Dissertation on

"Title of the project"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE17CS490B – Capstone Project Phase - 2

Submitted by:

Name 1	**<SRN 1>**
Name 2	**<SRN 2>**
Name 3	**<SRN 3>**
Name 4	**<SRN 4>**

Under the guidance of

Prof. Guide Name
Designation
PES University

January - May 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Title of the project'

is a bonafide work carried out by

Name 1	**<SRN 1>**
Name 2	**<SRN 2>**
Name 3	**<SRN 3>**
Name 4	**<SRN 4>**

in partial fulfilment for the completion of eighth semester Capstone Project Phase - 2 (UE17CS490B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2021 – May. 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature **<Name of the Guide>** Designation	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"Title of the project"** has been carried out by us under the guidance of <Prof. Guide Name, Designation> and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

<SRN 1>	**<Name 1>**	**<Signature>**
<SRN 2>	**<Name 2>**	**<Signature>**
<SRN 3>	**<Name 3>**	**<Signature>**
<SRN 4>	**<Name 4>**	**<Signature>**

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. <Guide Name>, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE17CS490B - Capstone Project Phase – 2.

I am grateful to the project coordinators, Prof. <Project Coordinators Name>, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

With the advent of the sphere of bioinformatics, of there is a new confluence engineering and biology that has led to remarkable changes in the way researches approach difficult and time consuming problems. The pharmaceutical industry has highly benefitted from this. Research and opened new gates with regards to how we look at drugs and their applications. With this window of opportunity, comes a big challenge: information on these drugs are not computationally friendly. In this work we take advantage of the structural and functional aspects of a drug to generate a drug embedding, an accurate representation which will serve as a gateway to other bio informatic applications like drug discovery, drug target interactions, drug reprofiling and drug repositioning. We employ machine learning techniques to learn the embeddings of the drugs and validate their efficacy by testing against a known biological classification. Additionally, we are aiming to solve an important bioinformatic application which is: early and accurate identification of potential adverse drug reactions (ADRs) for combined medication which is vital for public health. Since most clinical trials focus on a single drug and its therapeutic effects, most drug-drug interaction induced ADRs go unnoticed until the drugs are actually approved. This has been one of the top 10 reasons on death in United States according to the studies. To solve the same problem we employ various machine learning models to predict drug-drug induced ADRs using the structural and functional characteristics of drugs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The field of bioinformatics has been revolutionizing the way we look at drugs and their use cases. Bioinformatics is an interdisciplinary field at the confluence of biology and engineering that develops computational methods for analysing biological data. This emerging field provides useful information for the analysis of the molecular basis of diseases, the search for target proteins, and the analysis of the effect of drug therapies. It has also been instrumental in the advance of disease diagnosis and prognosis as well as therapy selection. With the wide range of life altering use cases the field presents to the world, it has become indispensable in today's world.

There has been a surge of computational breakthroughs to support the ever-evolving field over the years thanks to the current advancements that make it possible to look at drugs from all aspects- structural and functional.

Bioinformatic applications like drug discovery, drug target interactions, drug reprofiling and drug repositioning call for years of research and huge monetary investments from pharmacological companies. In the past decade there has been an enormous increase of financial investments in pharmaceutical R&D. This increase in investment has not reflected in the number of newly approved drugs. Drug approval process is very tedious and picky. This is the motive behind using computation to aide these conventional wet lab research is to increase the number of drug approvals or find new uses for already approved drugs, which will save a lot of time and monetary expense.

In the cut-throat commercial drug industry, designing chemicals with desired characteristics is a bottleneck in the development of new drugs. Drug design is still driven by intuition, chance and years of research. However, processing of structural and functional descriptors of drugs can lead to at an expedited development. This requires good 'Representation Learning' which is a precursor to help these descriptors become computationally friendly and meaningful.

Existing structural and functional indicators of drugs are not in consumable formats for mathematical processing. They are categorical, mostly through manual process. In this work, we explore the idea of drug embedding, a representation of a drug. The performance of machine learning methods relies heavily on the input data. Therefore, 'Representation Learning'- the design of data pre-processing and data transformation is of great concern to ensure that the data representation can support machine learning algorithms. This is one of the biggest challenge of bioinformatics and with our work we attempt to try to find a solution to this.

An important application of bioinformatics is classifying drugs into their respective adverse effects. Adverse drug reactions (ADRs) are unintended and harmful reactions caused by normal uses of drugs. ADRs caused by individual drugs and drug combinations constitute one of the top 10 causes of death in the United States. With an increase in the number of drugs used in pharmacology, there has been an increase in the number of ADRs due to drug-drug interactions. The issue with drug-drug induces ADRs is that they are not observed in the clinical trials which usually focus on a single drug and its therapeutic effect against a particular disease or condition

Clinical trials are conducted for each drug before approval of the drug which is conducted in 3 phases as described: Phase I clinical trials investigate the safety profile of a candidate drug on a small group of volunteers. Phase II trials evaluate its safety and efficacy in a larger group of volunteers. Finally, in Phase III clinical trials, these effects and ADRs are monitored in a large group of selected volunteers. Drugs that successfully pass these hurdles can then be approved for general clinical use. In fact, although most ADRs induced by individual drugs have been discovered and carefully monitored in clinical trials before drug approval, information on nearly all ADRs induced by drug-drug interactions (DDIs) has been generated after drug approval. This poses continuous and serious risks to patients' health.

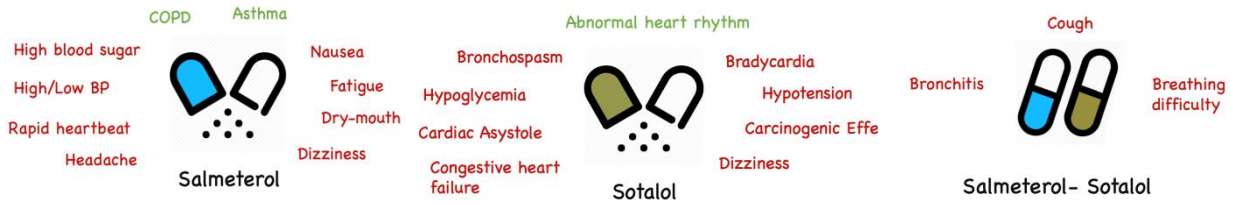

Figure 1: ADRs caused by drugs when used separately as compared to used together

Furthermore, in most scenarios the ADRs observed by individual drugs completely differs from the ADRs observed when both the drugs are taken together. Figure 1 describes one such example where the drug Salmeterol and Sotalol cause totally different ADRs when taken together as compared to when taken individually.

CHAPTER 2

PROBLEM STATEMENT

Wet lab testing has always been a costly and time intensive affair. Computation has attempted to bridge the gap between wet lab testing and timely results. Computational techniques can nudge researchers in the right direction without costing them a fortune of their time and money. One such example is drug repurposing. There exists millions of chemical compounds each with a specific set of properties that give it an appropriate use case. Finding drugs correctly suited for different requirements is generally considered a job for chemical laboratories where compounds are tested for various properties. It is difficult to accomplish this without extensive testing because drugs cannot be considered similar despite superficial similarities such as common functional groups. Two drugs with similar structures may not be suitable for the same application.

A solution for determining similarities between drugs stems from examining different legs of the drug like its structure and gene expressions. The structure of a drug can be documented by a simple string of atoms connected with bonds. A gene expression is obtained by applying drugs to RNA strands and observing proteins released. Two drugs that can be used for similar applications may have some inherently common patterns within these structural strings or gene expressions.

Since these inputs are quite varied and are not computationally friendly a pre cursor to such heavy bioinformatics problem would be to find a suitable multi-purpose representation of a drug. This embedding would serve as a gateway to determine different drug-target interactions, similarities between drugs and bases for drug discovery.

The problem that we are hoping to solve through our research for this project is to try to extract inherent patterns from structural and functional information on a drug using AI. Deep learning architectures will be employed to embed this information such that the embeddings of different drugs will be able to partake in many bioinformatic applications.

We analyse different types of information available on drugs and their representation, understand the nuances of the data to glean the most suitable dataset for our problem and compare different deep learning methods to arrive at an architecture that will give an embedding which will capture the details of the drug and can be scaled to different applications.

Most studies of ADR prediction are based on using drug-attributes based on chemical features of drugs like use of ECFP on organ-based ADR prediction, use of CACTVS and SCCA or drug-attributes based on biological features of attributes like Drugs with similar ADR profiles tend to have similar protein target.

On the other hand, work on ADR caused by combined medication, which occurs when individually safe drugs interact pharmacokinetically or pharmacodynamically, is limited to data mining and uses no X-omic study as per our literature study. Pharmacologic databases provide rich useful resources for identifying ADRs of combined medication, owing to the openness, high data quality, and coverage for both novel and rare drugs.

In this study we used machine learning models with the transcriptomic data from L1000 to predict the risk of ADR on combined medication of two drugs. We also performed a comparison of the predictive performances of five state-of-the-art and perhaps most commonly employed ML algorithms, including Decision Trees, Random Forests, Naïve Bayes, Logistic Regression as well as Stochastic Gradient Descent.

CHAPTER 3

LITERATURE SURVEY

In this chapter, we present the current knowledge of the area and review substantial findings that help shape, inform and reform our study.

3.1 Background on Embeddings and Drug Repurposing

This section details the papers read to gain information on background, data used, and the current methodologies being used in the field of bioinformatics.

3.1.1 Seq2seq Fingerprint: An Unsupervised Deep Molecular Embedding for Drug Discovery [1]

Zheng Xu et. Al

In this paper, the authors propose a novel unsupervised molecular embedding method (unsupervised seq2seq fingerprint method), to simply translate a SMILE string into itself. The intermediate vector generated can be considered as a continuous feature vector for each molecule of a drug to perform classification.

Architecture of the network: GRU cell is used, instead of LSTM, to accelerate the training process. Attention Mechanism is employed to centralize the finger-print space. Dropout layer is added to overcome the over-fitting.

The benefits of the seq2seq fingerprint are three folds:

1. The training phase of seq2seq fingerprint is unsupervised: completely label-free.
2. It is data-driven, eliminating the reliance on expert's subjective knowledge.
3. Unlabelled data is unlimited, the deep learning network can be trained well.

The author's network takes a very long time to train. The length of the embeddings are in the set of (512, 784, 1024) which is quite large and requires more computation and robust hardware.

3.1.2 Predicting New Molecular Targets for Known Drugs [2]

Michael J. Keiser et. al

The authors used a statistics-based chemo-informatics approach to anticipate associations between drug and targets. Most drugs had no significant similarities to most ligand sets. However, 6,928 pairs of drugs pairs of drugs and ligand sets were similar, with expectation values (E-values) better than 1×10^-10.

Not all the new off-targets predicted by their research are unanticipated. A third of their drugs which were predicted active on their off-targets are false positives when verified with wet lab tests.

3.1.3 Drug Target Identification Using Side-Effect Similarity [3]

Monica Campillos et. Al

The authors explore the relationship between drugs and their protein targets. It is observed through this research that there is an inverse correlation between side-effect frequency and the likelihood of two drugs to sharing a protein target. The authors tested the predictive power of this side-effect similarity measure on their reference set of 502 drugs with known human targets and observed a clear correlation between side-effect similarity and the likelihood that two drugs share a protein target. The authors observed in their reference set that chemically similar drugs [according to the two-dimensional (2D) Tanimoto chemical similarity score] are likely to have the same targets.

This study presented high accuracies (88% AUC) on the enzyme dataset however, the similarity metric was based off of only 502 drugs.

3.2 Generating Embeddings for SMILES

This section details the survey of the current state-of-the-art methods for various bioinformatic applications that employ SMILEs as their primary dataset.

3.2.1 SMILES2Vec: An Interpretable General-Purpose Deep Neural Network for Predicting Chemical Properties [4]

Garrett B. Goh et. Al

Based off of NLP techniques which gather grammar related features not explicitly but through training a network, so does Smiles2Vec train an RNN model to learn and understand the grammar in SMILEs. Through training, the network is expected to learn features that could be useful in understanding the chemical properties of the molecule. Embeddings are tested for properties like solubility, activity, toxicity and solvation energy.

The results presented by this paper show embeddings with a good representation of the molecule. Prediction of functional properties have not been attempted by this paper and could be a possible enhancement. Other feature gathering models like CNN could be utilised to better the quality of the embeddings.

3.2.2 SPVec: A Word2vec-Inspired Feature Drug-Target Interaction Prediction [5]

Yu-Fang Zhang et. Al

SPVec uses the word2vec model to embed smiles into embeddings that represent the molecule. The skip gram model coupled with a negative-sampling has been used to achieve a robust model that creates representative smiles. The learned embeddings are tested for their ability to predict various chemical properties using algorithms like random forest, gradient boosted decision trees and DNNs. Promising results have been presented.

While the results demonstrated in this paper showcase a decent ability of the embeddings to predict chemical properties, the functional information held by the embedding has not been tested and could be a possible future enhancement.

3.2.3 Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition [6]

Sabrina Jaeger et. Al

Employing the core strategies in natural language processing, the paper describes mol2vec a methodology for converting a molecule into a representative vector. Using unsupervised machine learning techniques, each substructure in the molecule is encoded into a vector using the skipgram model. Averaging the substructure embeddings gives a molecule level embedding which has been used to represent the molecule and it's efficacy has been verified using the embedding's ability to predict it's properties.

The representative ability of embeddings of substructures could be further observed by changing the radius at which substructures are obtained. Novel embedding techniques other than skip gram could be tested for better performance.

3.2.4 SWeeP: Representing Large Biological Sequences Datasets in Compact Vectors [7]

De Pierri CR, et. al

The paper looks towards representing various biological sequences like gene sequences in a compact form while retaining all or most of the information held in the original embeddings. The embedding is created by locating indices in the sequence to make a high dimensional vector which is later projected to a lower dimensional space to obtain a compact vector.

Spaced word projections works well but it's purely mathematical approach could be combined with machine learning techniques to further enhance the quality of the embeddings obtained.

3.3 Generating Embeddings for Gene Expression

This section details the survey of the current state-of-the-art methods for various bioinformatic applications that employ transcriptomic data as their primary dataset.

3.3.1 Drug Repurposing Using Deep Embeddings of Gene Expression Profiles [8]

Yoni Donner et. Al

Here, the authors report a new method for measuring functional similarity between drugs based on gene expression data using deep neural networks to learn an embedding that substantially denoises expression data, making replicates of the same compound more similar.

The method uses unsupervised deep learning method with each layer being a densely connected with SELU activation. Once the embeddings were retrieved, their method could identify drugs with shared therapeutic and biological targets even when the compounds were structurally dissimilar, thereby revealing previously unreported functional relationships between compounds. Their method presented good results, 95% accuracy for ATC classes.

Biologically, they did not account for the fact that a drug acting on a particular cell line would be comparable to the same drug acting on another cell line.

3.3.2 A Large-Scale Gene Expression Intensity-Based Similarity Metric for Drug Repositioning [9]

Chen-Tsung Huang et. al

Biological systems like gene expression data are usually robust to perturbations and the current similarity techniques that are in use, don't capture this. The authors propose that intensity-based similarity metric surpasses other standard metrics in drug clustering. This metric was applied to compare thousands of compounds for drug repurposing. The new metric emphasizes the genes exhibiting the greatest changes in expression in response to perturbation.

3.3.3 Gene2vec: Distributed Representation of Genes Based on Co-Expression [10]

Jingcheng Du et. al

The authors proposed a method that utilizes gene co-expression to generate a distributed representation of genes. The distributed representation of genes could be useful for more bioinformatics applications. Inspired by the success of word embedding, they intend to produce an embedding of genes. Since genes don't have an equivalent of a 'sentence', they use the notion of co-expression. This is analogy of the concept of co-occurrence in natural languages.

The authors were successful in creating an embedding, however for the purposes of training gene co expression can prove to be a very large matrix which is not easily managed.

3.3.4 Deep Learning Applications for Predicting Pharmacological Properties of Drugs and Drug Repurposing Using Transcriptomic Data [11]

Alexander Aliper et. al

In this paper the authors demonstrate classification of transcriptomic data into therapeutic categories. They used the perturbation samples of 678 drugs across A549, MCF-7, and PC-3 cell lines from the LINCS Project and linked those to 12 therapeutic use categories derived from MeSH. They were able to prove that gene expression data needs a deep learning network to classify accurately. A machine learning model like SVM will not suffice.

Given the skewed nature of the MeSH dataset as well as the small number of drugs (678) used to train the network. It may not scale well.

3.3.5 Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics [12]

Ehsaneddin Asgari et. al

In this method, the authors present a representational learning method to generate continuous vector representation for drugs using an artificial neural network. To evaluate this method, they apply it in classification of 324,018 protein sequences obtained from Swiss-Prot belonging to 7,027 protein families, where an average family classification accuracy of 93%±0.06% is obtained, outperforming existing family classification methods. The resulting embeddings prove to cluster protein sequences with well-defined boundaries.

3.3.6 Drug Repositioning: A Machine-Learning Approach Through Data Integration [13]

Francesco Napolitano et. al

The noisiness and scarcity of the gene expression are important limitations to using solely gene expression in computation. The authors propose a novel computational approach to predict drug repositioning based on integrating multiple layers of information:

i) Distances of the drugs based on how similar their chemical structures are.
ii) Closeness of their targets within the protein-protein interaction network
iii) Correlation of the gene expression after treatment.

Their classifier reaches high accuracy levels (78%), but this method may not be extended to other classification systems as the target closeness information for this particular dataset can be obtained

from protein-protein interaction network. There may not be similar networks that can provide information about other targets.

3.4 Adverse Drug Reactions

This section details the survey of the current state-of-the-art methods for bioinformatic application of classifying drugs into their adverse effects.

3.4.1 Systematic Drug Repositioning Based on Clinical Side- Effects [14]

Lun Yang, Pankaj Agarwal

Clinical side-effects (SEs) provide a human phenotypic profile for the drug, and this profile can suggest additional disease indications. The authors extracted 3,175 SE-disease relationships by combining the SE-drug relationships from drug labels and the drug-disease relationships from PharmGKB.

The AUC was above 0.8 in 92% of these models. The method was extended to predict indications for clinical compounds, 36% of the models achieved AUC above 0.7. The authors built Naive Bayes models to predict indications for 145 diseases using the SEs as features. The AUC was above 0.8 in 92% of these models.

3.4.2 Predicting ADR of Combined Medication from Heterogeneous Pharmacologic Databases [15]

Zheng Y et. al

For the task of predicting ADRs more effectively, the authors used highly credible negative samples (HCNS-ADR), fused heterogeneous information from various databases and represent each drug as a multi-dimensional vector according to its chemical substructures, target proteins, substituents, and

related pathways first. Then, a drug-pair vector is obtained by appending the vector of one drug to the other.

Next, they construct a drug-disease-gene network and devise a scoring method to measure the interaction probability of every drug pair via network analysis. Drug pairs with lower interaction probability are preferentially selected as negative samples. They did PCA for negative and positive samples.

Finally, a classifier is built for each ADR using its positive and negative samples with reduced dimensions. The models showed significant improvement with HCN-ADR samples.

The authors were able to get higher performance using the HCN-ADR samples created.

3.4.3 Predicting Adverse Drug Reactions Through Interpretable Deep Learning Framework [16]

Sanjoy Dey et. al

I In general, the basic steps of ADR prediction based on structural information can be broken down into two stages. First, each drug molecule is represented in a suitable feature vector based on its chemical structure. Second, a machine learning algorithm is applied on the resulting feature space to predict ADRs. Since, there is not a lot of work done regarding the first part of the project in this paper, they have developed machine learning models including a deep learning framework which can simultaneously predict ADRs and identify the molecular substructures associated with those ADRs without defining the substructures a-priori.

They used CNN to represent the complex structures in fixed length vectors and simultaneously use deep learning to analyse the fingerprints and relate them to ADR. In particular, they design R hidden layers in the deep learning framework, each corresponding to a particular radius. Therefore, their framework can search for all possible substructures up to radius R by successive increment of the radius of the substructure by one in each layer of neural network. Afterward, the similar structures are summarized into a final feature representation called fingerprint.

At each step (radius), they use an additional attention mechanism step to map the contribution of each of the substructures into the final fingerprint. Finally, the fingerprints are assessed in terms of how well they can predict ADRs. They built a predictive model using L2-norm regularized logistic regression method for each ADR separately using those fingerprints as features.

3.4.4 Detecting Potential Adverse Drug Reactions Using a Deep Neural Network Model [17]

Chi-Shiang Wang et. al

The objective of this study was to identify a method to detect potential ADRs of drugs automatically using a deep neural network (DNN). They demonstrated the usefulness of the proposed representation by inferring two types of relations: a drug causes a side effect and a drug treats an indication. To predict these relations and assess their effectiveness, they applied 2 modelling approaches: multi-task modelling using neural networks and single-task modelling based on gradient boosting machines and logistic regression. They used MeSH and SIDER dataset.

The drug representation includes co-occurrence frequencies between each drug and all other MeSH terms. These numbers have to be normalized to account for the variability in the total number of drug occurrences. They implemented 3 normalization methods:37 maximum term-frequency normalization (Max-TF), in which each coordinate in the representation vector is divided by the maximum value of that vector; Log + Max-TF, in which the logarithm of the terms count is taken followed by Max-TF; and TF–inverse document frequency, which is commonly used in text mining and information retrieval for term normalization.

To examine the prediction performance of the drug representation on the 2 tasks, they used 3 types of models: Multilayer fully connected NN architecture, GBM, implemented using the LightGBM package.

