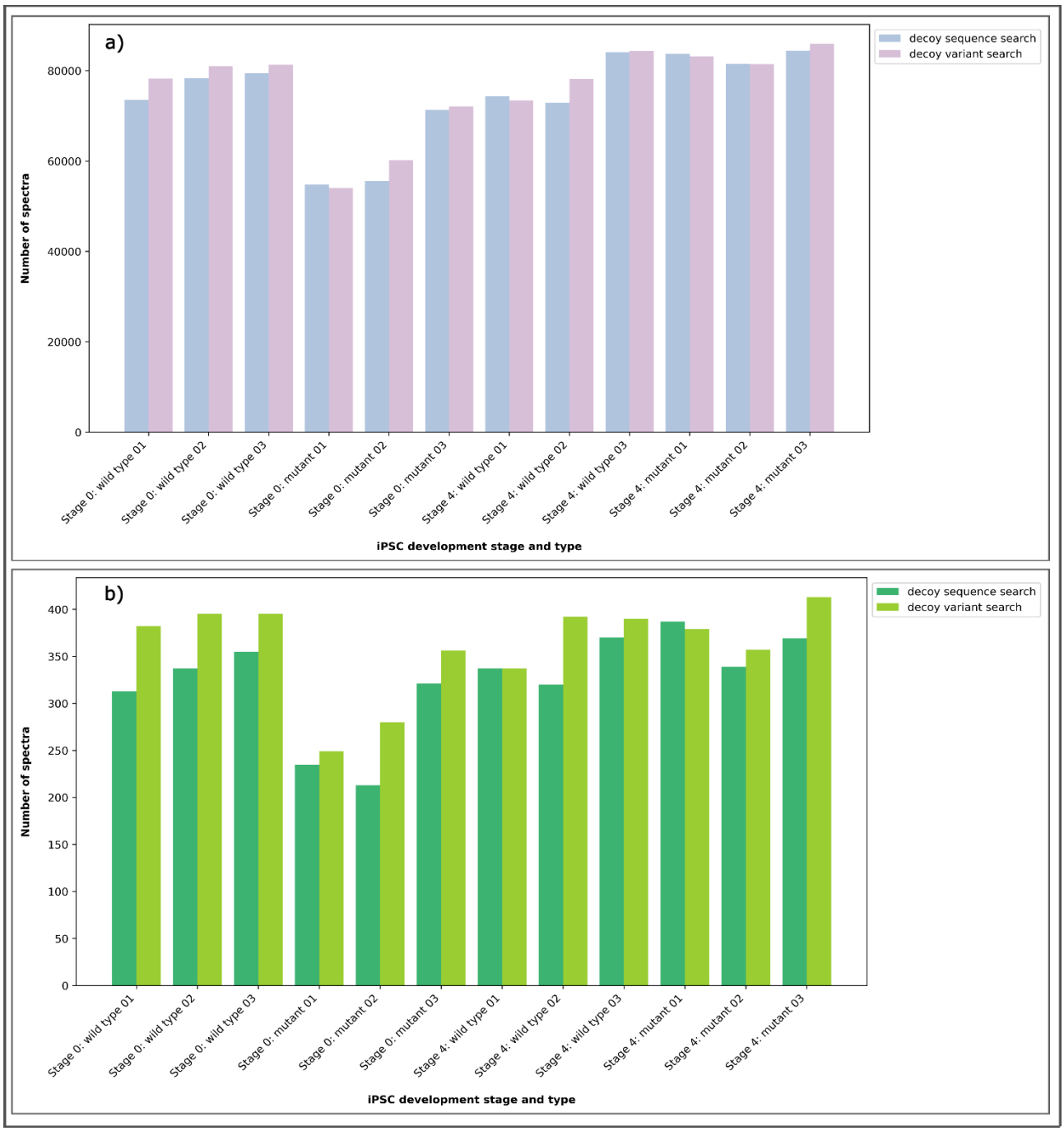


Figure 3.9 Performance evaluation significant spectra produced by the decoy sequence and decoy variant search strategies. Twelve iPSC PSM datasets were reprocessed using two search strategies. The number of canonical and variant spectra threshold at a PEP < 0.5 and $p(-1) < 0.5$ are compared. **(a)** canonical PSMs **(b)** variant PSMs. The legends on the upper-right corner of each plot provides a colour key of each search strategy.

