Multilabel Classification of Drugs Based on Mechanism of Action (MoA) Detection

Md Muhtasim Billah, Md Mahedi Hasan

Abstract

Traditional drug discovery method mostly took inspiration from natural products and these drugs were very often put into clinical use long before their pharmacological behaviors were fully understood. But with advanced technological tools at hand, more targeted models are used for drug discovery that utilizes the underlying biological mechanisms of a disease. The idea is to indentify a protein target associated with a disease and develop a molecule or compound that can regulate that protein target. The biological activity of such mocecules are scientifically termed as meahcanism-of-action (MoA). Connectivity Map, a joint project led by two research groups from Harvard and MIT, provides a dataset that contain 5000 potential drugs to detect and predict their MoA. A Kaggle competetion was luanched regarding this ineteresting problem associated with this dataset which has been selected for this project. Exploratory data analysis (EDA) was performed to find interesting insights about this dataset. Both supervised and unsupervised machine learning algorithms were implemented to extract important features as well as minimize the cross entropy loss by tuning the hyperparameters. From the overall analysis, the most important and interesting findings have been reported in this article to serve as the final report for the project.

Keywords: deep learning, drug discovery, mechanism of action, exploratory data analysis

1. Introduction

Historically, drugs have been derived either from natural products or were inspired by traditional remedies. Many common drugs such as acetaminophen were approved for clinical use decades before the biological mechanisms driving their pharmacological activities were understood. Now-a-days, having more powerful technologies at our disposal, drug discovery has improved from its serendipitous approach to a more deterministic, model based nature, relying on the understanding of the underlying biological procedure of a disease. In this new format, the researchers aim to identify a protein target associated with a disease and develop a molecule that can modulate that protein target. The scientific jargon for such biological activities is mechanism-of-action or MoA in short. If the MoA of a specific drug is known, it can then be related to a certain disease and be considered to have potential for treating that disease. The Connectivity Map [1], a project lead by the Broad Institute of MIT and Harvard, the Laboratory for Innovation Science at Harvard (LISH) [2] and, the NIH Common Funds Library of Integrated Network-Based Cellular Signatures (LINCS) [3], works on advancing the drug discovery and development through improvements to MoA prediction algorithms. A Kaggle competetion was luanched aiming at developing and improving such algorithms based on a unique dataset made available by the Connectivity Map project.

There are several approaches to determine the MoAs of a new drug. One widely used approach starts with treating a sample of human cells with the new drug. The cellular responses initiated by this drug administration can then be analyzed with algorithms that search for a similarity to known patterns in large genomic databases. These databases serve as libraries of gene expression or cell viability patterns of drugs with known MoAs. The dataset provided by the Connectivity Map project combines both gene expression and cell viability data. A new technology was used that measures simultaneously (within the same cell samples) human cells' responses to drugs in a pool of 100 different cell types (thus solving the problem of identifying which cell types are better suited for a given drug). The data were collected through a cell-based

assay designed to capture gene expression and cell viability levels. An assay is an investigational tool used in biology and chemistry for the detection or measurement of a target molecule's presence or its functional activity. The measurement of gene expression was based on the L1000 assay and the measurement of cell viability was based on the PRISM assay.

There are 772 gene expression features, each denoted by the `'g-'` columns. Each gene feature represents the expression of one particular gene. Therefor, there are 772 individual genes being monitored in the assay. Moreover, there are 100 different cell types on which the drugs were tested, as denoted by the `'c-'` columns. In addition, there are MoA annotations for more than 5,000 drugs in this dataset. For emoloying machine learning algorithms, the dataset has been split into testing and training subsets. Hence, the ultimate task is to use the training dataset to develop an algorithm that can accurately label each case in the test set as one or more MoA classes. Some rows can be associated with multiple targets which makes this a multi-label classification problem (as opposed to a multi-class classification problem).

There are numerous machine learning algorithms to choose from that can satisfy as a classifier with reasonable accuracy as well as tackle a multi-label classification problem. Naive Bayes and k-nearest neighbors (KNN) are two such algorithms that are simpler to implement but can raise concerns whenever there is a scalability issue. These two algorithms perform significantly better, even with relatively smaller dataset, for cases where the number of features are relatively smaller. These two models were initially considered to be starter algorithms, using all the available features as well as using the top features extracted from a dimensionality reduction algorithm called principal component analysis (PCA). However, the accuracy for these algorithms suffered terribly and thus other options were explored. The primary emphasis was put on building a deep neural network and tune its hyperparameters to improve the model accuracy by minimizing the cross entropy loss of the network.