CHAPTER 4

DATA

This chapter serves to describe the data under consideration. Understanding the way all the data work is vital in the process of creating a good solution to the problem at hand.

4.1 Overview

A drug is characterized by its structure, functional abilities and the side effects it causes. In order to generate a good embedding of a drug, we explore all of the above and keep what proves to be truly representative of a drug and what is not mathematically in concord with our problem.

4.2 SMILES: Structural Indication of a Drug

The simplified molecular-input line-entry system (SMILES) is a specification in the form of a line notation for describing the structure of chemical species using short ASCII strings. Encoded in each SMILES string is structural information that can be used to predict complex chemical properties.

SMILEs are representations of the structure of a drug. It is a “chemical language” that encodes structural information of a chemical into a compact text representation that is easy to consume. SMILEs use a strict grammar to maintain consistency. The alphabets (example, C, O, H) denote atoms, and in some cases also what type of atoms. Special characters (like ‘=’) denote the type of bonds (single, double, triple). Round brackets encapsulating numbers, and side chains denote rings. From this structural information, more complex properties can be predicted.

Example- Aspirin: CC(=O)OC1=CC=CC=C1C(=O)O

The SMILES data is extremely useful for mathematical representation of drugs as it contains both spatial and sequential data. We can leverage this to our advantage.

Our dataset holds 1004 smiles with their respective ATC classification (refer Section 4.5.1) This is the main dataset that we will be concerning with.

4.3 LINCS: Functional Indication of a Drug

The dataset under consideration to represent the functional aspect of a drug, i.e. how it interacts with cell lines is procured from the NIH Library of Integrated Network-Based Cellular Signatures Program. The L1000 assay (which is one of the many gene inclined datasets available as part of the NIH Library) measures mRNA transcript abundance of 978 “landmark” genes from human cells. Measurements of these 978 "landmark genes” are applied to an inference algorithm to infer the expression of 11,350 additional genes in the transcriptome. 1.6 million profile expressions are obtained by measuring mRNA abundance in perturbed cells.

Table 1: Statistics of the L100 Assay

Number of Perturbagens[1]	2107
Number of samples per perturbagen on an average	42
Dimension of each sample	11,350
Total number of samples	118,500

The dataset is divided into levels with different levels representing different aspects.

Level 1: There are various Luminex graphs, where x axis represents time and the y axis represents fluorescence, generated for different profiles.

[1] A perturbagen is a drug compound

Level 2: Reduced the 22k genes to 978 landmark genes for ease. Convert the above graphical data into tabular data [978 x 1.3 million]. 1.3 million columns are the various profiles. Control genes are those sequences that have been determined as unchanging on addition of any perturbagen in any of the genes. There are 80 such control genes which are represented as 80 columns in the dataset.

Level 3: This data has inferred information for 12K genes and 1.3 million profiles form level 2 data through a series of steps involving standardisation, normalisation and in-Ferring the remaining 11k genes through a multiplication with a weighted matrix.

Level 4: Indicates the up and down regulation in the gene expression on adding perturbagen. After getting the standard gene expression values from the control genes by average all the control probe values for all genes and average control probes for each gene, we perform a Z score like operation to give us the regulated value for the table.

Level 5: The replicates present in the level 4 data are consolidated which reduces the data dimension. For the purpose of this problem, Level 5 data is considered.

	Profile 1	Profile 2	1.6 million profiles
gene 1	+0.8	-1.43	
gene 2	-.2.5	+1.65	
gene 978			

Figure 2: Representation of the L1000 Assay

4.4 Combined Feature Set for ADR Detection

Collection of **transcriptomic data** and ADR-drug association Gene expression profiles for 978 landmark genes were extracted from **NIH funded LINCS L1000** dataset for level 5. The level 5 data provide transcriptomic data for 978 landmark genes for 20,413 small molecules in the form on consensus signature. A signature is a continuous values for the gene expression extracted from before and after the action of small molecule. Gene expression for a perturbagen is the expression for the strongest signature for perturbagen. The strongest signature for perturbagen is defined as the signature with highest "distil_ss" value for the signature common to given perturbagen. The feature "distil_ss" gives the magnitude of the difference of the differential expression for the signature from the average of DMSO treated control sample. The highest magnitude corresponds to most effective perturbagen and can be used irrespective of the cell line, concentration and duration of treatment. The preparation of gene expression data has been inspired by Wang et. al.

Two-dimensional **representation of the chemical structure** of the perturbagens were extracted in the form of **Simplified molecular-input line-entry system (SMILES)** from L1000 metadata for phase one. SMILES, which is present as metadata in L1000 dataset for small molecules, were found and converted to molecular fingerprint, namely **Molecular ACCess System structural keys (MACCS) - 166-bit structural key,** with each bit associated with SMARTS pattern. This conversion was made using the **Open Babel package.**

The genes from the LINCS dataset were represented using **Gene descriptors, specifically Gene Ontology**, using **Principal Angle Enrichment Analysis (PAEA)** method. Gene Ontology as gene descriptors were used in many publications for representations.

The feature set for the prediction consisted of the combined set of chemical fingerprints in the form of bit array of MACCS, molecular descriptors in the form of Gene Ontology and the effect of the drugs on genes in the form of RNA-expression from L1000 project.

From the feature set, we get transcriptomic data from L1000 for 978 landmark genes, Gene Ontology's expression value or gene descriptors for the perturbagen, for 4,439 genes and lastly 166-bit MACCS structural key. Giving us a total of 5,583 feature set for 20,413 small molecules.

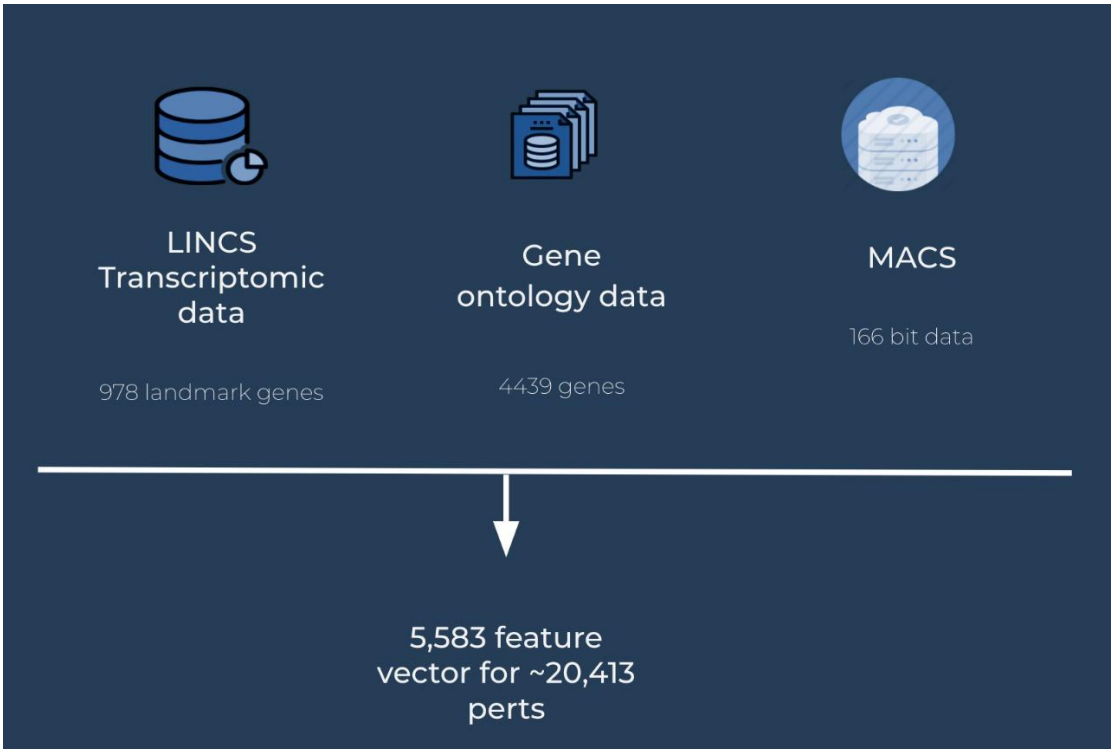

Figure 3: Combined feature set for ADR prediction

	PC1	PC2	5000 PC's
Drug Pair 1	-9.72	+6.8	
Drug Pair 2	+1.59	+20.9	
Drug Pair 34549			

Figure 4: Representation of the combine feature set2

4.5 Classification Systems

4.5.1 ATC: Anatomical Therapeutic Chemical Classification [18]

The Anatomical Therapeutic Chemical (ATC) classification system as a measuring unit are recommended by the WHO for drug utilization studies. In the Anatomical Therapeutic Chemical (ATC) classification system, the active substances are divided into different groups according to the organ or system on which they act and their therapeutic, pharmacological and chemical properties.

Drugs are classified in groups at five different levels. The drugs are divided into fourteen main groups (1st level), with pharmacological/therapeutic subgroups (2nd level). The 3rd and 4th levels are chemical/pharmacological/therapeutic subgroups and the 5th level is the chemical substance. The 2nd, 3rd and 4th levels are often used to identify pharmacological subgroups when that is considered more appropriate than therapeutic or chemical subgroups.

[2] PC = Principal components

The complete classification of metformin[3] illustrates the structure of the code:

A

Alimentary tract and metabolism

(1st level, anatomical main group)

A10

Drugs used in diabetes

(2nd level, therapeutic subgroup)

A10B

Blood glucose lowering drugs, excl. insulins

(3rd level, pharmacological subgroup)

A10BA

Biguanides

(4th level, chemical subgroup)

A10BA02

metformin

(5th level, chemical substance)

Thus, in the ATC system all plain metformin preparations are given the code A10BA02.

The main groups of the ATC classification system are listed below.

A Alimentary tract and metabolism

B Blood and blood forming organs

C Cardiovascular system

[3] https://www.whocc.no/filearchive/publications/1_2013guidelines.pdf

D Dermatologicals

G Genito urinary system and sex hormones

H Systemic hormonal preparations, excl. sex hormones and insulins

J Antiinfectives for systemic use

L Antineoplastic and immunomodulating agents

M Musculo-skeletal system

N Nervous system

P Antiparasitic products, insecticides and repellents

R Respiratory system

S Sensory organs

V Various

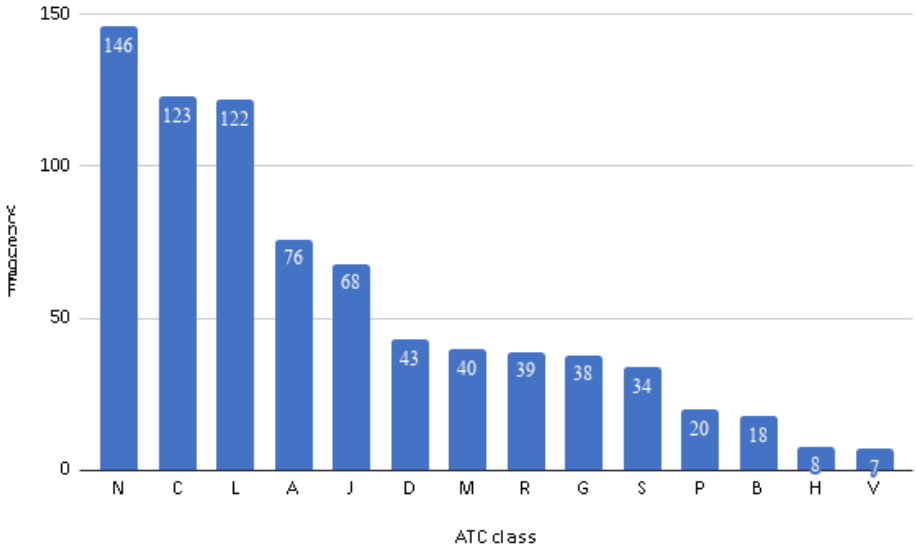

Figure 5: Distribution of ATC classes in LINCS perturbagens

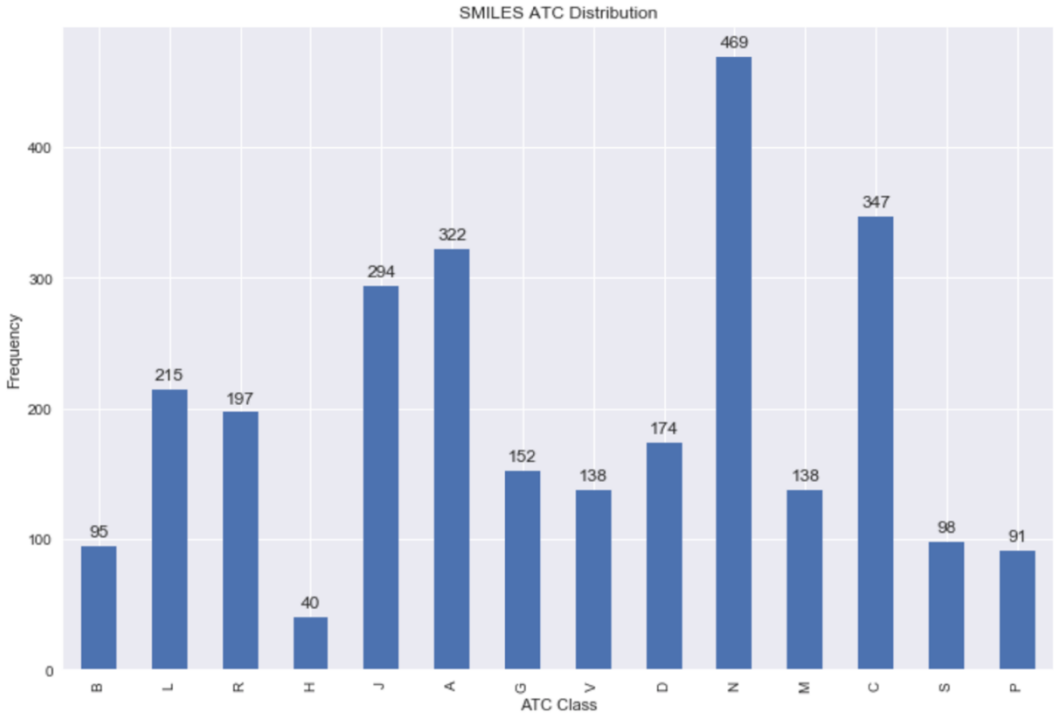

Figure 6: Distribution of ATC classes in SMILES perturbagens

4.5.2 SIDER: Side Effects Resource [19]

Unwanted side effects of drugs are a burden on patients and a severe impediment in the development of new drugs. At the same time, adverse drug reactions (ADRs) recorded during clinical trials are an important source of human phenotypic data. It is therefore essential to combine data on drugs, targets and side effects into a more complete picture of the therapeutic mechanism of actions of drugs and the ways in which they cause adverse reactions.

4.5.3 Adverse effects combined dataset

Drug Adverse Reaction (ADR) - Drug associations were extracted from two sources:

Side Effect Resource (SIDER): Stores information from the public documents about the adverse effect of the marketed drugs. At present SIDER contain data for 1430 drugs, 5868 side effects and 1396 drug-ADR association.

PharmGKB: From PharmGKB we used the database of drug-drug interaction side effects (Twosides). This database contains 868,221 putative drug-interaction effects from 59,220 pairs of drugs and 1,301 adverse events

Figure 7: Combined label set for ADR prediction

Combining the above datasets with the feature set (mentioned in Section 4.4 Combined Feature Set for ADR Detection):

From the 59,220 pairs of drugs present in PharmGKB, only 34,549 drug pairs had transcriptomic information available in LINCS (60% overlap). From the 59,220 pairs of drugs present in

PharmGKB, only 34,549 drug pairs had transcriptomic information available in LINCS (60% overlap).

The 5,583-feature set for each perturbagen in a pair was concatenated to get 11,166 combined feature set. PCA was then performed on this and the total number of features were brought down to 5000 for the pair.

The 1,301 adverse events were also reduced down to 243 most commonly occurring events.

Result: Dataset with 34,549 drug pairs and 5000 features, along with 243 drug labels

CHAPTER 5

METHODOLOGY

This chapter comprehensively details the methodology used to solve the problem at hand.

5.1 Overview

The overall outline of the methodology involves subjecting gene expression data or SMILES data to an embedding generating network. Once the embeddings are retrieved, they are passed through another classification model to validate the capability of the embeddings against a classification system.

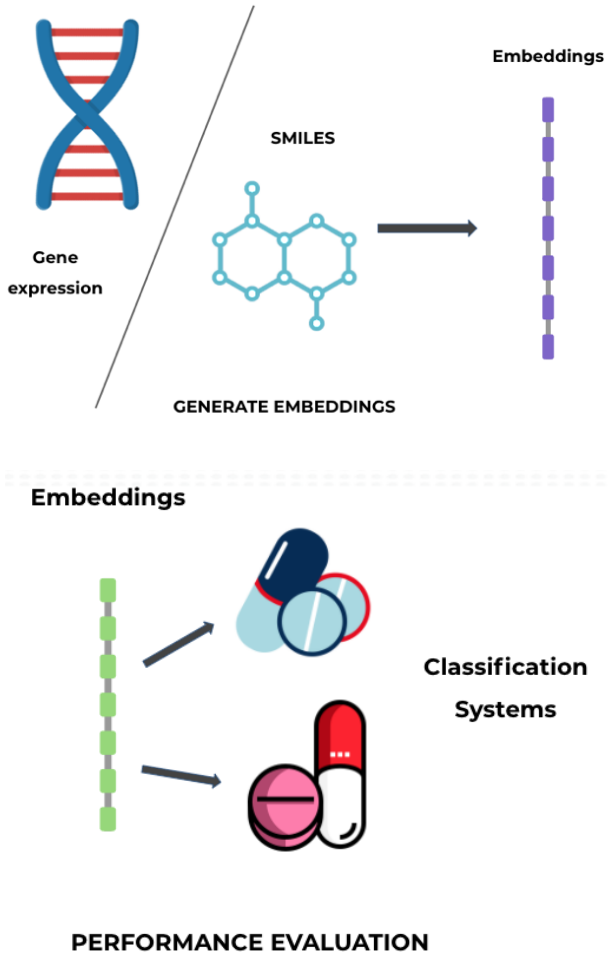

Figure 8: Overall view of methodology

The general workflow of ADR prediction using PharmGKB and LINCS dataset, is shown in Figure 9 and Figure 10, which consists of the following steps: represent drugs as multi-dimensional vectors according to their heterogeneous features, which makes it possible to employ advanced machine learning methods for predictions, building predictive models and interpreting those features for characterizing substructures associated with ADRs.

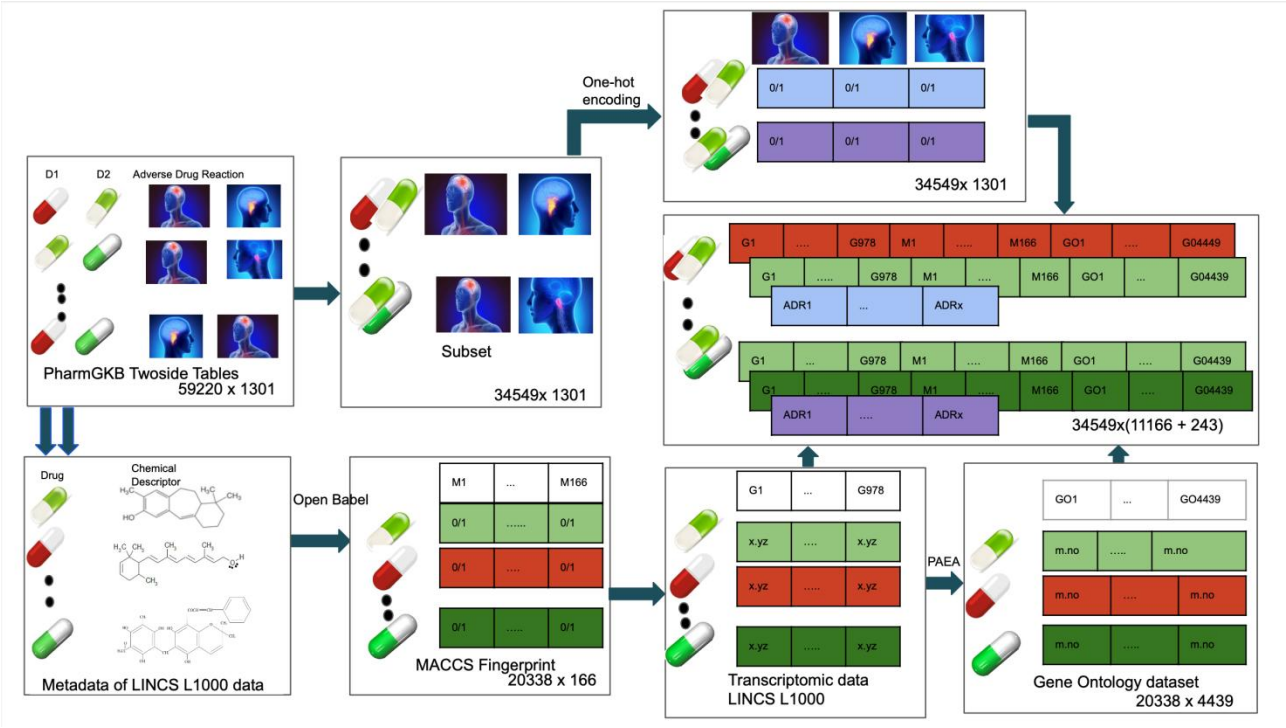

Figure 9: Data Preparation

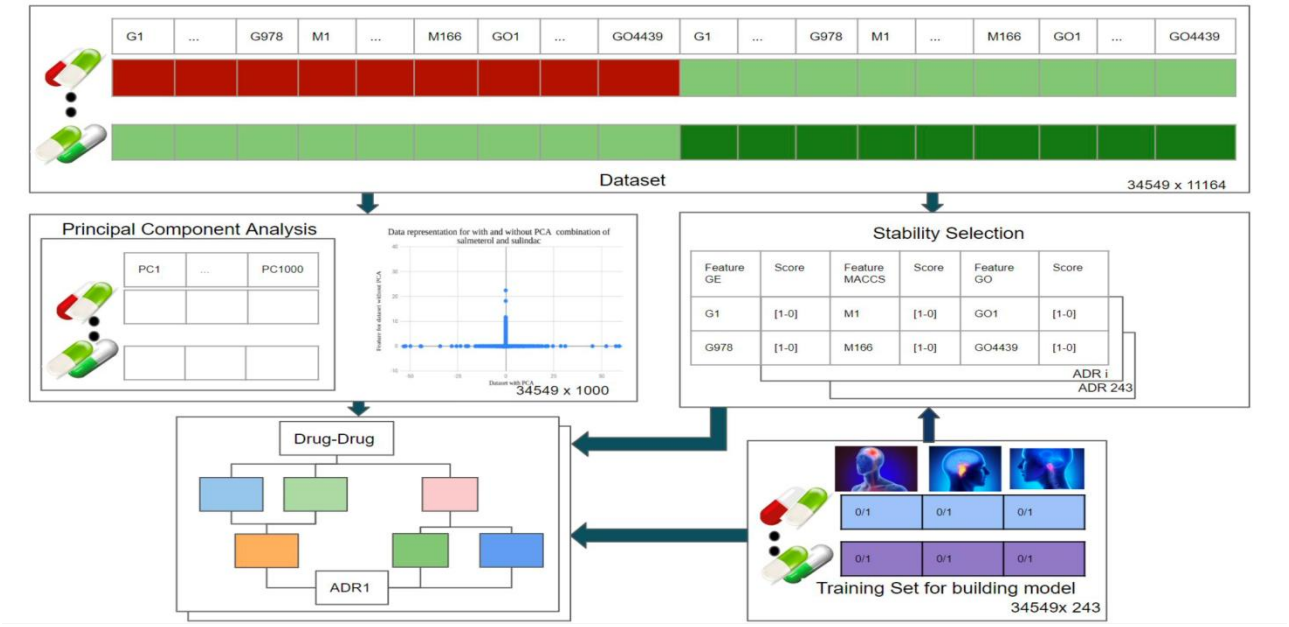

Figure 10: Training predictive models

5.2 Initial Analysis

5.2.1 LINCS

1. Delineating by Cell-line

   We figured out the **top 7 cell** lines that represent 90% of the dataset and **trained individual networks** for each of the cell lines. This did not yield ground breaking results and proved to be an impractical solution to the problem. By delineating by cell lines we could verify that there was no observable similarity of gene expression data within a cell line.