Stage 4 is the pancreatic progenitor stage in which cells are not fully differentiated into pancreatic cells but do express pancreatic protein markers. Here iPSCs were grown into the insulin producing β-cell of the pancreas. A variant was engineered on the HNF1A gene to created mutated iPSCs. The effect of the variant is not an inhibitor of insulin production but rather a suppressor. Expression of associated proteins is expected to be indirectly impacted. It is assumed that the *Stage 4 mutant* iPSCs produced more proteins than the *wild type* iPSCs which is supported by the larger amount of peptide matches. The increase is a possible indicator of over expression to compensate for the defective gene, however that is undermined 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 view the performance of the decoy search strategies in regards to identification of MODY genes, **Figure 3.10** is provided. In these images, the PEP distribution was used to investigate which MODY genes are present in each iPSC cell type and development stage. 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 nor peptide abundance. Peptide identification tasks are subjective the discriminative performance of the learning model and composition of the theoretical. The MODY genes displayed here are from the *canonical* labeled PSM class types. There is a single MODY gene observed of the *variant* PSM class type from a *wild type* iPSC of the *decoy variant* strategy for ABCC8(MODY12) visualised in the supplemetary figure on GitHub: Supplementary figures. Included in this folder, are 4 plots for the full list of possible MODY genes. There are two plots displaying both the PEP distribution of *canonical* PSM and *variant* PSM class types and two others observing only the *variant* PSM class. These plots are included for future works for protein-protein interaction investigations.

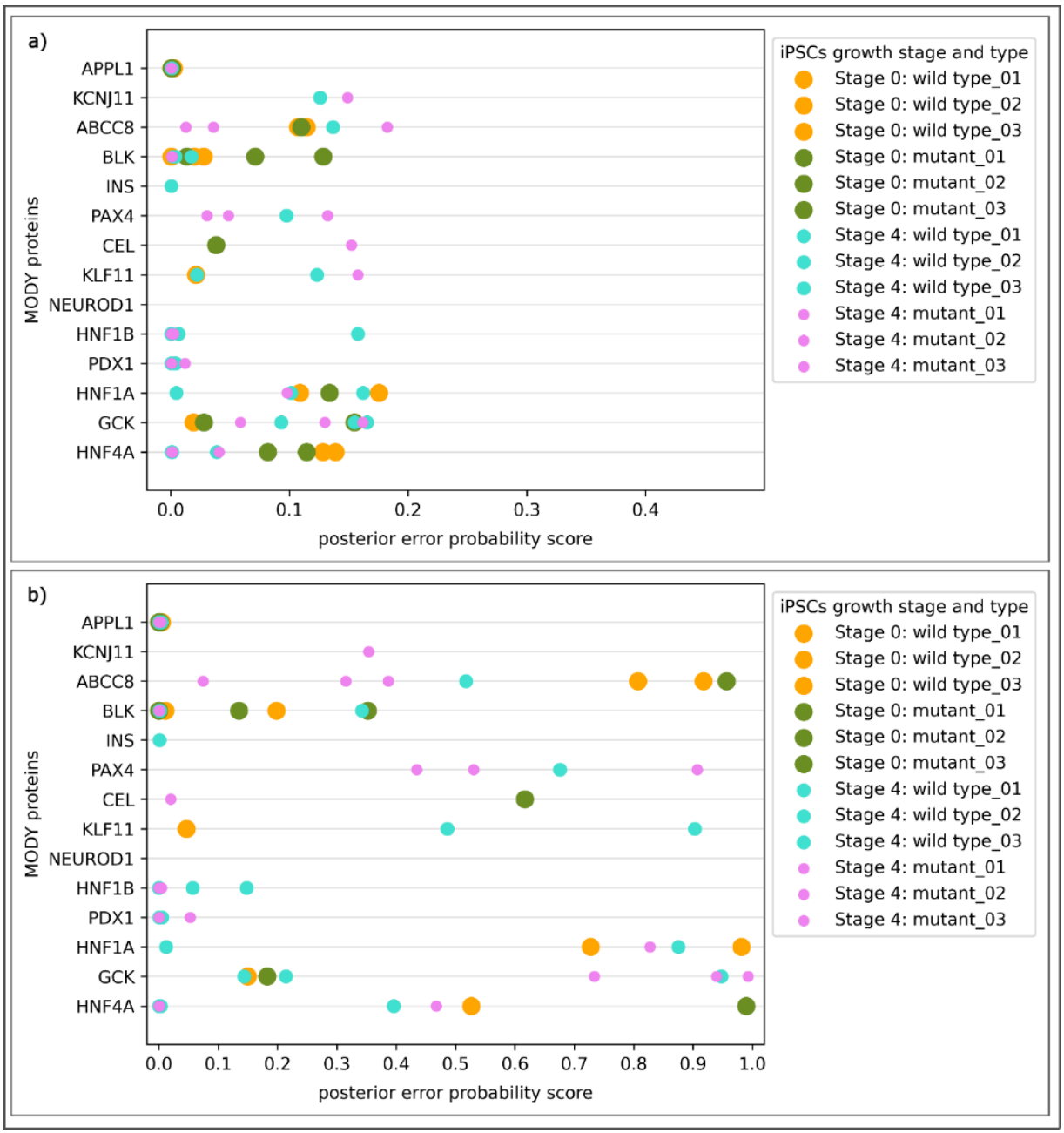


Figure 3.10 PEP distribution of 14 MODY genes. Gene identifiers of the 14 MODY genes are collected from the 12 classified iPSC PSM datasets for decoy search strategies. **(a)** Decoy variant search strategy. **(b)** Decoy sequence search strategy. Legends of the right for both plots indicate the development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