2. Dataset

The dataset for this project has been collected from an ongoing (ended November 30th) competition titled as "Mechanism of Action (MoA) Prediction" on the Kaggle website. [4] The dataset is essentially divided into training and testing subsets, among others, which are kept in the following files.

i. `train_features.csv`: This file includes the features for the training set. This dataset has 23,814 unique observations and 876 columns. The first column indicates the `sig_id`, which contains the labels for the different drug treatments. The second column (`cp_type`) denotes the samples which were either treated with a compound (trt_cp) or with a control perturbation (ctl_vehicle) where control perturbations have no MoAs. The third column (`cp_time`) indicates the three different time lengths for the treatments (24, 48 or 72 hours). The fourth column (`cp_dose`) indicates the dosage of the drugs (high/low). The features `g-0` to `g-771` signify gene expression, and `c-0` to `c-99` signify the cell viability.

ii. `train_targets_scored.csv`: This file includes the binary MoA targets that are scored (for the competetion). These are the target labels for the drugs that will be used for the training. There are 207 labels and one drug can belong to 0, 1 or multiple labels.

iii. `train_targets_nonscored.csv`: Additional (optional) binary MoA responses for the training data which contain 403 additional labels. Use of this dataset will be optional for this project.

iv. `test_features.csv`: Features for the test data. The probability of each scored MoA for each row (drug samples) in the test data has to be predicted.

v. `sample_submission.csv`: A submission file format is given for the competetion. This would be the predicted outcome for this project. The goal is to predict the probability of each drug belonging to each of the MoAs (207 targets).

For this project, the ultimate goal is to predict multiple targets of the Mechanism of Action (MoA) response(s) of different samples (`sig_id`), given various inputs such as gene expression data and cell viability data.

3. Algorithms and Implementations

3.1 Principal Component Analysis (PCA)

Machine learning algorithms can be broadly divided into two categories which are the supervised and the unsupervised learning algorithms. The major difference bewtween these two types of algorithms is that for supervised learning, dataset require to have labels corresponding to each example or observation while on the other hand, for unsupervised learning no labels are required. Though, most of the machile learning algorithms derive from the concept of supervised learning, there are a few unsupervised learning algorithms which prove to be extremely useful in certain scenarios. One of such unsupervised learning algorithms is the principal component analysis or PCA. PCA produces a low-dimensional representation of a dataset and finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated. Over the last decade, with a boom in the sophisticated data collection and processing techniques, accompanied by increaing computation capabilities, has called for dealing with enormous datasets. And often times, the dataset contain hundreds of thoudands, and even millions of features. Extracting information by analyzing such datasets can often be very difficult even with advanced computing systems. For such cases, extracting the most important features while not losing any valuable information is very crucial. Dimensionality reduction algorithms such as PCA can be very useful for such tasks by finding the top desired features which can lead to a smaller computational overhead. Apart from producing derived variables for use in supervised learning problems, PCA can also serve as a tool for data visualization.

Though the dataset used in this project comes with corresponding lables for each observations, unsupervised algorithms can be used on the features to extract the most important ones. PCA, for example, can reduce the dimensionality of the feature matrix to be used for some memory hungry algorithms such as KNN. There are 876 features in total from which 772 represent gene expressions. After performing feature scaling, PCA was performed on these 772 features to reduce the size of the feature matrix so that the extracted features can be used within the KNN and Naive Bayes algorithms. There are two ways the PCA can be implemented. The first way is to fix the number of the top k desired features that are the most important (explains the most variance). The second approach is to generate a scree plot to visualize how many components are required to explain what percentage of the total variance. However, the choice is subjective as to how many principal components should be used in the model or how much of the variance needs to be explained by these features. Conventionally, at least a variance of 80% is considered necessary to be explained by the extracted principal components and 95% is a reasonably high enough percentage to provide the best results.

3.2 Deep Neural Network (DNN)

The 21st century has been observing the rise of artificial intelligence (AI) from its very beginning and at its very core lie the subdomain called machine learning. The fundamental concepts of Machine learning are not novel and have existed in computer science for ages. There are numerous machine learning algorithms, both decades old and new, that are built upon the mathematical theories and statistical methods. While these algorithms vary in so many ways, one aspect common to all of them is the necessity for big datasets and more often than not, larger the dataset, the better the performance of these algorithms. Despite this similarity, there are a set of algorithms that can perform significantly better than others with the increasing size of the dataset and the model accuracy can improve commensurately, after some hyperparameters tuning. This set of algorithms are called deep neural networks (DNN) and popularized by the name 'deep learning'. DNN is a set of supervised machine learning algorithms and thus require labeled datasets.