   We trained the following networks:

   - Support Vector Machines (SVM)
   - Artificial neural networks (ANN)
   - K- Nearest Neighbour (KNN)
   - Random Forrest (RF)

2. Delineating by ATC Class

   We attempted to **find correlations within cell lines** of an ATC class but like the previous analysis there was no observable similarity or clustering of gene expressions within a particular class.

3. Transforming Gene Expressions

   We replaced the floating point values in the LINCS dataset (which indicate mRNA responses) by **integers to denote upregulation and downregulation** to find out similarity. A positive floating point value would be represented as a 1 and a negative floating point value would be represented as a -1. The thought process behind these experiments was to see if there was any upward or downward trend in the gene expression values when brought down to their very basic values. This experiment was inconclusive.

4. Visualizing Gene Expressions

   We plotted the ATC class clustering capability of **LINCS 978** dimensional vector using t-SNE algorithm. The idea of this algorithm is to take a set of points in a high-dimensional space and find a faithful representation of those points in a lower-dimensional space, typically the 2D plane. The algorithm is non-linear and adapts to the underlying data, performing different transformations on different regions. The x-axis and y-axis of the plot are the new 2D dimensions generated after the algorithm runs.

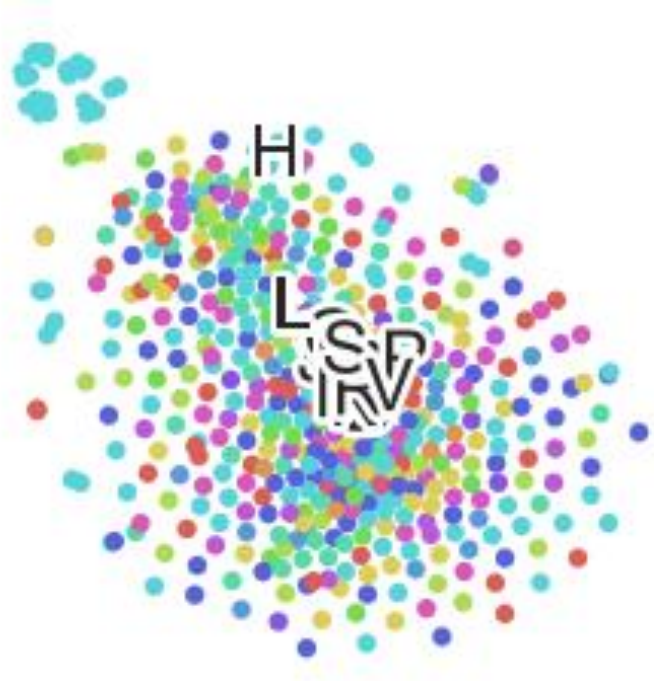

Figure 11: ATC class (All 14) clustering capability of 978-dimension gene expressions

From Figure 11, it is confirmed that there is no apparent clustering capability or correlation between gene expression data.

5.2.2 ADR Prediction

5.2.2.1 Drug Pair Representation

Each drug is represented as a vector of size 5583 (as described in section 4.4 Combined Feature Set for ADR Detection). A drug is presented by concatenating the feature vectors of the two drugs. Processing 11166 dimensional data which incurs a high computational cost to train a classifier therefore, PCA is performed to reduce the dimensionality and make it easier to fit machine learning models on the feature. Finally, the feature set represents the drug pairs as a single vector of 5000 principle components.

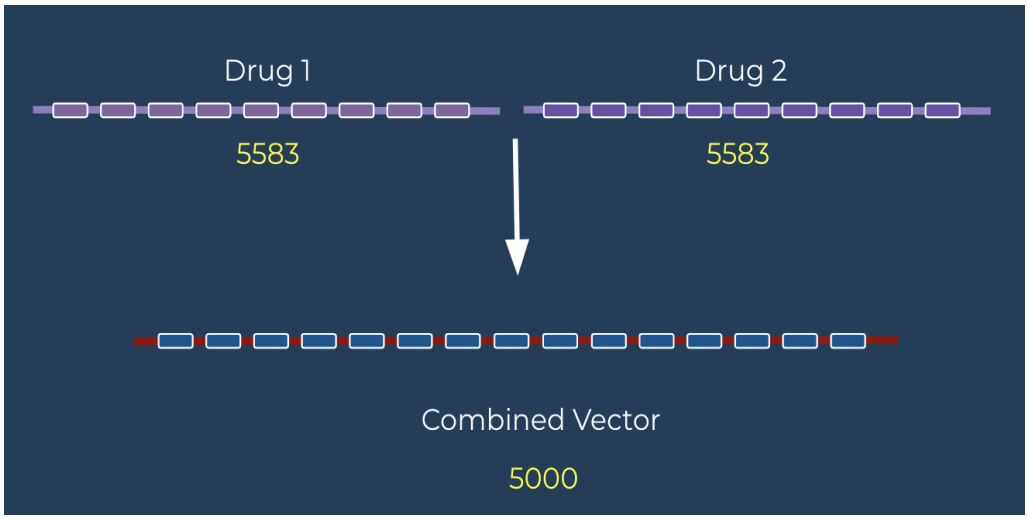

Figure 12: Drug-Pair Representation

5.2.2.2 ADR Prediction using SVM

Support vector machine is a representation of the training data as points in space separated into categories by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. It is one of the most renowned model when it comes to classification but solving the quadratic problem and choosing the support vectors is generally hard. The space complexity of the same is n^3 where n is the size of the training set, that is 34549^3. This was not supported by the machine being used and therefore the experiment was unsuccessful.

5.2.2.3 Other models for prediction

The ADRs prediction problem was considered as a binary classification problem for each of the 243 drugs where each drug-drug pair was either associated with a 1 (causes ADR) or a 0 (doesn't cause ADR). We employed different state-of-the-art machine learning approaches, Random Forest, Naïve Bayes, K Nearest Neighbours, Stochastic Gradient Descent and Logistic Regression to predict indications.

For each of the 243 ADRs, a classifier model was generated using training data that included PCA reduced features (5000 dimensions) for 34549 drug-drug pairs.

5.2.2.4 Artificial Neural Networks

Instead of treating the problem as multiple single class classification problem, this approach helps us to attempt the same problem as a multilabel classification problems to make the overall process faster by taking the advantage of correlations between he labels and sharing the feature space for all the labels.

5.3 Generating Embeddings

5.3.1 SMILES

As defined in section 6.2, SMILEs are structural representations of molecules denoted by a linear sequence of ASCII characters representing atoms, bonds between atoms and orientations of substructures (i.e. chirality). While SMILEs are descriptive representations of molecules containing important structural information, the strings cannot be used in their original states for any model-based classifications as these would require numeric representation. There is a requirement to find an effective mechanism to encode SMILE strings as numeric arrays while retaining maximum structural information. The chosen encoder must preserve information on the sequential details of the molecule and maintain sub-structural integrity. In the subsequent sections, 3 different approaches for SMILE encoding are proposed.

5.3.1.1 Character Encoding

This approach treats a SMILE string as a sentence containing a sequence of words, where each word is a character in the string. Each of the unique characters present in SMILE strings are mapped to unique one hot encoding allowing every SMILE to be encoded into an array of one hot vectors.

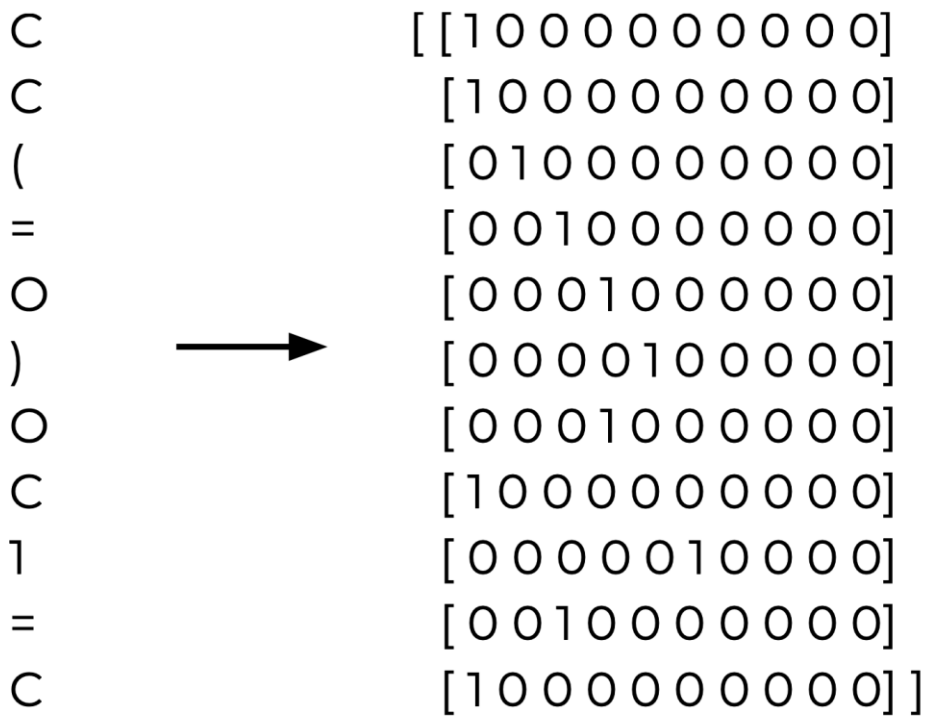

Figure 13: Sample SMILES one-hot encoded vector

The main advantages of this method are its simplicity and its ability to retain all sequential information. This sequential nature of the data can be leveraged by models that extract patterns within progressions such as Convolutional Neural Network and Recurrent Neural Networks. The following section demonstrates the classification models built on these embeddings.

Classification Model

Evaluation of the embeddings generated are primarily decided by their ability to classify drugs into the 14 ATC classes. The classification model used here is a hybrid of a convolutional network and a recurrent network.

A convolutional neural network or CNN is a network used to extract features of consequence, present in sequential data points (single or multi-dimensional). A convolutional layer in the network converts the given input into a sequence of features where each feature is a combination of adjacent data points. Multiple such layers make a CNN. Pooling layers can be added to extract important

features. In the character encoding generated for a smile, performing convolution will create features that are a sequence of atoms and bonds and pooling layers will extract those features that are important to the molecule.

A recurrent neural network is a feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. In the current application, a molecule's properties are determined not just by individual atoms or sub structures, but by the combination of its preceding substructure. Employing an RNN will allow the network to extract sequential patterns present in the data.

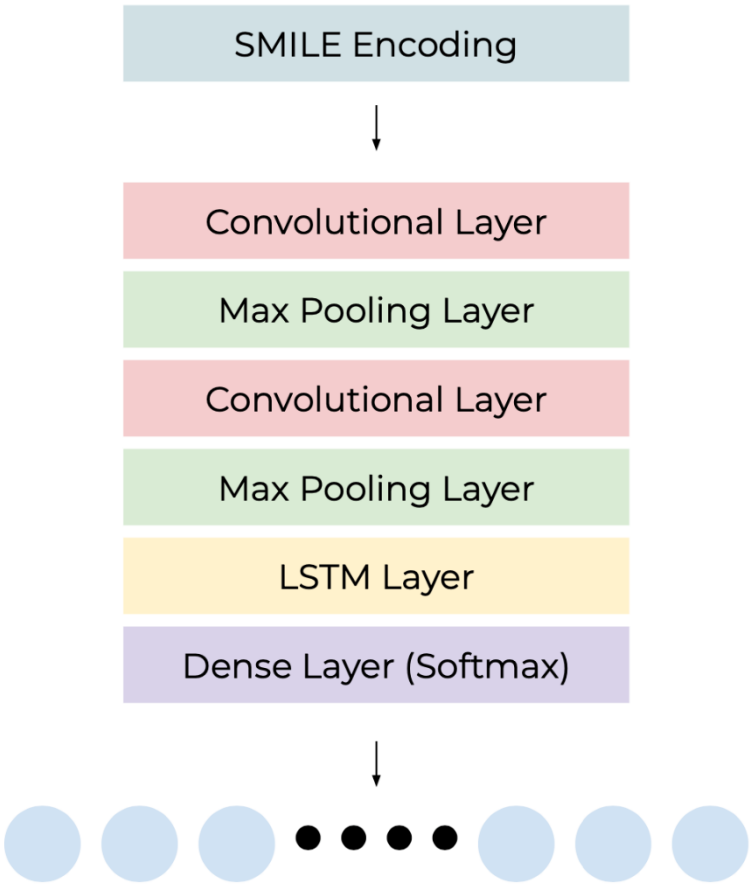

Figure 14: Architecture of the classification model

The labelled dataset consisting of character encodings of SMILEs mapped to the ATC level 1 class it belongs to is divided into a train set and a test set with a ratio of 7:3. The network is trained for 20 epochs until a saturation of training accuracy is seen.

Code:

All possible characters that can be present in a SMILE and their respective frequency of occurrence are calculated using the below snippet of code. A dictionary of the one-hot array of every character in the vocabulary is created.

```
vocabulary = set()
 allSmiles = ""
 for smile in df['smiles']:
    allSmiles+=smile
    for char in smile:
        vocabulary.add(char)
 freq = Counter(allSmiles).most_common(25)
 vocabulary = [a for a,b in freq]

oe = np.asarray(pd.get_dummies(np.asarray(vocabulary)))
oe_dict = {}

for a in range(len(vocabulary)):
    oe_dict[vocabulary[a]] = oe[a]
```

The function below encodes a smile into its character encoding by using the above vocabulary dictionary. For each character in the SMILE, the respective one-hot representation is calculated and summed up. This gives the final character encoding.

```
def encodeSmile(smile, oe):
    word = []
    for char in smile:
        try:
```

```
            temp = list(oe[char])
        except:
            temp = [0 for a in range(len(vocabulary))]
        word = word + temp
    return np.asarray(word, dtype=int)
```

The implementation of the network used to classify the smiles into their respective ATC classes is detailed below. It is a CNN-RNN architecture that makes use of the spatial and sequential information present in SMILEs.

```
class Network:
    def __init__(self, num_classes):
        self.model = Sequential()
        self.model.add(Reshape((250, 25), input_shape=(6250, )))
        self.model.add(Conv1D(192, 5, activation='relu', input_shape=(250, 25)))
        self.model.add(MaxPool1D(5))
        self.model.add(BatchNormalization())
        self.model.add(Conv1D(192, 5, activation='relu'))
        self.model.add(Conv1D(140, 3, activation='relu'))
        self.model.add(LSTM(70, dropout=0.7, recurrent_dropout=0.2,        activation='selu', return_sequences=True,))
        self.model.add(LSTM(70, dropout=0.4, recurrent_dropout=0.2, activation='relu'))
        self.model.add(Flatten())
        self.model.add(Dense(100, activation='relu'))
        self.model.add(Dropout(rate=0.4))
        self.model.add(Dense(num_classes, activation='softmax'))
        self.model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
        print(self.model.summary())

    def train(self, X, y, X_test, y_test, epochs=20):
        self.model.fit(X, y, epochs=epochs, validation_data=(X_test, y_test))

    def test(self, X, y):
        y_pred = self.model.predict(X).argmax(axis=1)
        y_true = y.argmax(axis=1)
        accuracy(y_true, y_pred)
```

5.3.1.2 Substructure Encoding

The previous section which details the implementation of character encoding, demonstrates that treating a SMILE as a sequential string with the intention of allowing feature mapping to naturally occur through the training process does not perform well. A different approach must be taken where the encodings generated by the encoder already contain chemically important features that define the molecule. Using a technique with a scientific foundation would allow for chemically and structurally accurate encodings. The following section describes one scientifically backed approach, the *Morgan Algorithm*.

The Morgan Algorithm

This is a method used to convert a SMILE into an array of substructures. The algorithm defines a substructure as a collection of atoms in a molecule centered around one atom, such that all other atoms in the substructure are within a distance ***r*** from the central atom. An atom is considered a distance ***r*** away from another atom if there are ***r*** bonds separating the two atoms.

The Morgan algorithm iteratively progresses sequentially through the SMILE creating an array of all the substructures surrounding each of the atoms. These substructures are ordered such that if central atom *A* occurs before central atom *B* in the SMILE, the substructures of *A* also occur before substructures of *B.* All the substructures centered around an atom *A* are ordered in the increasing order of their radii (Radius of a substructure is defined as the maximum distance between the central atom and any other atom in the substructure).

A substructure can no longer be represented by its chemical name. Every substructure is assigned a 32 bit identifier called the *Morgan Identifier* which are used in all molecular representations

The Morgan Algorithm with a radius of 1 was applied to each of the SMILEs in the dataset converting the string notations to numeric arrays

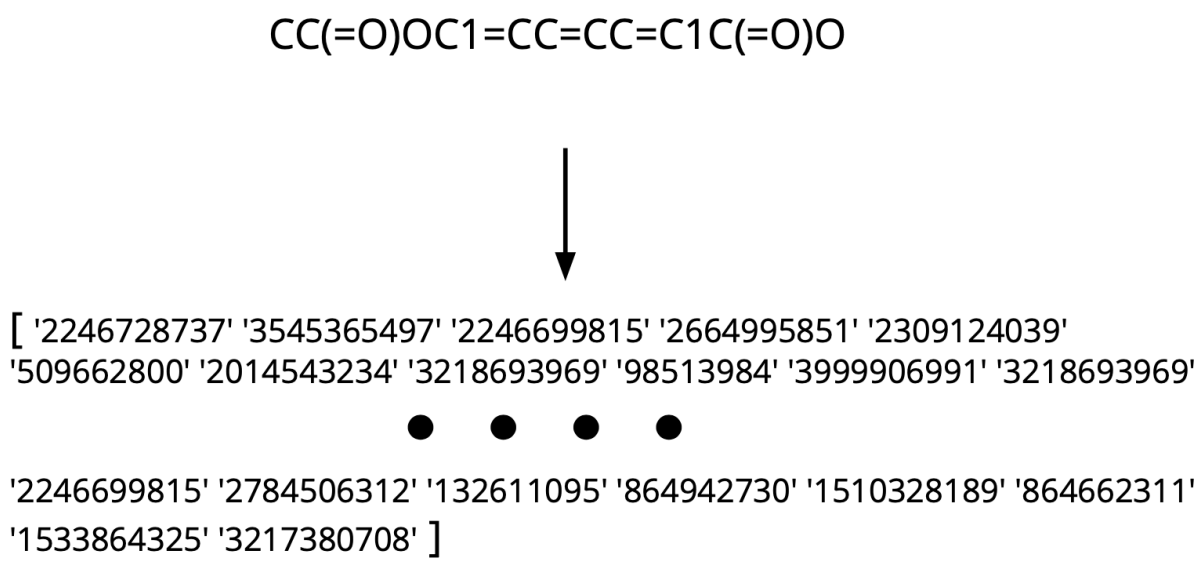

Figure 15: Substructure representation of a SMILE

Now each drug has a feature representation where each element in the vector representation is a 32 bit identifier signifying a substructure. The 32 bit identifier is in itself meaningless. Some means of encoding these identifiers is necessary. The following sections detail the skip gram approaches to encoding the molecular identifiers.