The *mutant* iPSCs were an engineered pancreatic β-cell with a variant of the HNF1A-MODY3 gene. The general expectation is that there would be an indirect effect on associated MODY genes and interacting proteins. A single *mutant* cell at *Stage 0* and at *Stage 4* is observed for the *decoy variant* strategy (**Figure 3.10[a]**). A single PSM for a *mutant* iPSC at *Stage 4* is observed for the *decoy sequence* search strategy (**Figure 3.10**).

The gene identifiers for the PSMs of each iPSC dataset were included in the theoretical dataset prior to peptide annotation. The MODY genes here therefore indicate a single peptide possibly associated with their protein structure was present in the iPSC files. It is unconfirmed if the peptide-matches are also associated with other unrelated protein sequences. The number of peptide sequences composite of each MODY protein is not confirmed as no PSM to protein-inference study was conducted. What can be learned from these results however, is how the different decoy search strategies impacts confidence of peptide identification.

The PEP is a probability distribution from 0.0 – 1.0 representing most confident PSM to least confident, 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* PSM to the combined *canonical* PSM and *variant* PSM target class type was disproportionate (see Chapter 3.5.1). The topic of imbalanced dataset impacting statistical inferece 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 downstream reprocessing 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 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 if the construct of the NT pipeline is successful in itself, therefore the implementation results are explored 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 and 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 *de facto* 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). There is therefore an increasing need to interpret this data and determine its quality in general for proteomic research. Evolvement of next-generation sequencing technologies are similarly improving genomic definition. More specifically, protein identification for gene expression analysis is paramount in investigations concerning disease pathology and therefore there are particular needs 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 structured to identify mass spectrometry acquired peptide sequences 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), which offer unique algorithms to identify the most similar sequences. This is achieved by 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 each search engine (America *et al*., 2006; Yu *et al*., 2006; Kwon *et al*., 2011). It is further assumed that genetic databanks e.g. Ensembl (Harrison *et al.*, 2024), UniProt (Bateman *et al*., 2024), RefSeq (Pruitt *et al*., 2007), contain all known and correctly identified protein-coding genome sequences and their protein products. This is unlikely to be true, therefore, any inferred peptide-matches to a theoretical dataset are understood as presumptuous during the explanation of proteogenomic data analytics. Database search engines operate on the assumption each observed spectrum is an accurate match (Nesvizhskii, 2014). 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 the protein databanks being more abundant with canonical proteome sequences over variant isotopes. In turn, this creates biased scoring functions which are reliant 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 identification to a reference peptide within the search space. A strict threshold can reduce the chance of incorrect matches although there is a risk. a

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 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. Considering this work, aimed to identify rare, low in frequency, MODY variants, these conundrums are considered.

4.2.1 MODY gene expression: decoy variant concept

Although the iPSCs were not fully developed into pancreatic β-cells, the pancreatic progenitor phase (*Stage 4*) expresses proteins relevant to the organ (van de Bunt *et al*., 2016). The early phase, *Stage 0*, are observed as premature cells (van de Bunt *et al*., 2016). While protein expression levels cannot be determined at this point, a preliminary expectation is that there would be relatively more proteins at *Stage 4* than *Stage 0*. In the MODY gene expression plot, **Figure 3.10**, it can be observed which MODY genes were identified from the classified iPSCs collectively. Disregarding the decoy search strategies and PEP expression for now, an initial observation is that the MODY genes of *Stage 0*, in both **Figure 3.10[a & b]**, are much less frequent (around half the amount) than those of *Stage 4*. This is in accordance with the expectations of iPSC cell development, and it does appear that the classification system of NT was able to capture this successfully.

Continuing with **Figure 3.10**, despite a few observations of *mutant* cell type PSMs associated with MODY, majority were of the *canonical* PSM class type. A single protein was observed (Supplementary figure GitHub link) as a *variant* PSM of the *wild type* cell line that was not associated to the HNF1A gene. Proteins are comprised of a family of fragmented peptides and PSM to protein-inference (Shi and Wu, 2009) was not conducted in this work. Generally, the confidence of each peptide of a protein would be assessed and here inference was assumed from a single peptide-match and duplicates were combined into an average (see Chapter 2.8.2.6). This single variant occurrence on an unaltered gene may be associated with other proteins or a random occurrence. Additionally, no *variant* PSM was observed for the HNF1A gene as the PSM observed for a *Stage 4 mutant* iPSC in either decoy strategy of **Figure 3.10** is from the canonical proteome. These two observations can be explained by referring to the caveats of proteogenomics stated earlier, reiterating the difficulty of identifying rare variant peptide sequences. The scope of this project aimed to improve resolution of rare MODY variants, although it is very possible that they were never included or mismatched in the PSM datasets to begin with prior to classification.

The histograms (**Figure 3.7**) of the *decoy variant* concept (see Chapter 3.5.1) displayed a high frequency of confidently predicted *variant* PSMs however it is unknown which are high and low frequent occurrences. The confidently predicted *variant* PSMs from these histogram plots are the likely biproduct from unknown MODY genes or other diabetic conditions. The diagram of **Figure 4.1** illustrates the many genes revealed to be associated with major diabetes types including monogenic diabetes (MODY and NDM). Considering T1D and T2D are afflicted by

insulin dysfunction or resistance; it is expected to observe shared regulatory genes amongst 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 are well-considered. To this end, the extended list of MODY genes that was conducted as a supplementary thought 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 of this extensive list may hold novel connections and would be an exciting topic for future works. This is especially so for the increased *variant* PSM observation from the *decoy variant strategy*. While that cannot be presently concluded, the concept of the decoy variants can be critiqued.

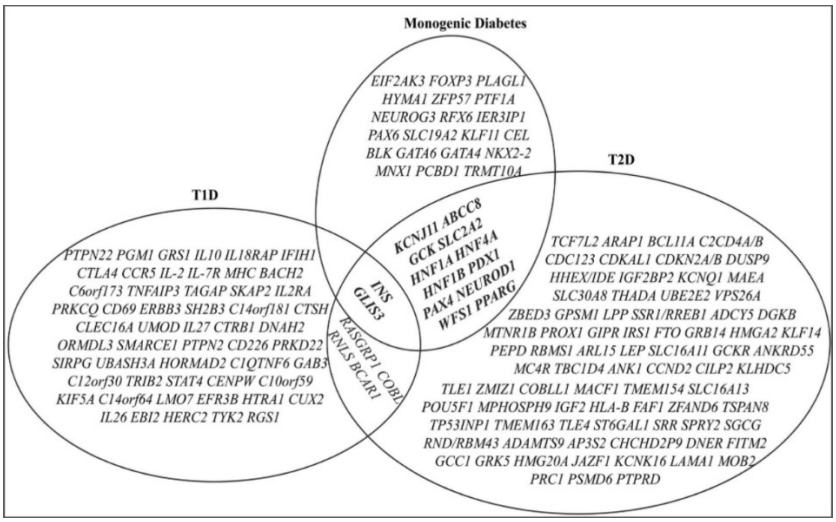


Figure 4.1 Diagram of genes associated with diabetes subtypes. The protein products of multiple genes have been identified as biomarkers of diabetes types: Type 2 diabetes, Type 1 diabetes and Monogenic. Some genes are known to share associated with other diabetes subtypes. This figure is unmodified from Yang and Chan (2016).

4.3 Decoy variant concept

A null model is a baseline reference used in data analytics to test hypotheses (see Chapter 1.5.2). It represents a simplistic assumption that the effect of variables is absent or unimportant, arising by chance. This is the inference of hypothesis testing concerning multiple unrelated variable types. The decoy PSMs represent the null model in the TDA applied to classification assignments. Decoy spectra are specifically designed to simulate random or false matches, while the 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 method

of creating the *decoy variant* PSMs hence differed from the standard approach of reversing and shuffling the sequences, as the *decoy variant* PSMs were required to simulate deliberate 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 cumulative count of *canonical* PSMs 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, it is not a new concept. Earlier approaches retained top scoring PSMs annotated by search engines as the positive class label while the remaining received a negative class. 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 dataset, unlike in this work which used a self-trainer system. A study by Choong and Sung (2021) explored eight variations of decoy sequences to investigated how each implementation method influences 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 accuracy of spectra ranking, and it is possible that the *decoy variant* concept has 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 the course of their development. Considering this, an overlap between the *true variant* PSMs and *decoy variant* PSMs was expected be observed from the histograms (**Figure 3.7(right)**). Instead, the *decoy variant* PSMs were clearly skewing to the right towards a p(-1) 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 is in favour to support the *decoy variant* concept.

The use of the *decoy variant* concept at its current development should however be cautioned against and reevaluated. Arguably it is not a reliant method, due to the caveats of proteogenomics previously stating that the mass spectrometry weights, and search engine scoring functions were not adjusted to account for classification of pure variant PSMs.