Skip Gram

"You shall know a word by the company it keeps"

John Rupert Firth

The main intuition behind the skip gram is that a word can be characterised aptly by the words in its vicinity. In order to model this hypothesis, a dataset of sentences is transformed into pairs of contexts and words. Every word in the dataset is coupled with an array of words in its immediate neighbourhood, the context. The skip gram model is trained such that, given a word, it predicts the context of that word. The output of the middle layer in this model is then used as the embedding to represent that word.

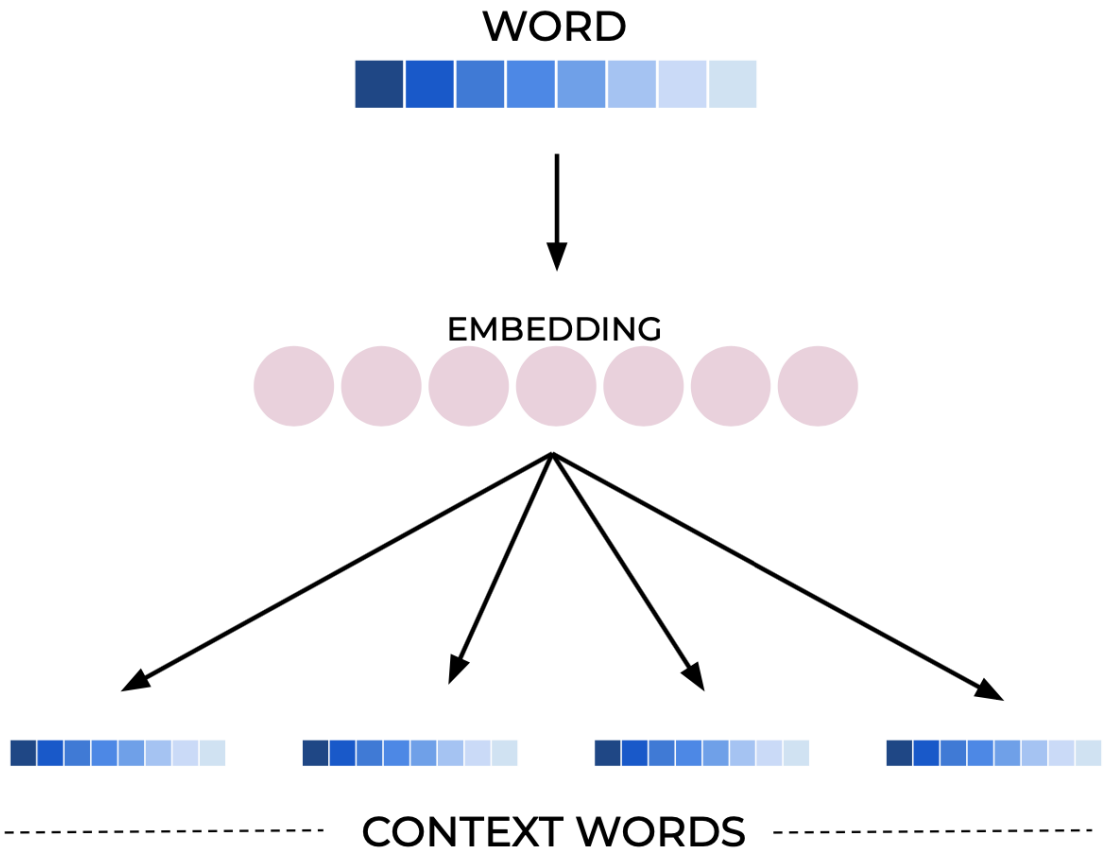

Figure 16: Architecture of the skipram model

The dataset consisting of molecular identifiers was trained using the skip gram model described above. Training was done for an embedding length of 50, 100, 150, 200, 250 300.

Table 2: Details of the architecture of current model

Dimension	50, 100, 150, 200, 250, 300
Window Size	5
Minimum word count	5
Negative Sampling	15

The network was trained for 20 epoch and word embeddings were generated. Arrays of word identifiers could be encoded into a series of word embeddings and can also be averaged into one embedding for the molecule.

Classification Evaluation

Every molecule is converted into an embedding by averaging the embeddings of every identifier in it. An initial evaluation was conducted to determine the quality of these embeddings. t-SNE was employed to map the embeddings onto a 2 dimensional space. The lower dimensional embeddings were plotted on a graph coloured by their ATC level 1 class.

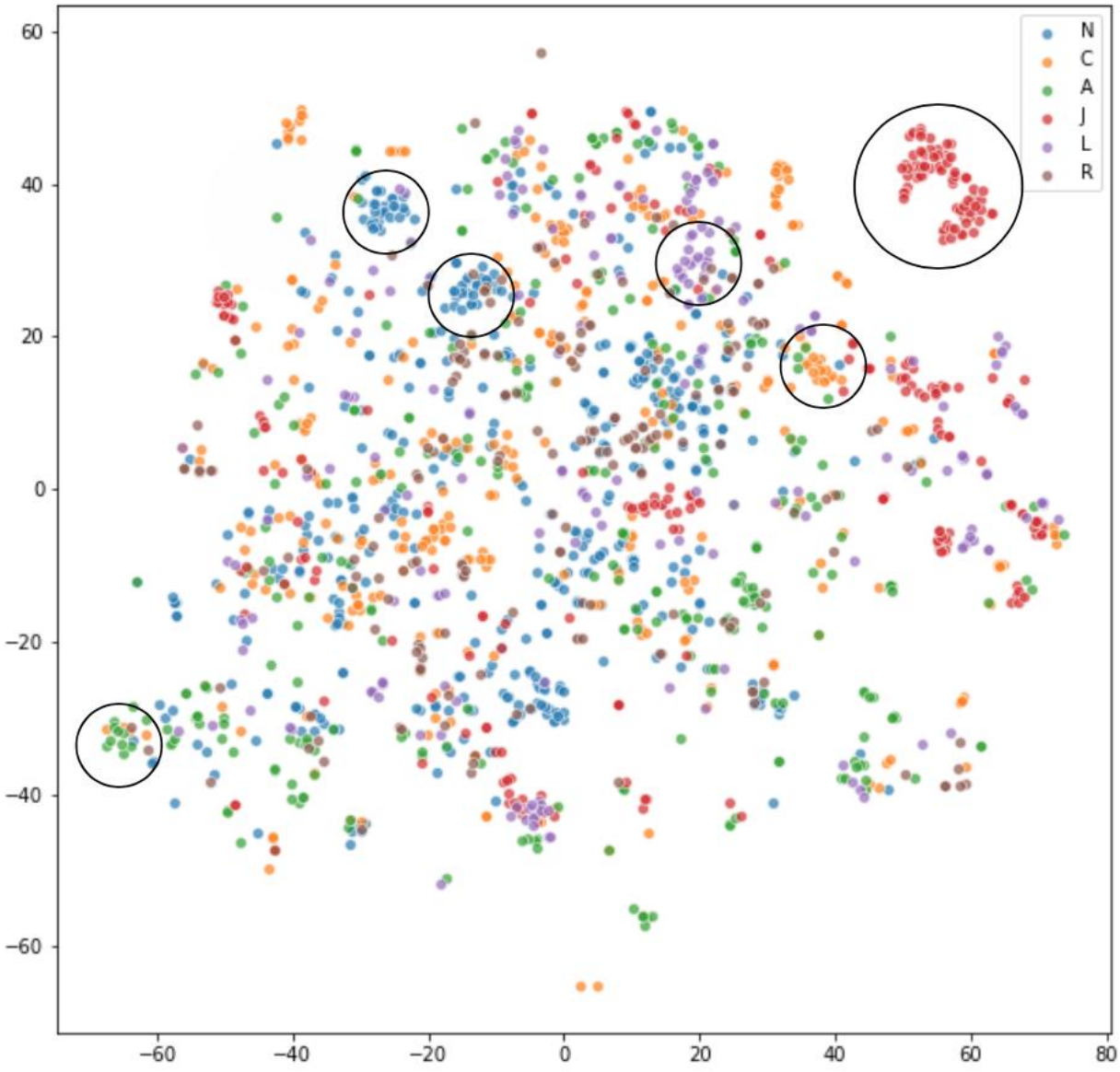

Figure 17: t-SNE plot of embeddings generated by the skipgram model against all 14 ATC classes with highlighted clustering capability

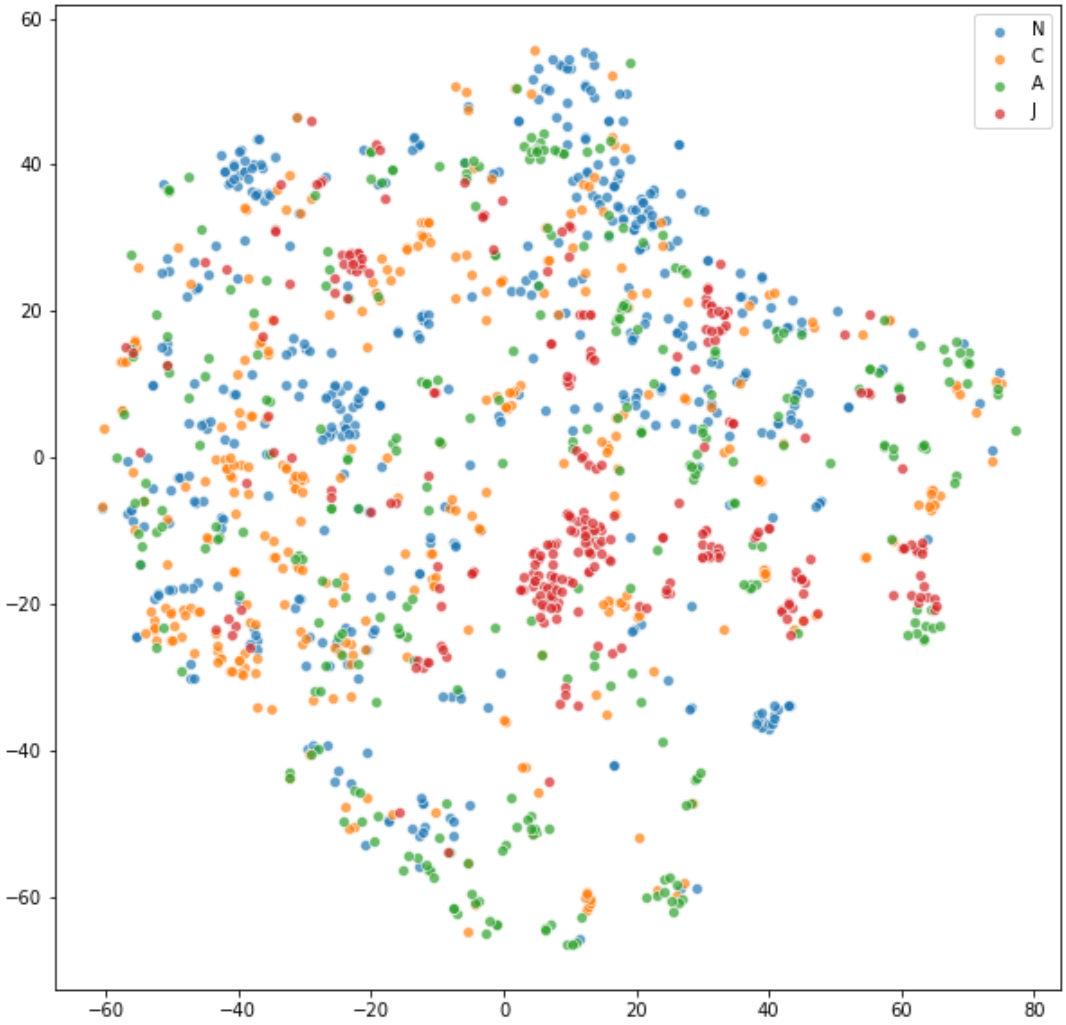

Figure 18: t-SNE plots of the embeddings generated by the Skipgram model against top-4 ATC classes

In the above t-SNE plots, it can be observed that there are clusters of classes located in the same area (Figure 17). This indicates that the embeddings show inherent similarity within a class. Based on the plot, it can be observed that a neighbor-based classification like KNN or a nonlinear classification model like tree-based models could prove effective.

In order to test the efficacy of these embeddings, several classification models were employed to determine the embeddings' ability to predict the level 1 ATC class of the molecule.

1. *K Nearest Neighbours*

This is a model where the classification of a data point is determined by the most frequent classification of its neighbours. The embeddings were split into train and test sets at a ratio 7:3. The model was fit and tested.

Table 3: Tabulated results of KNN evalutaion

Classes	Accuracy	Precision	Recall	F1 Score
14 (unbalanced data) [A B C D G H J L M N P R S V]	44.88%	0.49	0.45	0.46
4 (balanced data) [A C J N]	63.01%	0.66	0.63	0.64
3 (balanced data) [C J N]	72.31%	0.73	0.72	0.73

2. Random Forest Classifier

The random forest is a model made up of many decision trees. Rather than just simply averaging the prediction of trees, this model uses two key concepts: random sampling of training data points when building trees and random subsets of features considered when splitting nodes. The embeddings were split into train and test sets at a ratio 7:3. The model was fit and tested.

Table 4: : Tabulated results of Random Forrest evaluation

Classes	Accuracy	Precision	Recall	F1 Score
14 (unbalanced data) [A B C D G H J L M N P R S V]	48.54%	0.56	0.49	0.51
4 (balanced data) [A C J N]	62.43%	0.64	0.62	0.63
3 (balanced data) [C J N]	76.15%	0.77	0.76	0.76

3. Extra Trees Classifier

This is a model where the classification of a data point is determined by the most frequent classification of its neighbours. The embeddings were split into train and test sets at a ratio 7:3. The model was fit and tested.

Table 5: : Tabulated results of Extra Trees evalutaion

Classes	Accuracy	Precision	Recall	F1 Score
14 (unbalanced data) [A B C D G H J L M N P R S V]	48.54%	0.55	0.49	0.5

4 (balanced data) [A C J N]	66.47%	0.68	0.66	0.67
3 (balanced data) [C J N]	76.54%	0.77	0.77	0.76

Code:

The function below implements the logic used to represent a SMILE as its substructure encoding.

```
def mol2alt_sentence(mol, radius):
    radii = list(range(int(radius) + 1))
    info = {}
    _ = AllChem.GetMorganFingerprint(mol, radius, bitInfo=info)

    mol_atoms = [a.GetIdx() for a in mol.GetAtoms()]
    dict_atoms = {x: {r: None for r in radii} for x in mol_atoms}

    for element in info:
        for atom_idx, radius_at in info[element]:
            dict_atoms[atom_idx][radius_at] = element

    identifiers_alt = []
    for atom in dict_atoms:  # iterate over atoms
        for r in radii:  # iterate over radii
            identifiers_alt.append(dict_atoms[atom][r])

    alternating_sentence = map(str, [x for x in identifiers_alt if x])

    return list(alternating_sentence)
```

5.3.2 LINCS

Given the non-descriptive nature of LINCS (as presented in **Error! Reference source not found.**), gene expressions may not be the best mathematical candidate for generating embeddings. However, we can try to use deep learning techniques to gather inherent similarities which are not observed through basic analysis. For this we choose to employ a 'Densely Connected Triplet Network'. The network is an unsupervised learning technique that aims to predict if a gene expression belongs to a particular perturbagen or not. This way the embeddings of gene expressions that belong to the same perturbagen will be similar and those that belong to different perturbagens will be dissimilar. We will later verify if our embeddings work by classifying ATC classes.

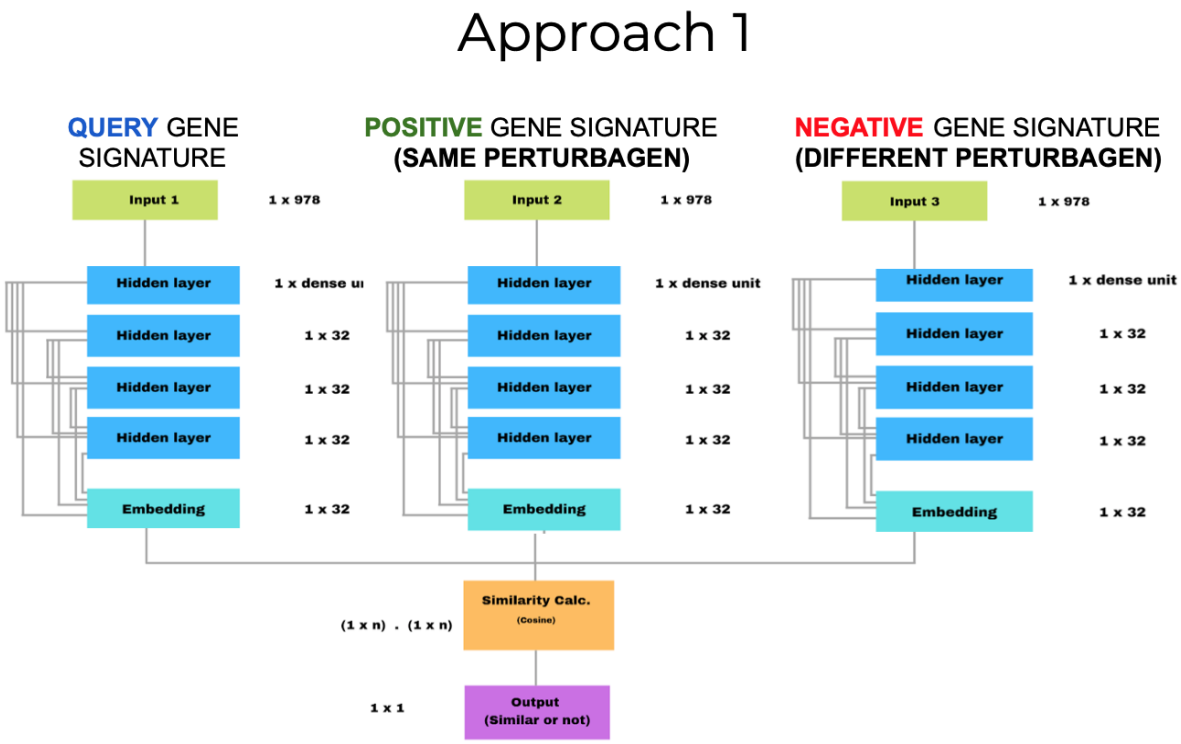

Figure 19: Architecture of the densely connected triplet network

5.3.2.1 Densely connected network

Densely connected means that the output of each layer is concatenated with the output of the previous layer and sent as input to the next layer. So the input to each layer is the concatenation of the input to the previous layer and the output of the previous layer. This propagation of each layer's inputs introduces original inputs into each layer allowing for fewer neurons in a layer. The network uses the activation function SELU[4]. The self-normalising nature of this activation function makes for faster training of the network. The last layer of the network is the size of the embedding (e). The normalized output from this layer is considered the embedding representing the perturbagen.

5.3.2.2 Triplet Network

The Triplet Network approach involves three identical fully connected neural networks that use the same weights while working on three different input vectors concurrently.

The three inputs to the network are referred to as 'Anchor', 'Positive', and 'Negative'.

- Anchor: The query gene expression.
- Positive: The gene expression which belongs to the same class as the Anchor.
- Negative: The gene expression which does not belong to the same class as the Anchor.

The three inputs go through identical networks which result in three embeddings corresponding to each of the inputs.

The architecture of the individual network is as follows:

- 3-5 layers
- 512-1024 neurons per layer
- RELU activation

[4] SELU: Self normalizing Exponential Linear Unit

- Dropout (0.5-0.9)

The loss function being employed by the network is a two-stage loss. Similarity between Anchor and Positive is calculated by minimising cosine distance between the two. Dissimilarity between Anchor and Negative is calculated by maximising cosine distance between the two.

5.3.2.3 Code

The triplet network is implemented using Tensorflow and python constructs. The class 'Triplet' contains the implementation of the network. The class' initialising function takes in a variety of user inputs like the type of loss (cosine / euclidean), type of network in each branch (simple neural network / Densely connected neural network), number of layers, number of neurons in each layer.

```
class Triplet:

  # Create model
  def __init__(self, loss='cos', network='densenet', num_layers=10, neuron=100, emb_len=32, dropout=0):d
    tf.reset_default_graph()

    self.x1 = tf.placeholder(tf.float32, [None, 978], "input1")
    self.x2 = tf.placeholder(tf.float32, [None, 978], "input2")
    self.x3 = tf.placeholder(tf.float32, [None, 978], "input3")

    if network == 'tripletnet':
      self.network = self.normalnet
    if network == 'densenet':
      self.network = self.dense_network

    with tf.variable_scope("triplet") as scope:
      self.o1 = self.network(self.x1, num_layers, neuron, emb_len, dropout)
      scope.reuse_variables()
      self.o2 = self.network(self.x2, num_layers, neuron, emb_len, dropout)
      scope.reuse_variables()
      self.o3 = self.network(self.x3, num_layers, neuron, emb_len, dropout)

    if loss == 'cos':
```

```
        self.loss = self.loss_with_cosine()
    if loss == 'euc':
        self.loss = self.loss_with_euclid()
```

The methods below implement each branch in the triplet network. The first function implements a simple neural network (normalnet). The next function implements the densely connected neural network. Each layer's output is concatenated with the layer's input and passed on to the next layer's input.

```
def normalnet(self, x, num_layers, neuron, emb_len, dropout):

    input = x
    for l in range(num_layers):
        name = "fc" + str(l)
        fc1 = self.fc_layer(input, neuron, name)
        ac1 = tf.nn.relu(fc1)
        if (dropout):
            d1 = tf.nn.dropout(ac1, dropout)
            input = d1
        else:
            input = ac1

    fc_last = self.fc_layer(input, emb_len, "fc_embedding")
    fc_last = tf.nn.l2_normalize(fc_last, axis=1, name = "fc_normal")
    return fc_last

def dense_network(self, x, num_layers, neuron, emb_len, dropout):
    fc1 = self.fc_layer(x, 400, "fc1")
    ac1 = tf.nn.relu(fc1)
    op_concat1 = tf.concat([x, ac1], axis=1)

    fc2 = self.fc_layer(op_concat1, 400, "fc2")
    ac2 = tf.nn.relu(fc2)
    op_concat2 = tf.concat([op_concat1, ac2], axis=1)
```

```
    fc3 = self.fc_layer(op_concat2, 400, "fc3")
    ac3 = tf.nn.relu(fc3)
    op_concat3 = tf.concat([op_concat2, ac3], axis=1)

    fc4 = self.fc_layer(op_concat3, 32, "fc4")
    fc4 = tf.nn.l2_normalize(fc4, axis=1,name = "fc_normal")
    return fc4
```

The ‘fc_layer’ method implements a fully connected layer. It is equivalent to the Keras API’s ‘Dense’ layer.

```
def fc_layer(self, bottom, n_weight, name):
    assert len(bottom.get_shape()) == 2
    n_prev_weight = bottom.get_shape()[1]
    initer = tf.truncated_normal_initializer(stddev=0.01)
    W = tf.get_variable(name + 'W', dtype=tf.float32, shape=[n_prev_weight, n_weight], initializer=initer)
    b = tf.get_variable(name + 'b', dtype=tf.float32,
                        initializer=tf.constant(0.01, shape=[n_weight], dtype=tf.float32))
    fc = tf.nn.bias_add(tf.matmul(bottom, W), b)
    return fc
```

The triplet loss is calculated by calculating the loss between the first two branches and the first and third branch separately.

```
def loss_with_cosine(self):
    cos2 = tf.multiply(self.o1, self.o2)
    cos2 = 1 - tf.pow(tf.reduce_sum(eucd2, axis=1), 2)
    cos2_mean = tf.reduce_mean(eucd2, axis=0)

    cos3 = tf.multiply(self.o1, self.o3)
    cos3 = 1 - tf.pow(tf.reduce_sum(eucd3, axis=1), 2)
    cos3_mean = tf.reduce_mean(eucd3, axis=0)
```

```
    return cos2_mean, cos3_mean, cos2, cos3
```

Once the gene expression has been passed through the network we get a 16 dimensional embedding.

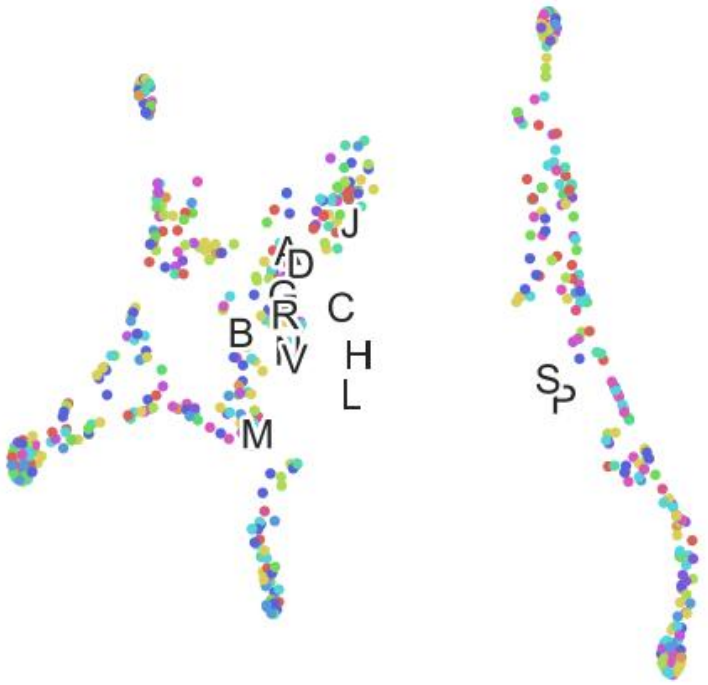

Figure 20: Clustering capability of embeddings against 14 ATC classe

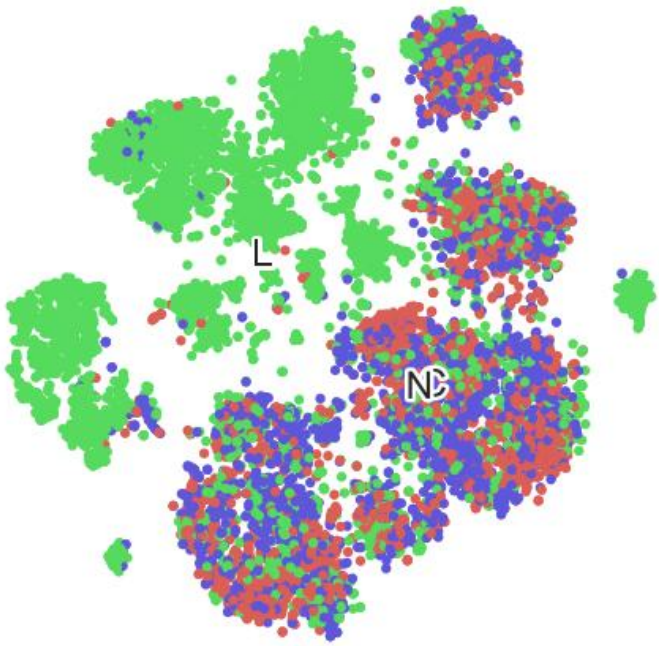

Figure 21: Clustering capability of embeddings against top-3 ATC classes (L, N, C)

5.3.2.4 Evaluating embeddings

The embeddings are evaluated against ATC classes to verify their usefulness. The above embeddings are passed through a simple neural network with the following architecture:

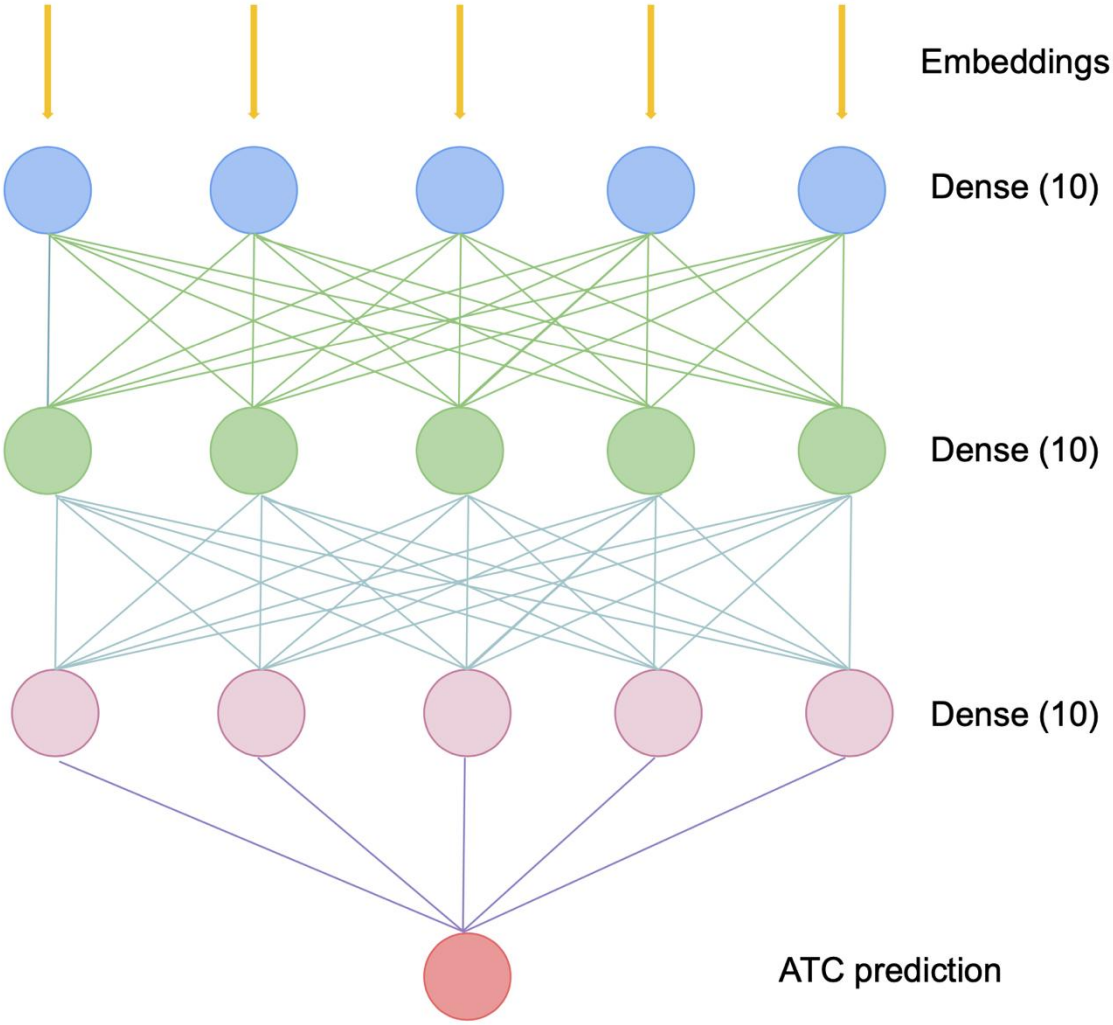

Figure 22: Architecture of simple classification network

Due to the extremely skewed representation of the ATC classification dataset (presented in Figure 5), we take in account the top 3 ATC classes (L, M, N) for the sake of representative labels.

5.4 ADR Prediction

This section describes in detail the various techniques used for the prediction of adverse effects using the feature dataset obtained from transcriptomic data. We explore the use of multiple pharmacologic databases as a new data source to identify ADRs of combined medication.

5.4.1 Training

We build a binary classifier for each ADR using the feature dataset. The training process was done in batches for 10, 20, 30, 50, 100, 200, 300, 500, 800, 1000, 1300, 1500, 2000, 2500, 3000, 3500, 4000, 4500 and 5000 PCA's and the results were compared.

5.4.1.1 Decision Trees

Decision Tree algorithm is used for both predictions as well as classification in machine learning. Using the decision tree with a given set of inputs, one can map the various outcomes that are a result of the consequences or decisions. In our model we used the following parameters:

- max_depth: 10
- random_state: 101
- min_samples_leaf: 15
- criterion: gini index

The results of the same are represented in Figure 23. In this figure we can see different macro scores for each PCA level. The model seems to perform optimally when 1000 PCAs are used for training. It is evident from the graph that more than a 1000 PCAs do not result in significantly better results.

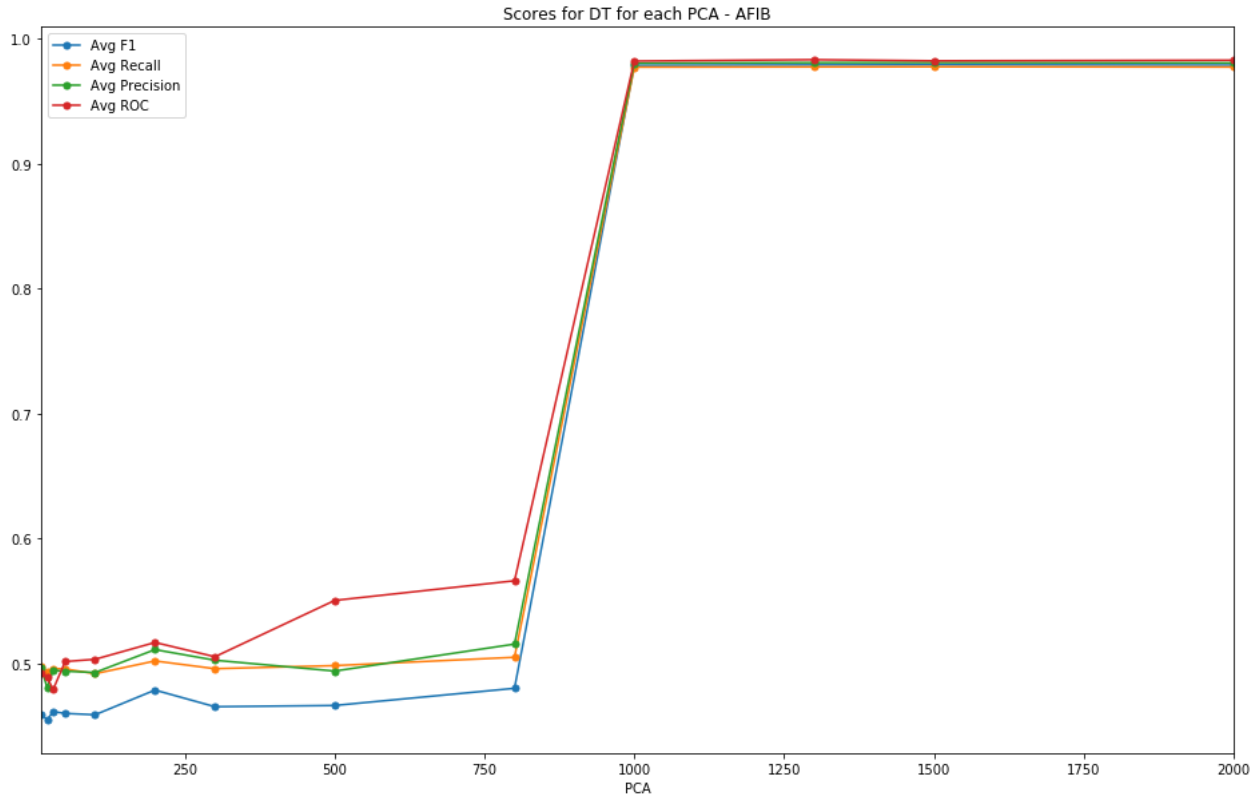

Figure 23: Results for Decision Trees - AFIB

5.4.1.2 Random Forest

Random Forest is an ensemble classifier developed by Leo Breiman that involves integration of multiple decision trees. A multitude of trees is generated during training of the learning model and the output class is the mode of the classes predicted by the individual trees. The larger the number of trees, the higher the accuracy, and the highest-voted prediction is the final output class for the test case. With the help of these random forests, one can correct the habit of overfitting to the training set. In our model we used the following parameters:

- oob_score: True
- n_jobs: -1

- random_state: 101
- n_estimators: (√Number of samples)/2
- criterion: gini index

The results of the same are represented in Figure 24. Similar to Decision Trees, Random Forest as well The model seems to perform optimally when 1000 PCAs are used for training.

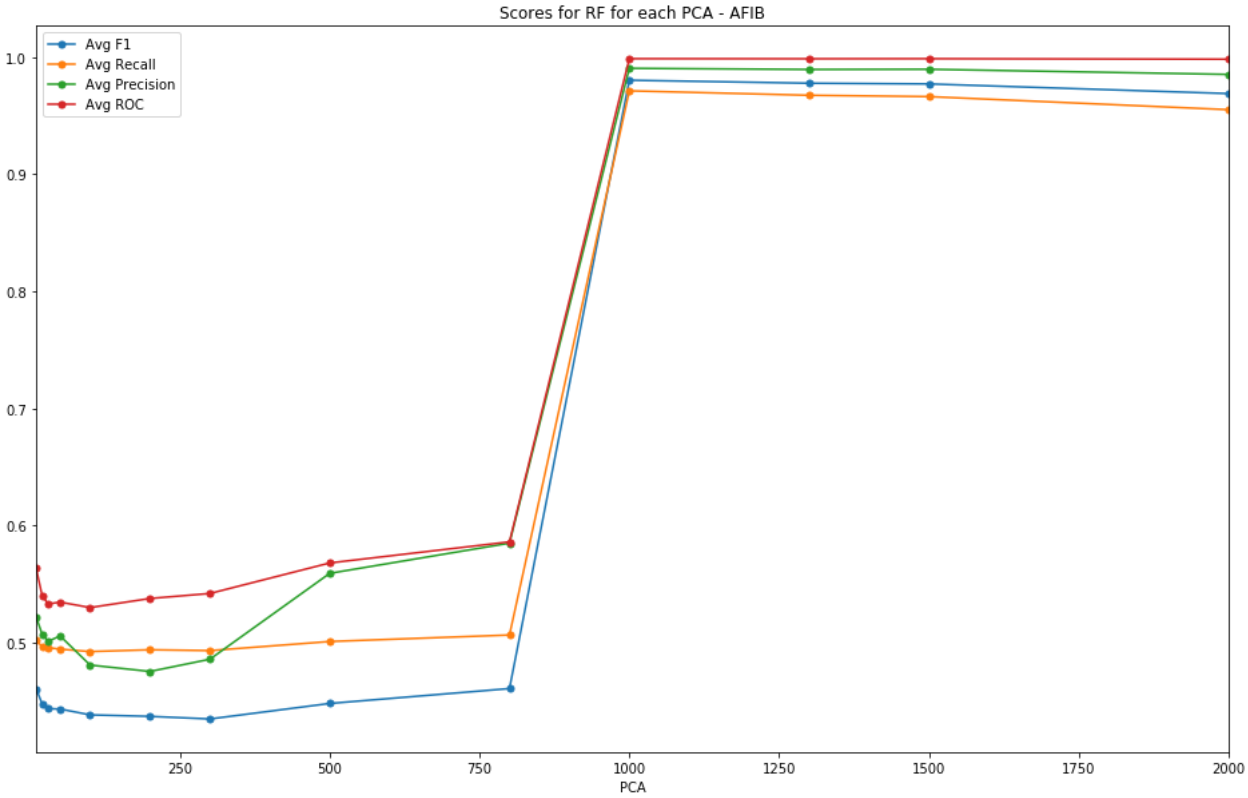

Figure 24: Results for Random Forest - AFIB

5.4.1.3 Naïve Bayes

Naive Bayes algorithm based on Bayes' theorem with the assumption of independence between every pair of features. Naïve Bayes classifiers work well in many real-time. This algorithm requires a small amount of training data to estimate the necessary parameters, hence it performs extremely well with small number of PCA's (to be discussed in section 6.3). Even though naïve bayes is

extremely fast, despite its assumption our features are not independent. Thus does not give accurate. A basic model of Naïve Bayes was used with default parameters.

Figure 25 shows the results for Naïve Bayes Classifier for different number of PCAs. Naïve Bayes seems to perform optimally at 500 PCAs and also the overall macro metrics are lower as compared to other models.

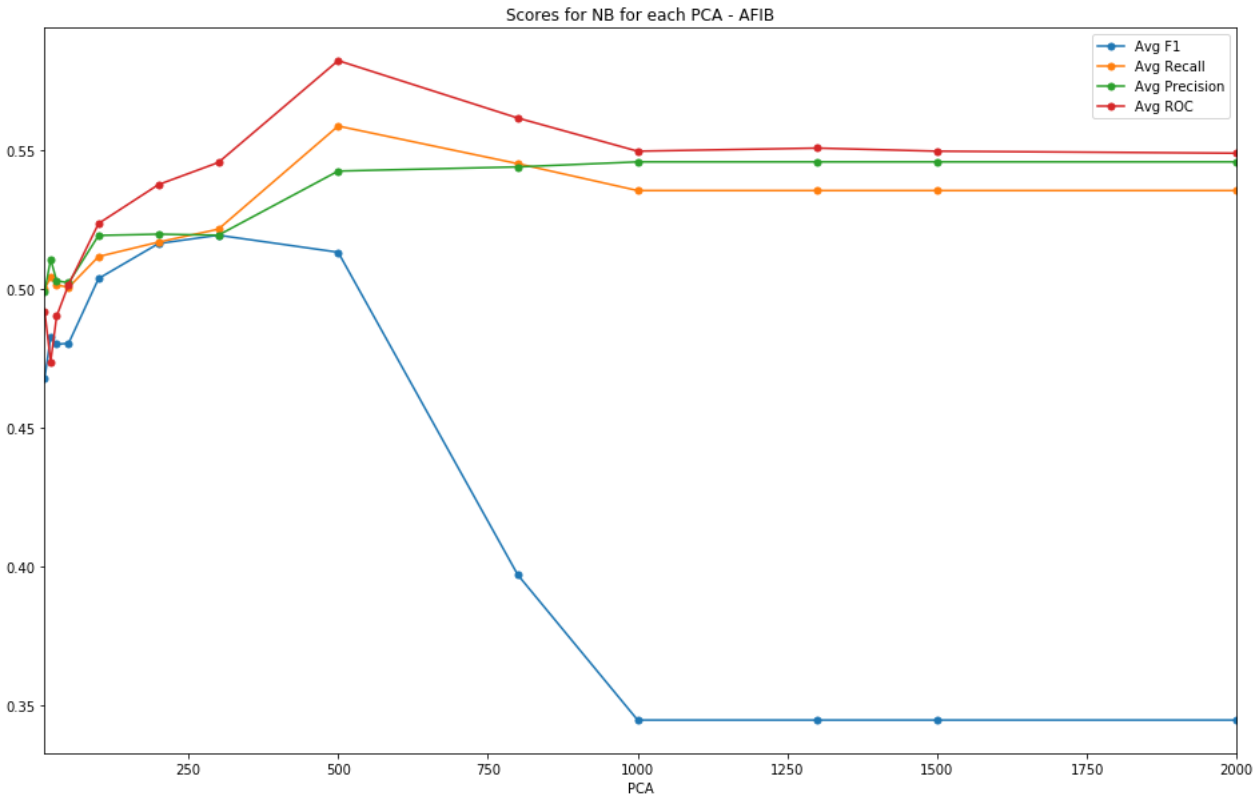

Figure 25: Results for Naive Bayes - AFIB

5.4.1.4 Logistic Regression

Logistic regression is a machine learning algorithm for classification. In this algorithm, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function. It is most useful for understanding the influence of several independent variables on a

single outcome variable. Again with this algorithm, although it is faster than other more advanced algorithms like SVM, RF, and DT. It assumes independence of feature variables just like Naïve Bayes and thus the predictions are not that accurate. A basic model of Logistic Regression was used with default parameters.