Furthermore, the *decoy variant* peptides were created with SNP alterations only, and other structural variants such as frameshift and indels (see Chapter 1.3.2.1) are not included. This may explain why no *variant* PSM was observed for HNF1A-MODY3 in the MODY expression plot (**Figure 3.10**) as explained in the previous chapter. These factors should be considered in the future to improve reliability of this approach.

One other acknowledgement is that the *decoy variant* PSMs were proportionately lower than the *decoy sequence* PSMs. To reiterate, iPSC cell line 03 contained 1 344 *decoy variant* PSMs and 9 854 *decoy sequence* PSMs that 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 of significant spectra from the *decoy variant* PSM strategy (**Figure 3.10[b]**) 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 imbalanced datasets is of importance with 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) stated two classification conditions for its process: 1) the proportion of class types should be roughly equal to ensure well-balanced discrimination, 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 by using the *StratifiedKFold* method under Scikit-learns framework. The first condition, however, is not always possible due to duplicate and poor-quality decoy spectra being removed during database creation, and PSM frequency being dependent on search engine annotation. Previously in Chapter 4.3, the numerous existing methods to create decoy spectra were highlighted, therefore influencing factors can impact spectra frequency. In task 2, the iPSC PSM datasets were divided into four class types , thus reducing spectra proportions during classification. The logic of the TDA, implies that the frequency of decoy PSMs represents the proportion of incorrect matches of the target spectra (false positives) (Elias and Gygi, 2007, 2010). This is debated in the community as shown by the rise in alterative error rate metrics accounting for PSM datasets with disproportionate target and decoy class types (see Chapter 1.5.3 and 1.5.4).

Imbalanced datasets are a phenomenon in data analytics which 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 and the other representing 20%. This imbalance poses an issue for ML problems, as learning algorithms would have a bias towards the majority class. Error rate metrics are also vulnerable to overestimate confidence of data points. Typically, computational processes and error metrics can be adjusted to account for such occurrences, however whether this imbalance poses an issue for determining fitness of PSMs is assessed in the succeeding chapters. The prediction performance on the decision tree ensemble models is evaluated next, regarding class imbalance and overall functionality for each implementation task. Then lastly, the statistical inference and error rate metrics used in this work is evaluated for reporting PSM significance.

4.4.1 Machine Learning

4.4.1.1 Imbalanced learning

Imbalanced datasets are a general concern for semi-supervised learning (Huynh *et al*., 2021) tasks due to the learning algorithms being inaccurately trained causing poorly predicted class labels onto unseen data points. Class label prediction accuracy is irrelevant to PSM processing as the goal is to rank spectra in order of annotation quality. This is achieved by obtaining a probability metric describing the likelihood a spectrum is a correct or incorrect match. ML classification algorithms trained with labelled samples issue a probability test statistic onto unlabelled testing samples which are more informative. In this work, the *prediction probability* p(-1) was employed under the Scikit-learn framework. 'Prediction accuracy' in this sense is 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) 0.5 are still acceptable as significant peptides. The decoy PSMs, which represent random peptide matches 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 is how significance is inferred. Imbalanced PSM class types can influence the learning algorithm to learn patterns of the majority class type and overestimate spectra probabilities. The performance of the decision tree architectures is further discussed in reference to the qualities of the PSM datasets of both task 1 and task 2.

The distribution of spectra class types was diverse for the iPSC files in task two (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 the *variant* PSMs were the main interest, they were combined with the *canonical* PSMs as the target class type for each decoy PSM classification strategy. Considering this, it may be expected that that canonical PSMs were favoured by majority leading to a biased classification. Acknowledging the *variant* PSMs alone, the learning models could favour the decoy PSMs considering the large difference in their frequencies. However, a good p(-1) distribution was observed for all PSM class types as viewed from the histograms (**Figure 3.7**). Interestingly, the *decoy variant* PSM strategy has similar distribution trends to that of the *decoy sequence* PSM strategy despite the lesser frequency of spectra and neither were harshly skewing to the right to indicate bias performance.

The resolution of the *variant* PSMs in the *decoy variant* strategy appear to be improved when compared to the *decoy sequence* strategy which offered more negative samples for classification and no biased performance was observed. The proficiency of decision tree ensemble models requires for there to be enough data points for a learning algorithm to efficiently sample and aggregate an accurate average (see Chapter 1.8). Including the *canonical* PSMs as the target class type in task 2 may have aided the classification process of the ExtraTrees architecture. This is noted as the 3-fold cross-validation of NT divided the training samples in smaller subsets, and had only variant PSMs been used, classification may unreliable.

The ExtraTrees architecture, operates by randomly setting thresholds for each feature variable and creates tree nodes with the best split (see Chapter 1.8.2.1). In task 2, the proteogenomic pipeline included a feature extraction step, which combined with the increased randomness from ExtraTrees may have resolved any occurrence resulting from the imbalance. If the majority class type was favoured, the target PSMs for both the *canonical* and *variant* class types, would be heavily skewed to the left. Instead, observing spectra across each interval of the p(-1) distribution reassures the decision-making process of ExtraTrees. This observation may indirectly promote the feature extraction process, and it would be interesting to rerun the iPSC classification without the additional features to compare its influence. Furthermore, the interval

distribution such as that observed in the iPSC histograms informatively describes the quality of the spectra from the collective scoring functions. This may not have been achieved had a *Boosting* architecture been selected for this process.

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 PSM was roughly 1:4, respectively. The histograms of task 1 (**Figure 3.3**) displayed the distribution trends of the target and decoy spectra which were overall positive for each of the decision tree ensemble models, despite displaying unique outputs for the *Bagging* and *Boosting* model types. The spectra of the *Boosting* models; Gradient Boosting, Histogram-gradient Boosting and XGBoost, technically skewed correctly as seen in the histograms of **Figure 3.3[b, c, e]**, however a distinct divide in spectra distribution is also observed. While this is technically an unbiased performance it is questionable if there is actual benefit to collecting target PSMs with a $p(-1)$ of 0.0. It may be viewed as over optimistic and uninformative of the 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 effort to minimize errors. If the logic of the TDA is considered, the *Boosting* models appeared to deliberately collect random target PSMs to the frequency of decoy PSMs at similar proportions 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 the performance is dependable on construction of the architecture. Assuming the significant target PSMs are correct, perhaps the idea of ranking PSMs for post-processing can be reconsidered regarding *Boosting* models.

Random Forest or *Bagging* models in general, appears to be suitable for informative PSM processing as viewed in the histograms of task 1 (**Figure 3.3[a, c]**). Although Random Forest produced the least amount of confidently predicted target spectra (**Table 3.2**), the hyperparameter tuning conducted in NT may not have been suitable for this classification task. While creating NT, Random Forest proved to be the most computationally expensive architecture, however certain default variables were producing unreliable spectra distributions that were seemingly biased. The variables for hyperparameter tuning of Random Forest (**Table 2.4**) were purposefully selected to increase randomness of the algorithm, however not extensively compared to ExtraTrees, whose run time was the quickest and easier to test more parameter variables. This is to not rule out Random Forest as a PSM classifier and to note the

benefit of increasing randomness for better spectra distribution. 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 robustness of their algorithms as compared to linear processes of SVMs. Although the work of Spivak *et al*. (2009) noted that ML architectures do not generalise well across different platforms and experimental conditions, and thereby parameter adjustment is requirement for new training samples. After reviewing NT, it was indeed necessary and altered the possible outcomes of this work (in the case for Random Forest), and 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, and 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 should be able to optimally process the spectra to the best of its ability. This is further influential for the inference of error rate metrics which rely of spectra ranking. Subsequently this may explain the increase in random target spectra with a 1% FDR observed in the final classification of *Boosting* models (**Figure 3.6**) which had 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 displayed excellent spectra distribution however the true behaviour of the target and decoy PSMs were revealed by the histograms and FDR analysis. While any ML architecture has potential as a PSM processor, the data analytics of this proteogenomic step does prove to requires alternative approaches to that of classical ML techniques. Arguable, this is more of a matter concerning thresholds and error rates, which ultimately infers spectra significance beyond the baseline measure of probability test statistics.

4.4.2 Statistical inference

The statistical realm is filled with controversies, made so by statisticians debating the rationality of the 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 prioritizing the

purpose of probability as a means to characterize 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, that evaluation of a hypothesis metric is not a conclusive statement of whether it is good or bad, and rather a demonstration of a phenomenon as stated by Sir Ronald. A. Fisher (1955).

Computational techniques and the rise of high-throughput technologies are able to identify patterns in data, regardless of the algorithm being a correct or incorrect tool for a 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 NT attempts to improve the occurrence of false predictions by using *cross-validation* to operate as a self-trainer (see Chapter 1.6.2.2), it is acknowledged that the classification capabilities of NT are nuanced to PSM datasets used in this work. Results are vulnerable to differ accordingly to unaccounted for characteristics, such as size, scoring functions and proportion of decoy PSMs. It is therefore necessary to gauge the rationality of the statistical metrics.