Figure 26 represents the results for Logistic Regression Classifier for different number of PCAs. Logistic Regression seems to perform optimally at 1000 PCAs.

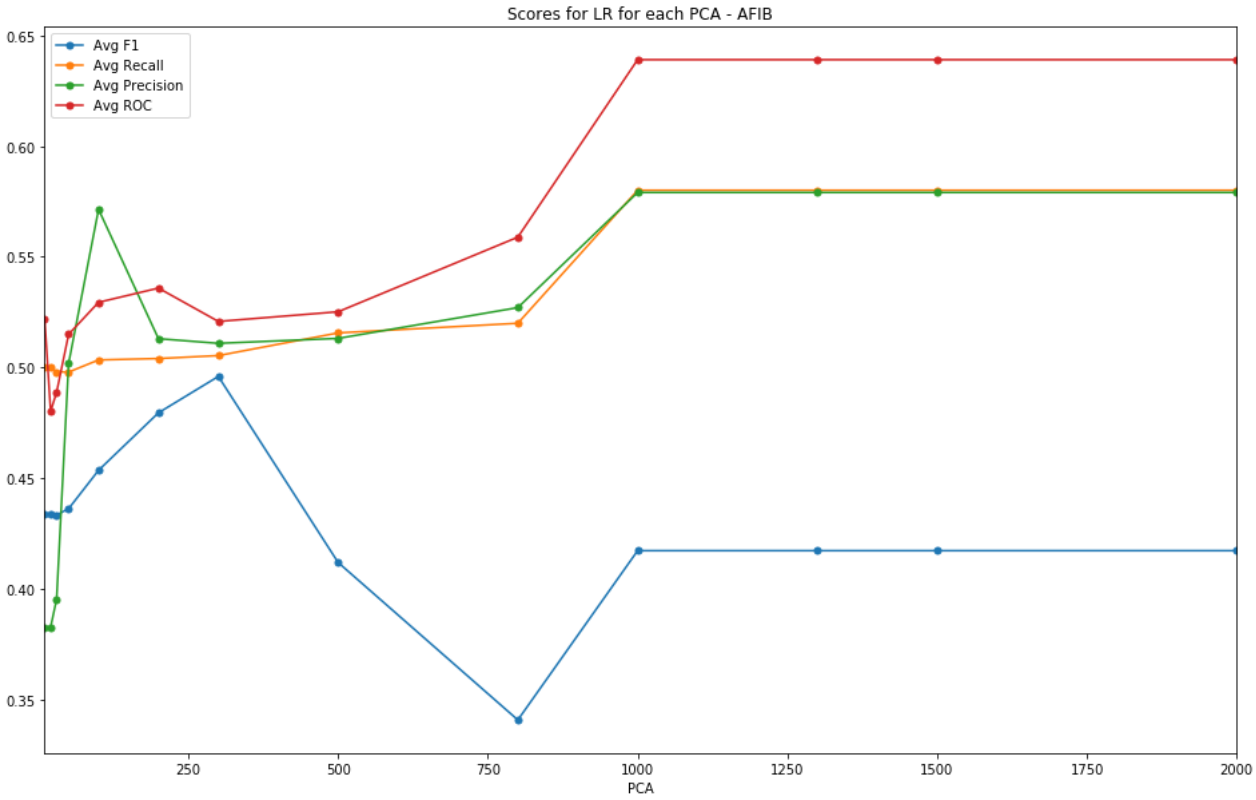

Figure 26: Results for Logistic Regression - AFIB

5.4.1.5 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a class of machine learning algorithms that is apt for large-scale learning. It is an efficient approach towards discriminative learning of linear classifiers under

the convex loss function which is linear (SVM) and logistic regression. The following parameters were used for the same:

- loss: hinge
- n_jobs: -1
- random_state: 101
- n_shuffle: True
- max_iter: 1500

The results of Stochastic Gradient Classifier is shown in Figure 27. Similar to most of the other models even the Stochastic Gradient Descent performs optimally for 1000 PCAs.

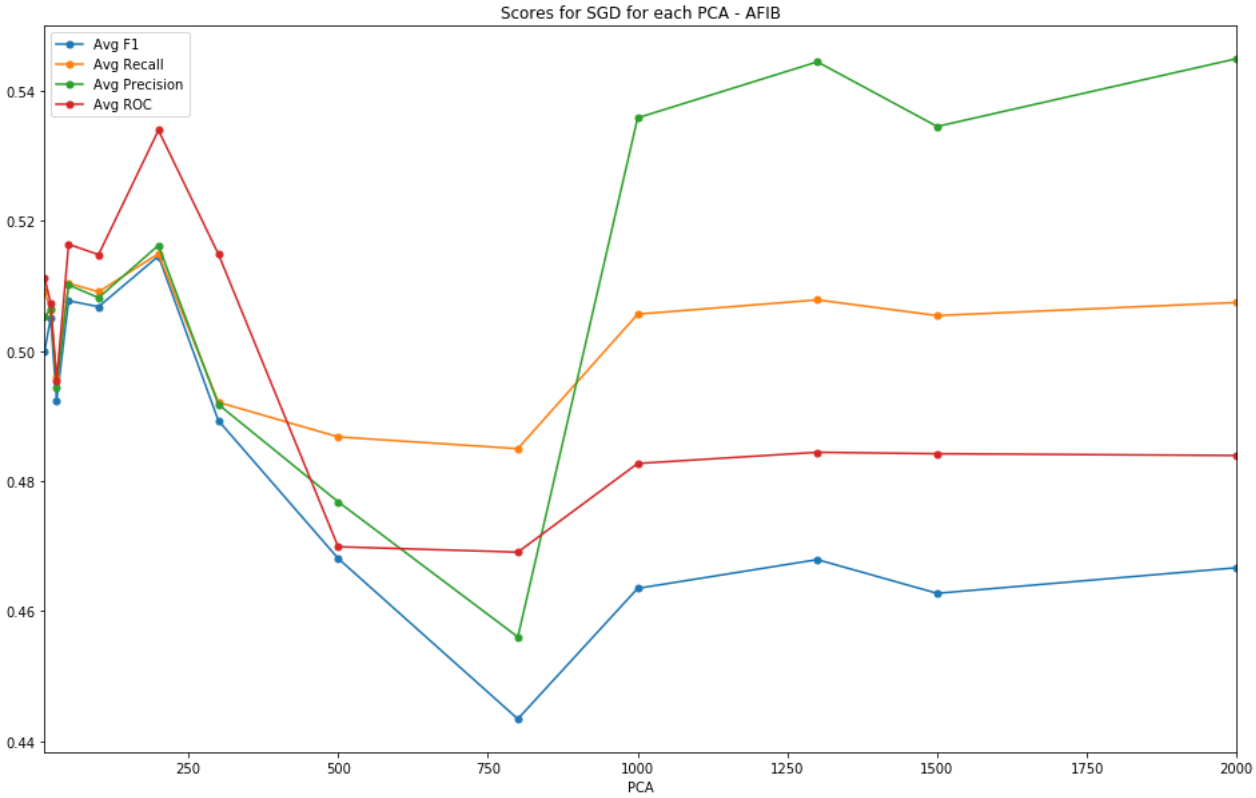

Figure 27: Results for Stochastic Gradient Descent - AFIB

5.4.1.6 Comparison

The Figure 28 compares the F1 score of the five machine learning models tested for various PCAs. As we can see in the figure, for the initial iterations we can see that linear models like Naïve Bayes, Logistic Regression and Stochastic Gradient Descent work slightly better, than Random Forests and Decision Trees but as we increase the amount of the data, Decision trees and Random Forests dominate over the other algorithms. This could be justified by the fact that with the increase in dimensionality we are adding to the non-linearity of the data which is better recognized using models like Decision Trees and Random Forests rather than the linear models discussed above.

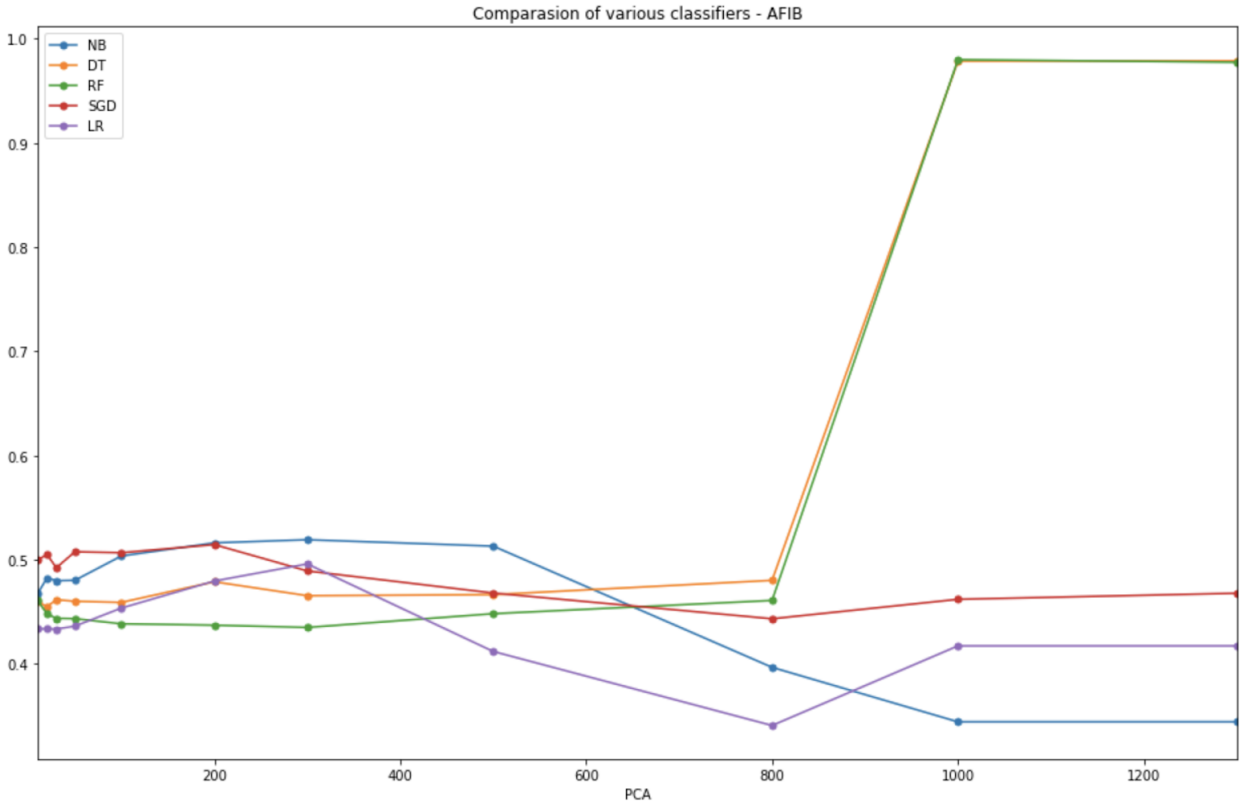

Figure 28: Comparison of different types of algorithms based on F1 Scores - AFIB

5.4.1.7 Code

```
def doClassification(X,Y, adr, lim):
```

```
    k = int(math.sqrt(len(X))//2)
    classifiers = [
                GaussianNB(),
                DecisionTreeClassifier(max_depth=10, random_state=101, min_samples_leaf=15),
                RandomForestClassifier(random_state=101, n_jobs=-1, oob_score=True,
n_estimators=k),
                SGDClassifier(max_iter=1500, loss="hinge", n_jobs=-1, n_shuffle=True,
random_state=101),
                LogisticRegression(random_state=0)
               ]

    classifierNames = ["NB", "DT", "RF", "SGD", "LR"]

    for i in range(len(classifiers)):
        clf = classifiers[i]
        clfName = classifierNames[i]

        scores = cross_validate(clf, X, Y, cv=5, scoring=scoring, return_estimator=True)

        array = [[0,0],[0,0]]
        for i in range(5):
            estimator = scores['estimator'][i]
            y_pred = estimator.predict(X)
            array += confusion_matrix(Y, y_pred)

        array = array // 5

        sn.set()
        df_cm = pd.DataFrame(array)
        plt.figure(figsize = (7,6))
        ax = sn.heatmap(df_cm, annot=True)
```

```
        plt.title(clfName + ' - ' + adr)
        plt.show()



warnings.filterwarnings('ignore')
lims = [10, 20, 30, 50, 100, 200, 300, 500, 800, 1000, 1300, 1500, 2000, 2500, 3000, 3500, 4000,
4500, 5000]

scoring = ['f1_macro', 'recall_macro', 'precision_macro', 'roc_auc']

 for lim in lims:
    X = pca.iloc[:, 0:lim]
    for adr in label.columns:
            Y = label[adr]
            scores = doClassification(X,Y, adr, lim)
```

5.4.2 Artificial Neural Network

For our second approach to make the classification more efficient by using one single model rather than 243 different models, we tried multilabel classification using Artificial Neural Network. The advantage of using a shared network is that the correlations between classes or labels can be learned, and specifically a shared feature space can be represented by the network. Which generally leads to a drastic speed-up in the performance.

The details and results obtained using this approach are described in detail in this section.

5.4.2.1 Structure

Figure 29 illustrates the structure of the Artificial Neural Network used. The number in each layer indicates the number of neurons present in that particular layer. **Relu** activation function was used in between the layers rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimize with gradient-based methods. We also used a batch normalization layer which allows much higher learning rates, increasing the speed at which network trains adding another dropout layer to avoid overfitting. The last layer used Sigmoid activation function and we also used Binary Cross Entropy loss.

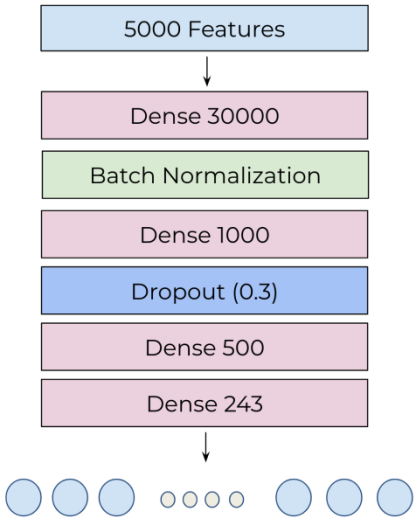

Figure 29: Structure of the ANN

5.4.2.2 Code

```
def baseline_model():
    model = Sequential()
        model.add(Dense(30000, input_dim=5000, activation='relu'))
        model.add(BatchNormalization())
        model.add(Dense(1000, activation='relu'))
        model.add(Dropout(0.3))
        model.add(Dense(500, activation='relu'))
        model.add(Dense(243, activation='sigmoid'))
opt = SGD(lr=0.1, momentum=0.9)
```

```
metrics=['accuracy'])
    print(model.summary())
    return model

estimator = KerasClassifier(build_fn=baseline_model, epochs=10, batch_size=10000, verbose=1)
kfold = KFold(n_splits=5, shuffle=True)
results = cross_val_score(estimator, X, label , cv=kfold)
print(results)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

5.4.3 Evaluation

We also used **fivefold cross** validation to evaluate the performance of the generated predictive models and avoid overfitting. Here, the original dataset was randomly partitioned into 5 equivalent sized folds or subsamples. One of the subsamples was considered as the validation set and was retained, the left overs were used as part of the training set. The process was repeated five times until each subsample had been once used as validation data. At the end, the results obtained from all fivefold were averaged to get a single result.

The data used for generating the models is severely imbalanced, and **imbalanced data** classification would lead to predictions biased towards the negative class which is typically the dominating class. To solve this issue we used **macro** scores for f1, recall, precision and roc_auc for evaluation as they calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

CHAPTER 6

RESULTS AND DISCUSSION

This chapter is a compilation of all the major results along with associated discussion.

6.1 SMILES

This section details the results gathered when SMILEs data was subjected to various representation learning methodologies.

6.1.1 Atomic Encoding

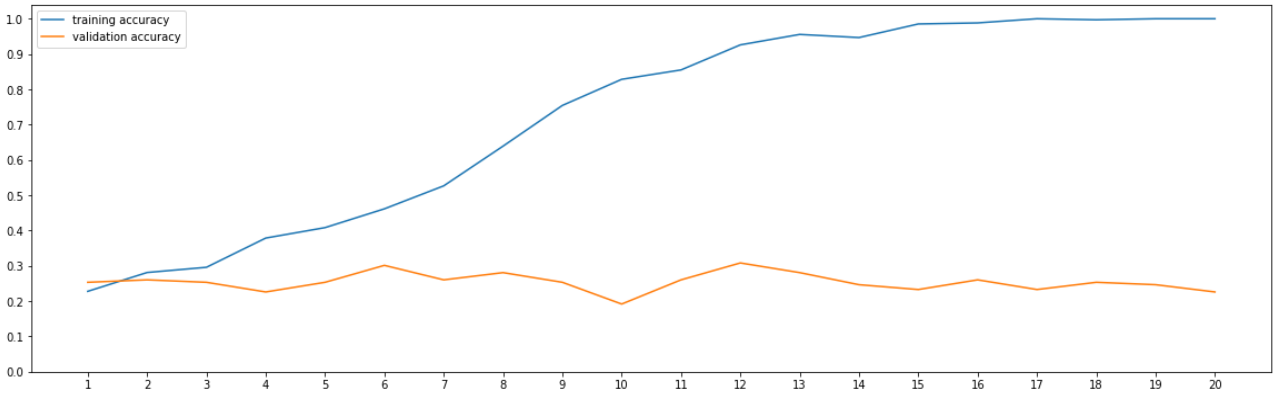

Figure 30: Training and validation accuracy for the Atomic Encoding Model

The training and testing accuracies are plotted in the Figure 30. It can be observed that no significant learning happens over the course of the training as the validation accuracy fluctuates irregularly indicating that the character encoding does not contain the information needed to ascertain the functional group.

6.1.2 Substructure Encoding

The procedure from substructure encoding (detailed in 5.3.1.2 Substructure Encoding) is used to create embeddings for each substructure. A molecule level embedding is obtained by averaging the embeddings of the individual substructures present in it.

There is a large class imbalance in the ATC classes. Down sampling is employed to create a balanced datasets for all evaluations. We have detailed the accuracies obtained by our drug embeddings when tested against the top 3 (Figure 31), top 4 (Figure 32) and all 14 (Figure 33) ATC classes below.

The Top 3 and Top 4 ATC classes were chosen because of the balanced nature of the resulting dataset. As observed in Figure 6, the dataset containing all 14 ATC classes is very skewed.

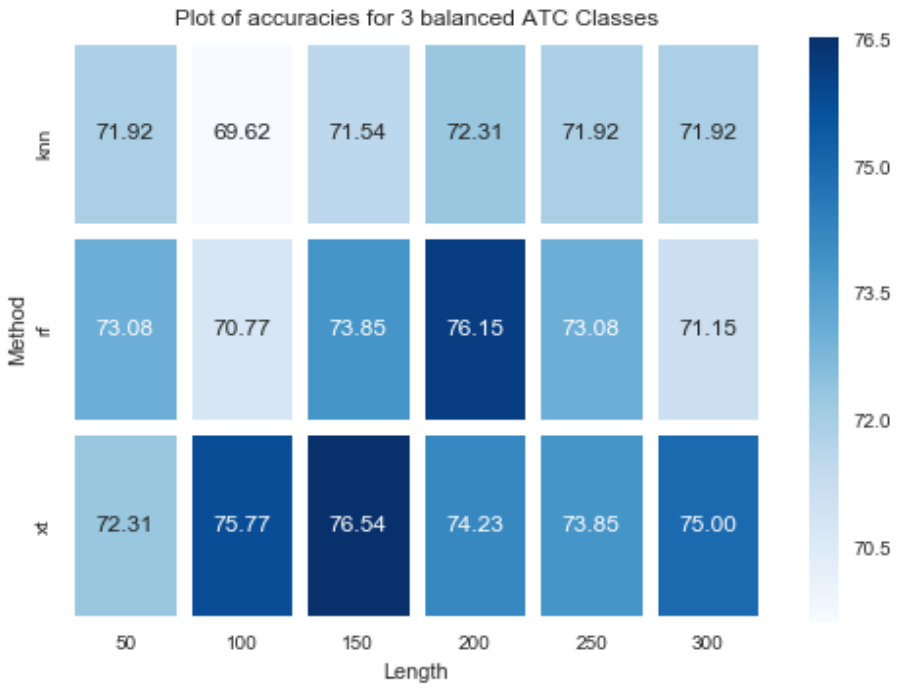

Figure 31: Plot of accuracies for top 3 balanced ATC classes

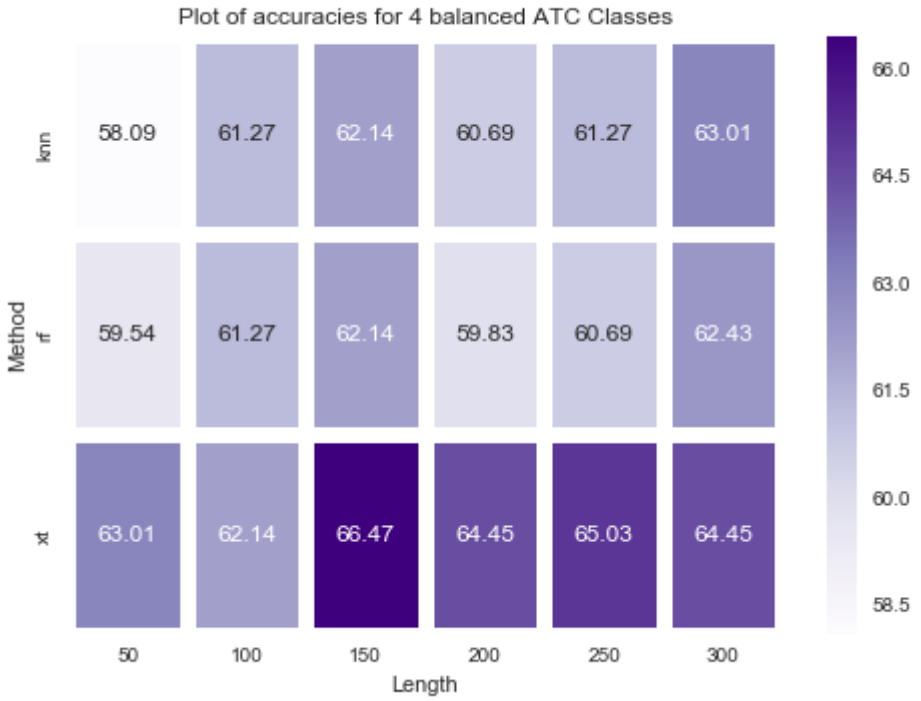

Figure 32: Plot of accuracies for top 4 balanced ATC classes

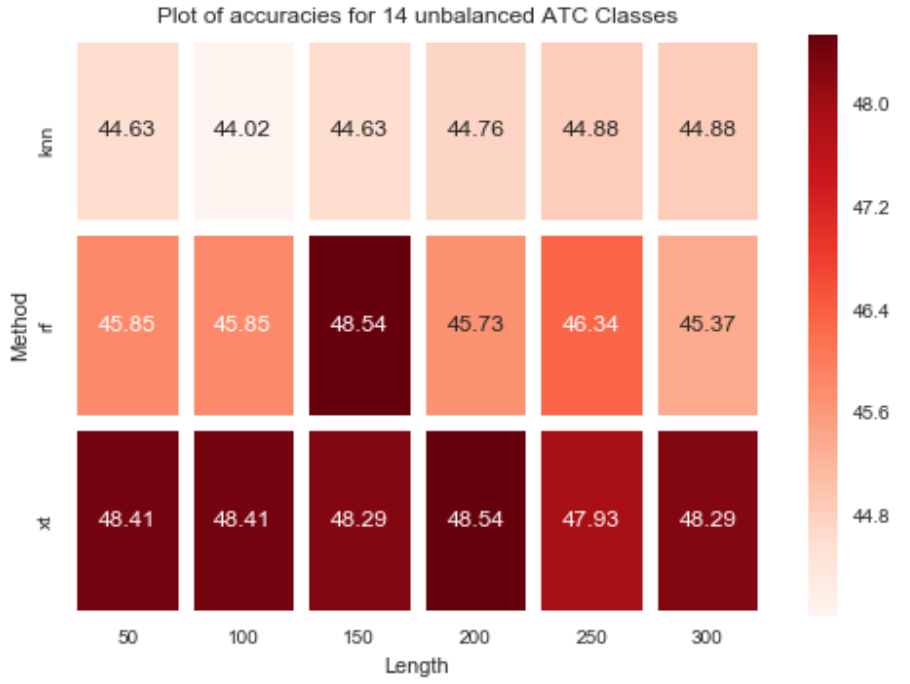

Figure 33: Plot of accuracies for 14 un-balanced ATC classes

From Figure 31, Figure 32 and Figure 33 we observe that on a whole, extra trees outperforms the other two methods (KNN and Random Forest). The drug embedding length of 100 dimensions appears to work well with all classifier and gives the highest accuracy of all the combinations in most cases.

The motive behind creating a representation of a drug is to be able to use this embedding for more than one application. To validate if the embedding is versatile, we tested how well it would perform on an unseen dataset like SIDER.

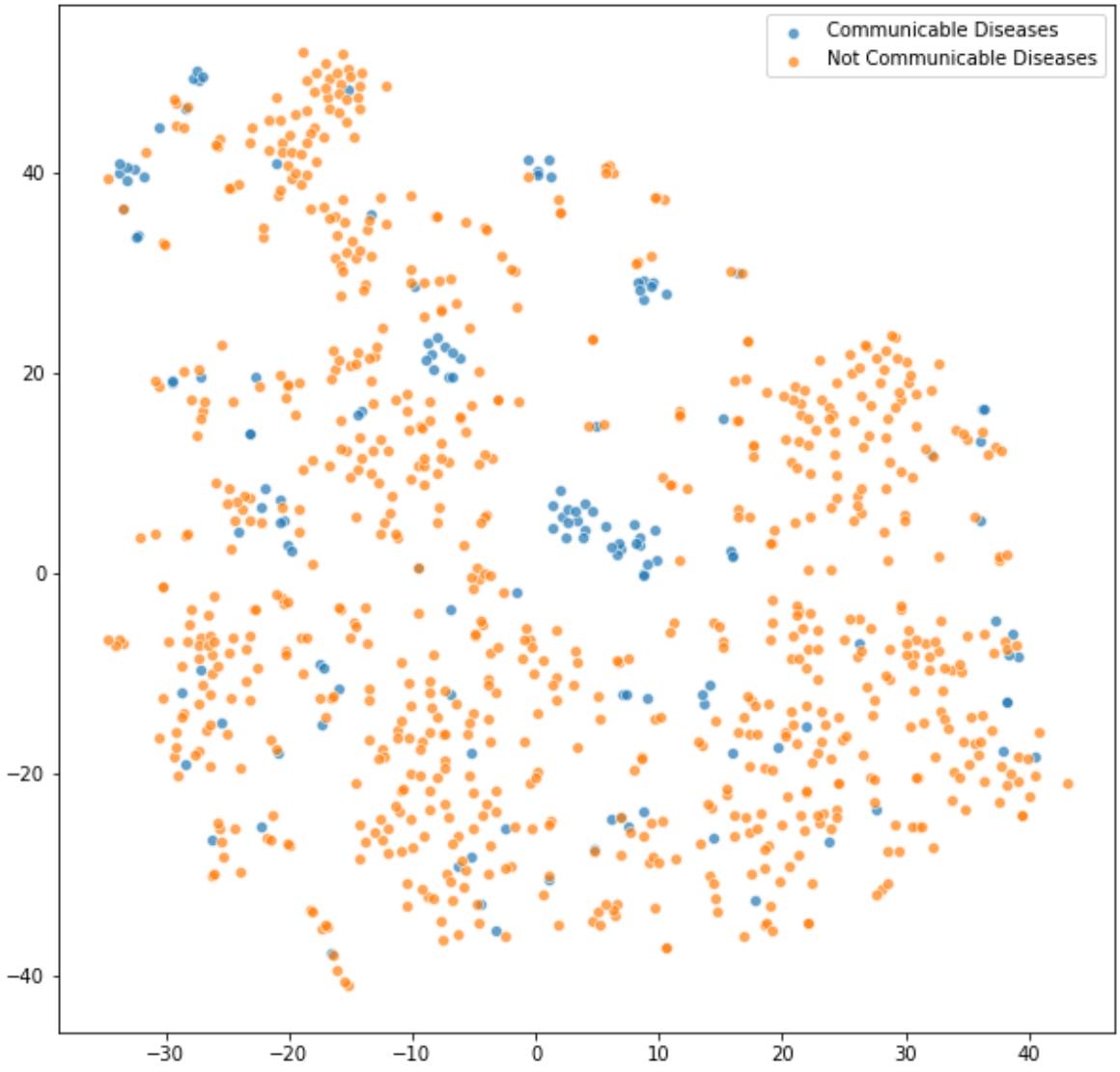

Figure 34: t-SNE plot of substructure embedding against SIDER (communicable disease)

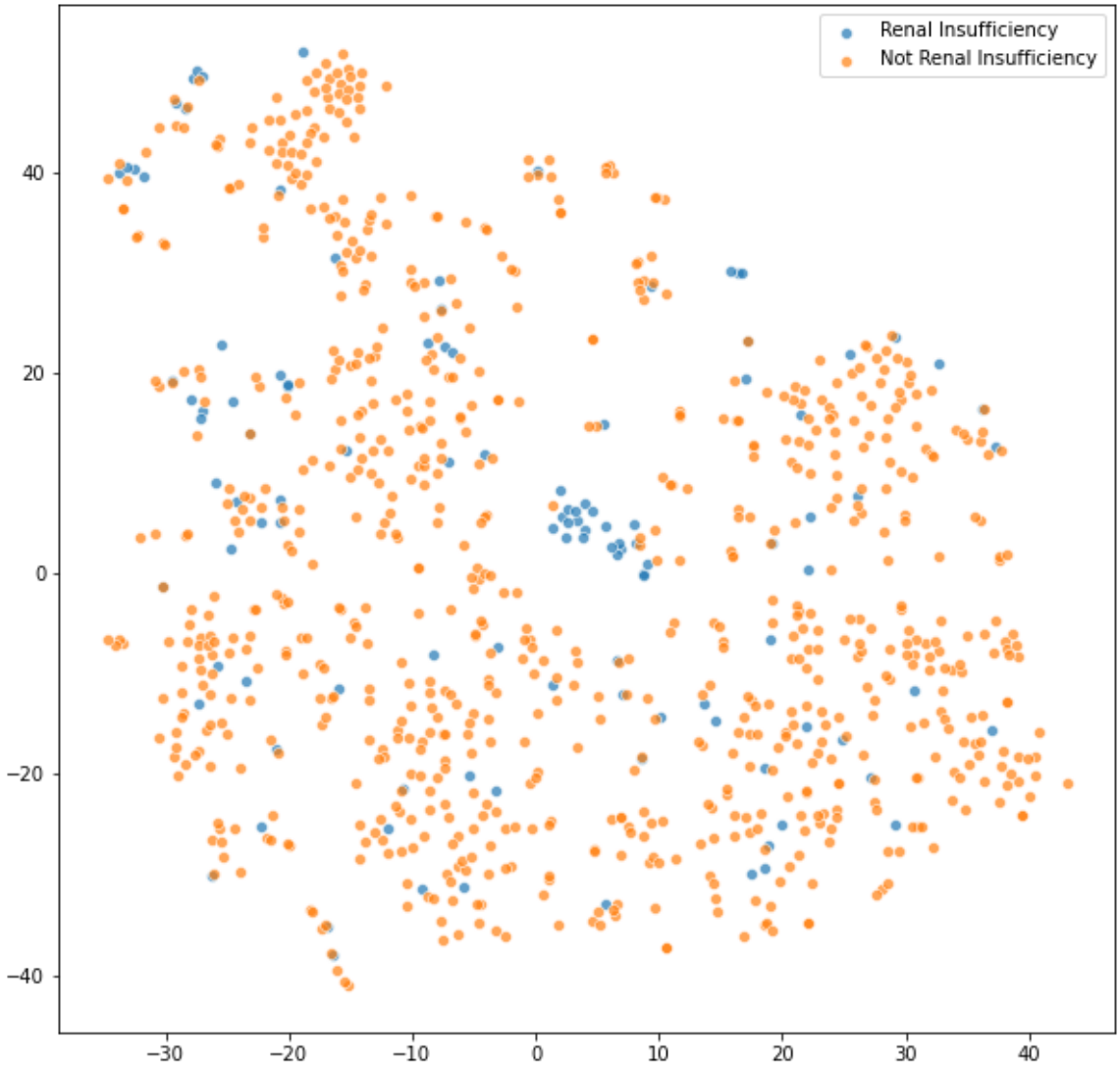

Figure 35: t-SNE plot of substructure encoding against SIDER (renal insufficiency)

Figure 34 and Figure 35 show that our representations perform well even though the dataset is extremely skewed. In Figure 34, the number of drug embeddings with side effects being communicable diseases are quite low but they still manage to capture the cluster quite well. A KNN can be used to predict this property.

6.2 LINCS

This section details the results gathered when the gene expression data obtained from LINCS was subjected to various representation learning methodologies.

6.2.1 Models trained on individual cell lines

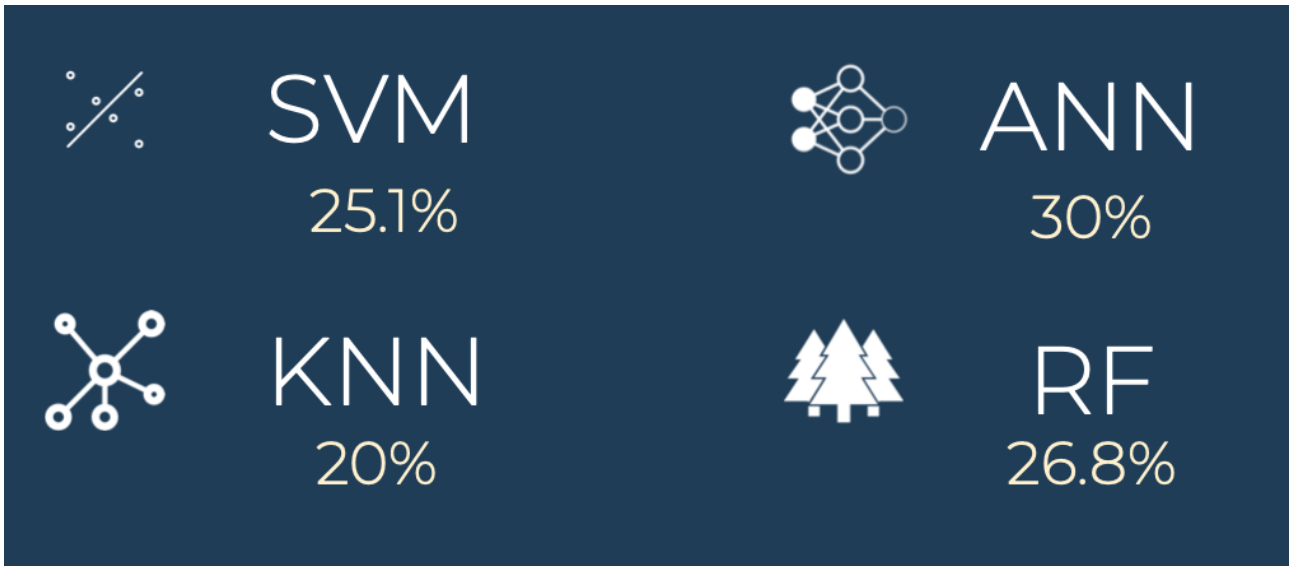

Figure 36: Average accuracy of baseline models trained on L1000 Assay for the top 3 ATC classes (L, M, N)

6.2.2 Densely Connected Triplet Network

Figure 37: t-SNE plot of embeddings obtained from the densely connected triplet network

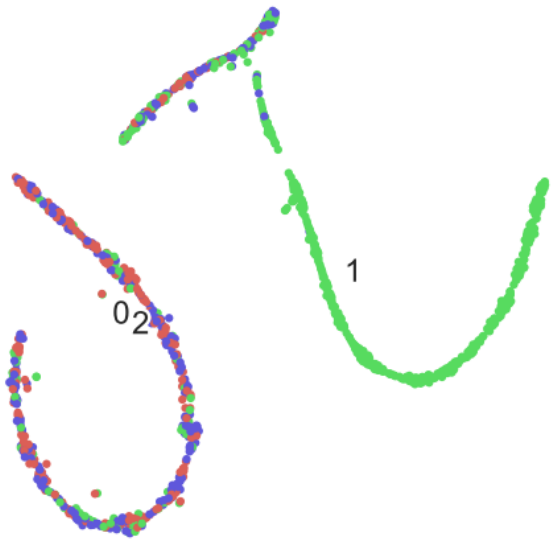

Figure 38: t-SNE plot of the clustering ability of the above embeddings after passing through a simple neural network

For the top 3 ATC classes (L, M, N), the train and test accuracies are tabulated in the table below.

Table 6: Accuracy of classification of Top 3 ATC classes for embeddings generated by densely connected triplet network

Training Accuracy	70%
Testing Accuracy	65%

The low accuracies can be explained by the fact that LINCS is very noisy in nature and Is very robust to perturbations. More powerful techniques must be used to capture the similarities between these gene signatures.

6.3 ADR Prediction

This section presents the results of the approaches we tried for ADR prediction.

6.3.1 Machine Learning Model

The training was done several times, with different number of PCA's for the feature set, to determine the optimal number of principle components required to get the best results. The results of the same are plotted in Figure 39 and Figure 40 for ADRs 'AFIB' and 'Abnormal Gait' and the scores and classifier names are tabulated in **Error! Not a valid bookmark self-reference.** and Table 8. From these tables we can clearly see that Random Forests and Decision Tree Classifiers outperformed the other classifiers, with both algorithms performing optimally at 1000 PCAs input.

Table 7: Macro scores and names of best classifiers for each PCA iteration-AFIB

	Avg F1	Avg Precision	Avg ROC	Avg Recall	Model	PCA
0	0.499884	0.505150	0.511236	0.509188	SGD	10.0
1	0.505066	0.506475	0.507318	0.506324	SGD	20.0
2	0.492304	0.494426	0.495427	0.496058	SGD	30.0
3	0.507686	0.510121	0.516372	0.510396	SGD	50.0
4	0.506770	0.508152	0.514774	0.509056	SGD	100.0
5	0.516226	0.519610	0.537486	0.516757	NB	200.0
6	0.519241	0.519247	0.545460	0.521409	NB	300.0
7	0.513104	0.542370	0.582198	0.558611	NB	500.0
8	0.480254	0.515607	0.566279	0.505060	DT	800.0
9	0.980047	0.990215	0.998433	0.971000	RF	1000.0
10	0.978923	0.980727	0.983121	0.977312	DT	1300.0
11	0.979046	0.980828	0.982181	0.977454	DT	1500.0
12	0.978762	0.980524	0.982698	0.977193	DT	2000.0
13	0.979479	0.981525	0.982390	0.977619	DT	2500.0
14	0.979554	0.981734	0.983155	0.977571	DT	3000.0
15	0.979558	0.981690	0.982531	0.977614	DT	3500.0
16	0.978729	0.980390	0.982379	0.977260	DT	4000.0
17	0.978803	0.980598	0.982841	0.977212	DT	4500.0
18	0.979512	0.981763	0.982957	0.977467	DT	5000.0

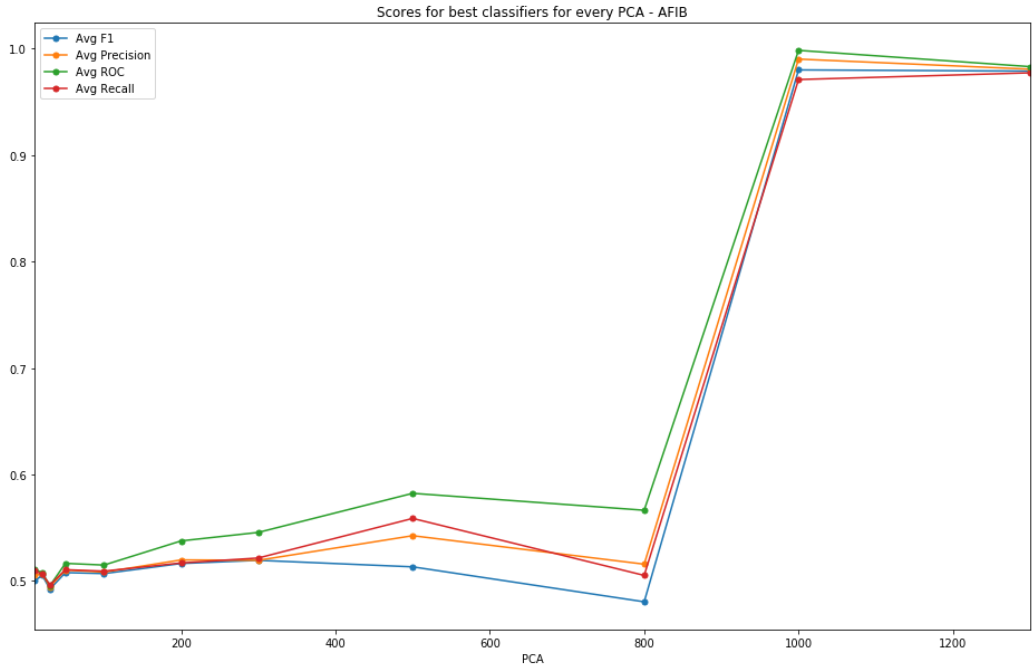

Figure 39: Macro scores for best classifiers in every PCA iteration-AFIB

Table 8: Macro scores and names of best classifiers for each PCA iteration- Abnormal Gait

	Avg F1	Avg Precision	Avg ROC	Avg Recall	Model	PCA
0	0.500026	0.509869	0.519728	0.507284	SGD	10.0
1	0.504873	0.508627	0.502832	0.507687	SGD	20.0
2	0.508451	0.516667	0.523914	0.511452	NB	30.0
3	0.507661	0.514920	0.536360	0.510580	NB	50.0
4	0.513154	0.518606	0.534440	0.514295	NB	100.0
5	0.516828	0.521183	0.540715	0.517203	NB	200.0
6	0.524500	0.524872	0.558265	0.526957	NB	300.0
7	0.517140	0.548763	0.606001	0.572846	NB	500.0
8	0.476577	0.500382	0.567656	0.500204	DT	800.0
9	0.991212	0.991991	0.994826	0.990446	DT	1000.0
10	0.991212	0.991991	0.994269	0.990446	DT	1300.0
11	0.991212	0.991991	0.994577	0.990446	DT	1500.0
12	0.991350	0.991901	0.994313	0.990805	DT	2000.0
13	0.991350	0.991901	0.994280	0.990805	DT	2500.0
14	0.991350	0.991901	0.994279	0.990805	DT	3000.0
15	0.991389	0.992118	0.994957	0.990671	DT	3500.0
16	0.991212	0.991991	0.994466	0.990446	DT	4000.0
17	0.991212	0.991991	0.994227	0.990446	DT	4500.0
18	0.991478	0.992156	0.994502	0.990809	DT	5000.0

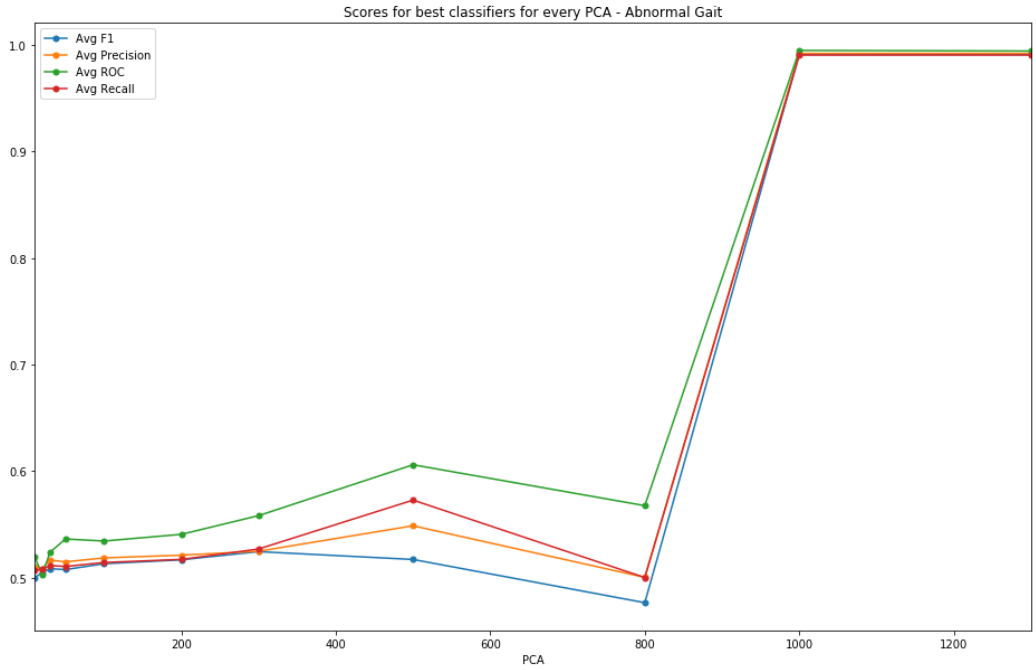

Figure 40: Macro scores for best classifiers in every PCA iteration-Abnormal Gait