The concept of the TDA to infer confusion matrix metrics is a two-sided debate in the community. On one side a side estimating a substantial proportion of false positives endorses the reliability of significant true peptide matches of the target spectra (Elias and Gygi, 2007; Wang *et al*., 2009). However, on the other, it is further noted that not all spectral identification tasks are TDA compliant (Shen *et al.*, 2009; Gupta *et al.,* 2011). As stated earlier, roughly 10 – 50 % of peptides from search engines are identified, therefore it is of interest to not further promote the loss of true positive spectra during PSM processing. While there are different aspects that influence the occurrence of false positives a safeguard feature 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 is 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. There are other statistical metrics to infer accuracy for classical ML methods, of which two common tests are the F1-score (Melo *et al.*, 2016) and Matthews correlation coefficient (Chicco and Jurman, 2023). To include more metrics can further validate a model's performance (Pang *et al*., 2021). Although an opposing study by Caelen (2017) observed each ML performance metric infers error rates indifferently and suggests data points are required to be generalized to appropriately adjust the confusion matrix using Bayesian tactics. The ROC and PR curves provided a baseline observation of the distribution of classified spectra for each ML architecture. In the previous chapter it was discussed that classical ML methods are nuanced to PSM classification and alternative strategies are required to be specifically defined for its processes. Whether these extended performance metrics would benefit ML evaluation for PSM processing is questionable as it appears unnecessary. However, they should not be disregarded as additional statistics can effectively validate ML performance 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 proportion which is also debated (Gupta *et al*., 2011; Cooper, 2012). To curb these issue, various adaptions to the FDR are being considered, however, there is no single agreed upon method (Aggarwal and Yadav, 2016). Either formulae type is accepted as the reports of its effectiveness is situational to the research hypothesis. Subsequently, its correlation to a statistical inference is subjective. It is possible for spectra to receive a probability test statistic representing a random match from 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** in which the p(-1) intervals of predicted random and non-random target spectra within a 1% FDR are represented. The figure showcases the p(-1) interval composition of the final classified PSM dataset and does not represent the 1% FDR target spectra selected for the training processes of NT. However, this inference implies that random target PSMs were likely to be included in the training processes although the purpose of this method served to optimally train the learning models.

These are unattractive qualities with regards to peptide identification tasks, however, the TDA for FDR estimation is still a *de facto* method widely used in proteomic studies (Ebadi *et al.,* 2023) including this work. The application of the FDR is useful to isolate a generous proportion of confident spectra matches globally, and in **Figure 3.6**, a larger proportion of non-random target PSMs is observed for each of the decision tree ensemble architectures. Although the p(-1) scores are a baseline measure, a second threshold may be applied in the future to control this purpose. Alternatively, a local error rate estimation such as the PEP can be applied to improve resolution of significant spectra for training the learning models.

4.4.2.3 Posterior error probability

There are various strategies investigated to estimate PEP values through Bayesian inference for mass-spectrometry spectra and search engines peptide matches. The potential of PEP measurements is explored in the work of Zhang *et al*. (2008) in which a filtering model based on Bayesian inference is proposed. The framework of the Bayesian approach is reviewed in the work of Alterovitz *et al*. (2007) concerning the likelihood of events and parameter control for disease diagnostics and peptide identification. The opposing work of Serang *et al*. (2010), revealed the sensitivity of posterior probabilities as insufficient for error rate inference concerning peptide-to-protein identification tasks. A protein may contain a single peptide with a significant PEP estimate while the remaining score poorly. In the same work it was stated to be a concern regarding spectra matching to large datasets vulnerable to an increase of annotation errors and the use of post-processing tools overall was discredited.

In this work a novel approach to PEP estimates was suggested which entailed a 'container' method (see Chapter 2.8.2.5) to include spectra within a 1% p(-1) range, as an alternative to the method as proposed in Percolator (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 which matched 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 for decoy spectra to obtain probability scores indicating non-random matches at a low frequency and larger datasets are susceptible to a 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. The FDR is vulnerable to include false positives in its estimates (**Figure 2.5**), and requires scaling for imbalanced decoy spectra as mention in Chapter 1.5.3. In opposition, the PEP, solely relies on the power of the classifier 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 as 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 spectra 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 degree of randomness (Hung *et al.*, 1997)(see Chapter 2.3.2). Additionally, neither are uniform estimates meaning they are subjective to change with estimation method and dataset size for example.

The p-value is under criticism from statisticians against its use as a measurement of evidence. This is due to the inference of thresholds implying statistical significance from a preliminary null hypothesis rejection (H_0) (see Chapter 1.5.2) region which are generally at 0.05 or less, on assumption the power of the probability metric is reliable and accurate (Mayo, 2019). Low p-value estimates representing significance are therefore judged as misguided. The implication being that real errors are underestimated for practical application and represent the dataset used for estimation only (Mayo and Cox, 2006; Andrade, 2019). This is understood with regards to the p(-1) values owing to the self-training classification procedure of NT implying there is no intention to apply a trained learning algorithm onto new PSM datasets. Unlike the p-value which infers 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 cumulative count of **Figure 3.5** which provides an inclusive comparison that observed ExtraTrees as the better model. The ability to minimally recognize spectra with a p(-1) below 0.5 guards against unnecessary loss of PSMs during processing and the performance of PSM processing can be reviewed completely as in **Figure 3.9** for the iPSC PSM datasets. 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 as probability ranking is prioritised.

Typically, confidence intervals are used to compare estimates of various conditions and provide a range of values with a strong likelihood of containing the true threshold to infer significance (Neyman, 1941). A p(-1) threshold of 0.01 (1%) 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). The cumulative counts of **Figure 3.4** served 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 view if 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 are spectra ranking and PSM-to-protein inference as the succeeding step of the proteogenomic workflow. The usage of p(-1) and error metrics circles back to the arguments posed in the previous Chapter 4.4.1 concerning the potential of ML applications. The choice of using decision tree ensemble models is additionally beneficial 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 point-of-view (Radzvilas *et al.*, 2021), however the p(-1) values and statistical metrics applied in this work did serve its purpose as a tool to process PSMs and its potential should not be discouraged.

5: Conclusion

Peptide identification is heavily nuanced being subjective to false positives entering the annotation space. 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 reprocessing 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 uniform in nature (Käll *et al*., 2007) and as the quality is uncertain there is question as to which ML architecture offers the appropriate robustness for the task of classification. Specifically, the binary classification of target and decoy PSMs which provides the ability of error rate estimation. While the TDA is a classic technique for null model distribution, the use of ML makes it difficult to comment whether or not it is the most appropriate strategy. ML is a powerful tool which combined with hyperparameter tuning is purposefully designed to provide optimised predictions to the best of the learning algorithms ability. 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 prediction performance of 5 architectures of which ExtraTrees 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 ExtraTrees from task 1, although positive, are not transferrable to task 2 and its selection is based on a heuristic observation. The size of the iPSC PSM datasets were significantly smaller than that of task 1 (see Chapter 4.4) and the occurrence of class imbalance is typical for 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.

Separating the spectra into four individual class types for task 2 also 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 datapoint leads to 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 SVM, is known to be suitable for smaller datasets and is likewise suited for high dimensional classification. Validation of NT should be compared to Percolator and other existing tools in future works for legitimate reporting. Decision trees fall in the median for data analytics and has proved itself as a well-rounded learning style for various classification scenarios. 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 system* (**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. To bypass the ranking requirement opens the door for a filtering strategy instead which can focus on refining probability estimates and collect quality PSMs.

Overall, NT was able to complete the task of classifying PSMs as a self-training system. 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 nor improve existing methods but to rather 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 into the script (see Chapter 2.8.2.5).

Are there practical application complications?

Whether or not the outcome of this research is 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. Furthermore, there are considerable difference 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). Missing transcription factors does not impact gene expression of iPSCs; however, it does lead to incomplete findings of specific cells such as pancreatic β-cells, which does limit practical application (Efrat, 2021). It is also assumed that theoretical databases are representative of varying populations, and as this is unlikely, these cases are subjective to missing or indifferent data (see Chapter 1.3.4). The complexities of patient clinical data may in fact require a processing structure that is both rigid yet flexible enough to account for anomalies such as variant peptides. The nature of this work paints a picture as an example 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) highlighted that modified peptides account for up to 50% of false positive identifications owing 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 a correct peptide match and imposed error rates and thresholds to achieve the task. Knowing this, it is possible the statistical direction taken in this work is misguided. The work of Savitski *et al*. (2015) proposes the 'picked' FDR approach which tracks the decoy version of target spectra and compares the individual probability scores for significance. Anders *et al*. (2021) likewise used FDR estimates to prove unannotated peptides, likely due to variation, occurring 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. Aggregated statistics is accomplished with ML techniques and the PEP 'container' method of this work; however, PSM-to-protein inference was not conducted. Furthermore, the decoy variant concept task did not evaluate unknown variant peptides to the depth of the studies mentioned. 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

Peptide identification in itself is heavily nuanced being subjective to false positives entering the annotation space from varying sources. Considering this overview of proteogenomics, it is fair to state that processing PSMs is inherently a statistical problem. However, statistical inference is nuanced in itself. Serang and Käll (2015) states the key to improving peptide classification to incorporate more statistical measure. Upon reviewing the results from this work, however, it is questioned if it is misguided to work towards highly accurate error rate estimations. The thoughtful mind of Dyson (2004) recalled a meeting with the esteemed Enrico Fermi to critique certain results in theoretical physics. Fermi prescribed that formulae should either a be precise and 'self-consistent' or rather defines a system with a 'clear physical picture', and remarked on 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 encapsulates an idea and sets a goal that is beneficial towards the application of statistics. It with this mindset this work hopes to contribute to the foundation of proteogenomic research for precision medicine, amongst those of likeness, and drive new 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.