The confusion matrix for Random Forest Classifier and Decision Tree for AFIB and Abnormal Gait for 1000 PCA represented in Figure 39 and 40 respectively. Here we can see clearly that the classifiers are successfully able to classify the data correctly.

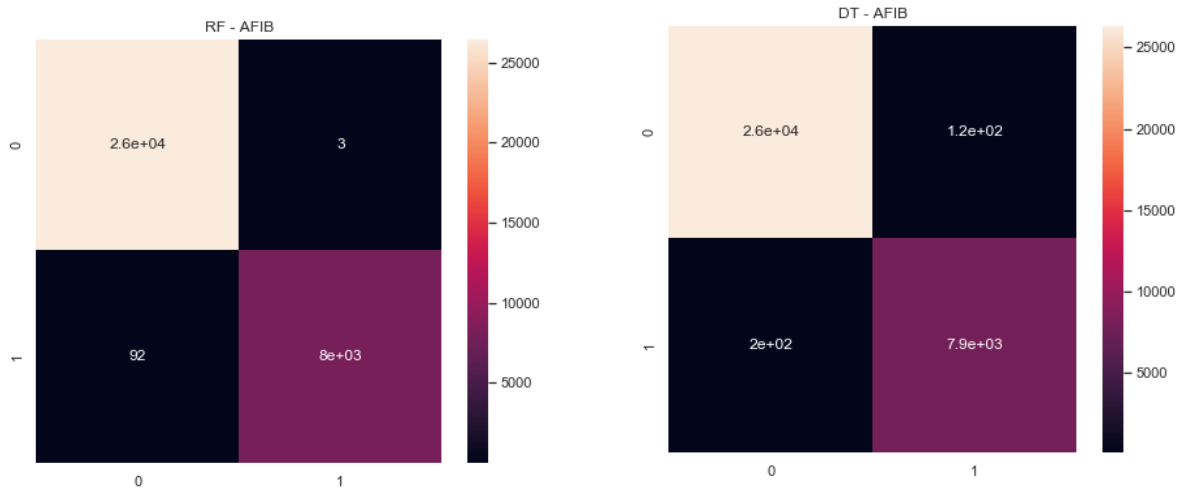

Figure 41: Confusion matrix of RF and DT for 1000 PCAs -AFIB

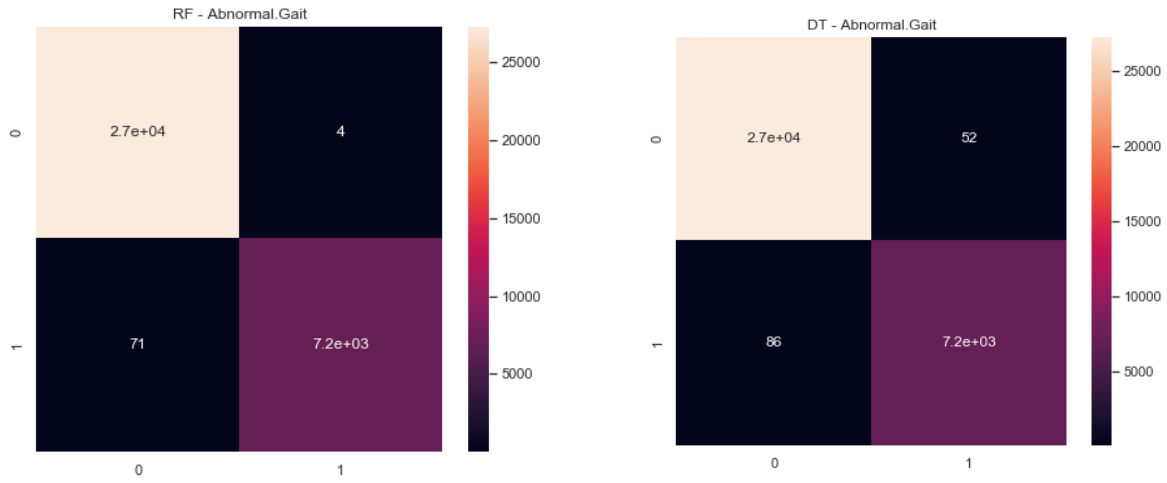

Figure 42: Confusion matrix of RF and DT for 1000 PCAs -Abnormal Gait

6.3.2 Artificial Neural Network

In Figure 43 and Figure 43: Graph of testing and training accuracyFigure 44 represent the accuracy and loss achieved per epoch during training and testing phases. We can see that the model has comparable performance on both train and validation datasets (labelled test). The averaged testing and training accuracies for five fold cross validation are listed in Table 9.

Table 9: Training and testing accuracies

Phase	Accuracy
Training	81.6% (0.14%)
Testing	82.2% (0.23%)

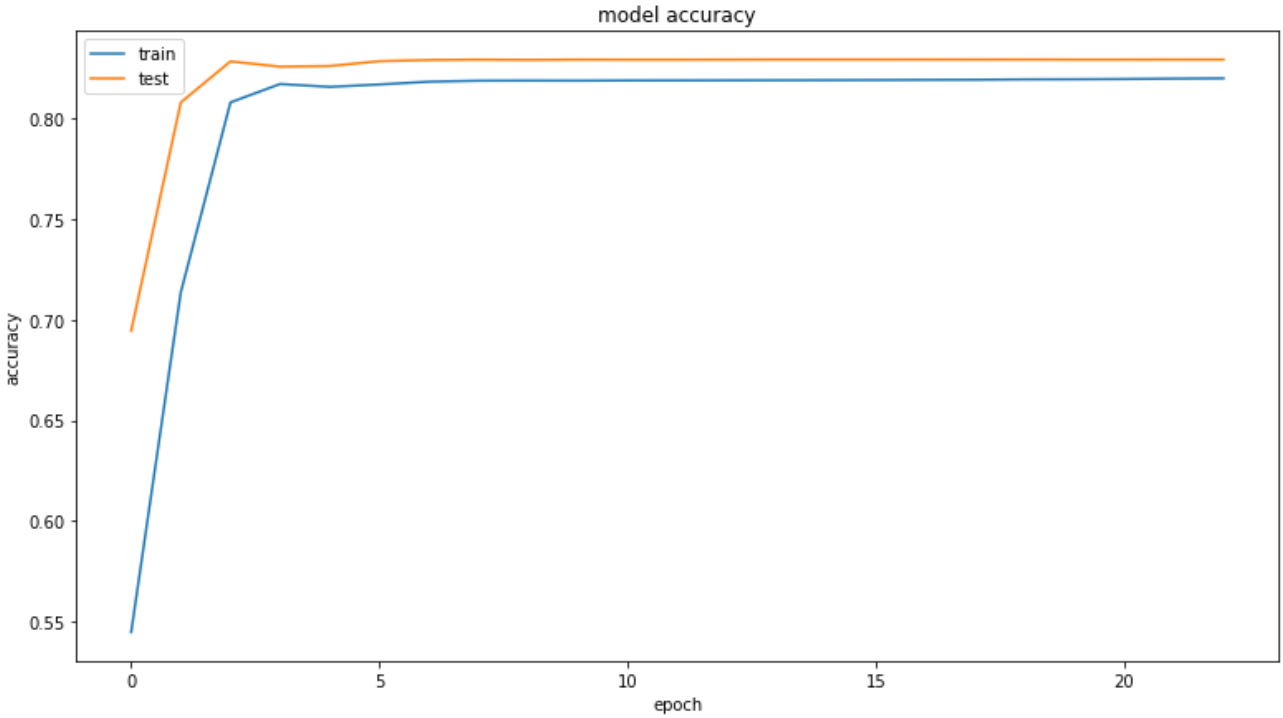

Figure 43: Graph of testing and training accuracy

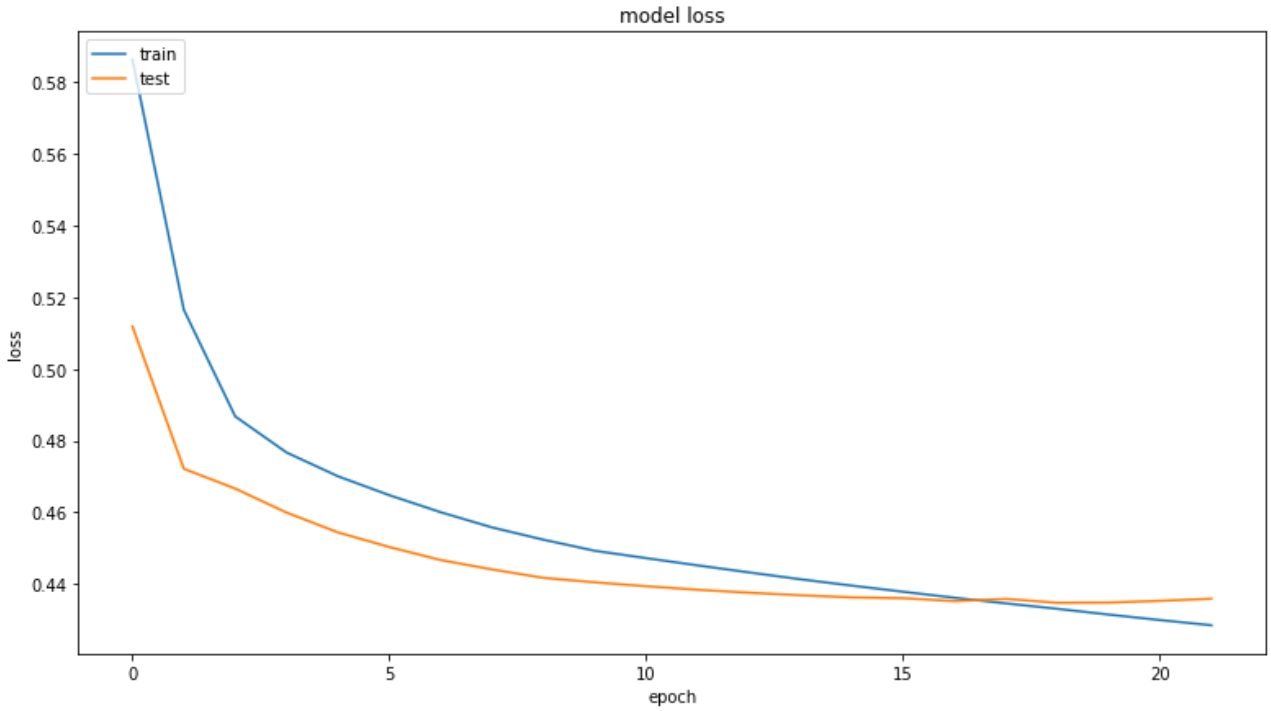

Figure 44: Graph of testing and training losses

From this we can derive that the artificial neural network proves to be successful in predicting the ADRs with high accuracy and low loss.

CHAPTER 7

CONCLUSION AND FUTURE WORK

SMILES contains important structural information about a chemical. The way a drug interacts with its environment is highly dependent on its structure. Therefore, SMILES makes a good candidate for creating good drug embeddings. These embeddings are created independent of any sort of classification information at the time of representation learning so they are not restricted to ATC classes. When the embeddings were tested on the SIDER dataset for many different ADRs, we observed that the embeddings were capable of clustering the smiles with and without the dataset accurately despite SIDER's extremely skewed nature.

Gene expression is notoriously hard to embed as it has seemingly no apparent correlation on its own. However deep learning can extract those patterns to a certain extent. However, that may not be enough to obtain a powerful embedding, capable of being used for many classifications. Advanced models can be used to generate better embeddings by conditioning generative adversarial networks (GANs) with gene expression signatures and SMILES data which can help in generating new molecules.

This study also focused on generating machine-learning based classification models for the prediction of Adverse Drug Reactions caused by drug pairs. We represented the drug using three types of data namely, Gene expression, MACCS and Gene Ontology. Five state of the art machine-learning algorithms and one Artificial Neural Network algorithm were used to generate computational models trained for 243 ADRs. Out of which Random Forest and Decision Tree classifier outperformed the other four when trained for 1000 PCA reduced features. In conclusion, the proposed machine learning based data-integration approach could be a promising method for the prediction of potential ADRs prior to preclinical testing stages.

This work can be expanded to predicting targets to other chemical or physical properties by using them against different, more diverse classification systems.

Extended Connectivity FingerPrinting (ECFP) are one of the most common molecular fingerprinting methods available today. It could be used to complement or enhance our embedding. Randomized SMILES and DeepSMILES variants of SMILES could also be used to achieve these embeddings

Substructure encoding can be combined with ProtVec which is a continuous distributed representation of biological sequences and can be used for proteochemometrics approaches such as such as family classification, protein visualization, structure prediction, disordered protein identification, and protein-protein interaction prediction.

This study can be extended in multiple directions in the future in terms of both features and models. Sometime, the severity of a particular ADR is available in the SIDER dataset, which can be taken into account during model development. At the same time, some of the more advanced, deep learning models can be implemented to do this task execute feature engineering on it own. As discussed in 5.2.2 ADR Prediction SVM can be tried on high end machines, also other versi0ons of SVM like LibSVM and LiquidSVM which have proven to be faster, more advanced and more space efficient can be tried out. The paper “Predicting adverse drug reactions of combined medication from heterogeneous pharmacologic databases talks about Highly-Credible-Negative suing which they achieved significant performance improvements for machine learning models, This can be one approach to be followed in the future. Furthermore, other types of available data sources such as chemical-protein binding that can be integrated into our models for a data-driven approach for ADR prediction.

BIBLIOGRAPHY

[1] Z. Xu, S. Wang, F. Zhu and J. Huang, "Seq2seq Fingerprint: An Unsupervised Deep Molecular Embedding for Drug Discovery," 2017.

[2] M. J. Keiser, "Predicting New Molecular Targets for Known Drugs," *Nature*, 2009.

[3] M. . Campillos, M. . Kuhn, A.-C. . Gavin, L. J. Jensen, L. J. Jensen and P. . Bork, "Drug target identification using side-effect similarity," *Science*, vol. 321, no. 5886, pp. 263-266, 2008.

[4] G. B. Goh, N. O. Hodas, C. . Siegel and A. . Vishnu, "SMILES2Vec: An Interpretable General-Purpose Deep Neural Network for Predicting Chemical Properties.," *arXiv: Machine Learning*, vol. , no. , p. , 2017.

[5] Y.-F. Zhang, X. Wang, A. C. Kaushik, Y. Chu, X. Shan, M.-Z. Zhao, Q. Xu1 and D.-Q. Wei, "SPVec: A Word2vec-Inspired Feature Representation Method for Drug-Target Interaction Prediction," *Frontiers in Chemistry*, 10 January 2020.

[6] S. . Jaeger, S. . Fulle and S. . Turk, "Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition," *Journal of Chemical Information and Modeling*, vol. 58, no. 1, pp. 27-35, 2018.

[7] C. R. D. Pierri and R. Voyceik, "SWeeP: Representing large biological sequences datasets in compact vectors," *Scientific Reports*, 9 January 2020.

[8] Y. . Donner, S. . Kazmierczak and K. . Fortney, "Drug Repurposing Using Deep Embeddings of Gene Expression Profiles," *Molecular Pharmaceutics*, vol. 15, no. 10, pp. 4314-4325, 2018.

[9] C.-T. Huang, "A Large-Scale Gene Expression Intensity-Based Similarity Metric for Drug Repositioning," *iScience*, vol. 7, 2018.

[10] J. . Du, P. . Jia, Y. . Dai, C. . Tao, Z. . Zhao and D. . Zhi, "Gene2vec: distributed representation of genes based on co-expression," *BMC Genomics*, vol. 20, no. 1, p. 82, 2019.

[11] A. . Aliper, S. M. Plis, A. . Artemov, A. . Ulloa, P. . Mamoshina and A. . Zhavoronkov, "Deep Learning Applications for Predicting Pharmacological Properties of Drugs and Drug Repurposing Using Transcriptomic Data," *Molecular Pharmaceutics*, vol. 13, no. 7, pp. 2524-2530, 2016.

[12] E. . Asgari, M. R. K. Mofrad and M. R. K. Mofrad, "Continuous Distributed Representation of

Biological Sequences for Deep Proteomics and Genomics," *PLOS ONE*, vol. 10, no. 11, p. , 2015.

[13] F. . Napolitano, Y. . Zhao, V. M. Moreira, R. . Tagliaferri, J. . Kere, M. . D'Amato, D. . Greco and D. . Greco, "Drug repositioning: a machine-learning approach through data integration," *Journal of Cheminformatics*, vol. 5, no. 1, pp. 30-30, 2013.

[14] L. . Yang and P. K. Agarwal, "Systematic Drug Repositioning Based on Clinical Side-Effects," *PLOS ONE*, vol. 6, no. 12, p. , 2011.

[15] Y. . Zheng, H. . Peng, X. . Zhang, Z. . Zhao, J. . Yin and J. . Li, "Predicting adverse drug reactions of combined medication from heterogeneous pharmacologic databases," *BMC Bioinformatics*, vol. 19, no. 19, p. 517, 2018.

[16] S. . Dey, H. . Luo, A. . Fokoue, J. . Hu and P. . Zhang, "Predicting adverse drug reactions through interpretable deep learning framework," *BMC Bioinformatics*, vol. 19, no. 21, p. 476, 2018.

[17] C.-S. Wang, "Detecting Potential Adverse Drug Reactions Using a Deep Neural Network Model," *Journal of Medical Internet Research*, 6 February 2019.

[18] "Anatomical Therapeutic Chemical Classification," [Online]. Available: https://www.who.int/medicines/regulation/medicines-safety/toolkit_atc/en/.

[19] [Online]. Available: http://sideeffects.embl.de.

[20] M. Abadi, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[21] G. Landrum, "RDKit: Open-source cheminformatics," [Online]. Available: http://www.rdkit.org.

[22] "Drugbank," [Online]. Available: https://www.drugbank.ca.

[23] "The LINCS Consortium," [Online]. Available: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE70138.