The idea of DNN comes from the linear model theories in statistics that discuss ways of finding the best fit for a set of data points in a Cartesian plane. The method is called linear regression where a dependent variable (also called response and generally denoted by Y) is regressed on one or more independent variables (also called explanatory variables/predictors/features and generally denoted by X). Depending on the number of features, the linear regression can be divided into simple or multiple linear regression. The goal is to find a linear combination of the parameters so that for a set of known values of the features X, the response Y can be predicted. A special of such linear models is called the logistic regression which is useful for cases when

the response Y has binary values (0 or 1). For logistic regression, the nonlinear sigmoid (also called logistic or logit) function is used which forces the response Y to be bound between the values 0 and 1. Based on the predicted value of Y, an example/observation is classified as 0 or 1 (sometimes as -1 or 1). Thus, logistic regression can be thought of as a very simple binary classifier. A neural network, though inspired by how neurons interact with one another in human brain, is an expansion of the logistic regression framework and can tackle multi-class as well as multi-label classification problem.

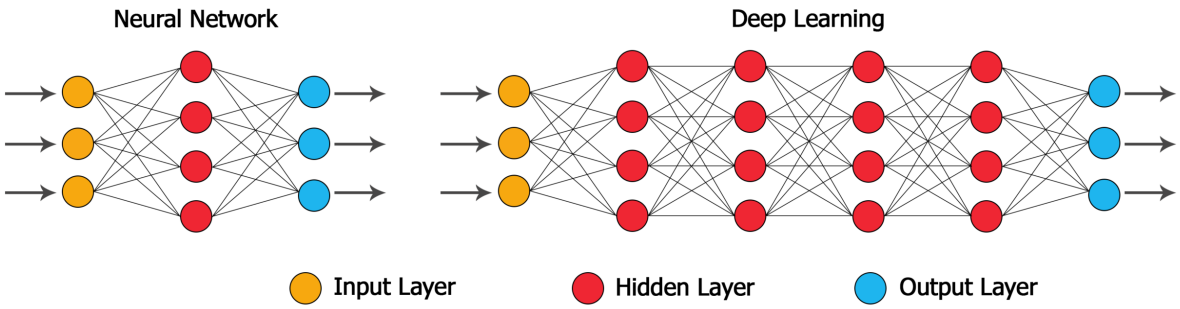

Fig 1. Shallow and deep neural network architecture. [5]

A simple neural network architecture consists of three basic layers- the input layer, the hidden layer and the output layer. When there are multiple hidden layers, the network is called a deep neural network or DNN. The simple DNN architecture is often denoted as feed forward neural network or FFNN as well. A standard DNN architecture schematic is given in Fig 1. A basic DNN architecture contains at least three layers which are the input layer, hidden layer and the output layer. In each layer, there are multiple neurons where non-linear activation functions are deployed. The neurons of the same layer are not connected with one other; however, each neuron of a layer is connected to each neuron of the following layer. Each of the neurons in the input layer is called an input feature and there are randomly initialized wights and biases associated with them. The linear combinations of these features, weights and biases are then passed through a non-linear activation function in each of the neurons of the following hidden layer. The output of the activation function in the hidden layer then works as the input features for that layer. Each of these neurons are again connected to each of the neuron of the output layer and they have randomly initialized weights and biases associated with them. Finally, the linear combination of the inputs, weights and biases are calculated, passed through the activation function and a value is found at the output layer neuron. This value is compared with the label or ground truth value and the cross-entropy loss of the network is calculated. An optimization algorithm can be used to minimize this loss by iteratively updating the weights and biases of the network. All deep neural networks at their core follow this baseline model.

The datasets for this project is ideal for making use of a deep neural network for several reasons. First of all, the dataset comes in a tidy, tabular format with approximately 23,000 examples in the training set and another 4,000 examples in the test set and there are 876 features. The dataset is labeled and divided into 207 different classes. Each of the examples has its corresponding label where it may belong to one or multiple classes or no classes at all. This makes the problem a multi-label classification problem and thus, a feed forward deep neural network can be considered to be the best choice given the above characteristics of the datasets and the problem definition. That is why, the primary focus for this project has been to implement a deep neural network.

In a deep neural network, there are both parameters and hyperparameters. Parameters are usually the wights, biases etc. that the network is trying to learn going through the iterative optmization algorithm. These parameters can again be divided into trainable and non-trainable parameters. Trainable parameters are those that are being updated after each iteration of the optimization step. Finding those parameter values for which the network experiences the minimum cross-entropy loss is the ultimate goal of the neural network. However, there are some other non-trainable parameters that are not updated at each iteration step of the optimization algorithm, rather they are usually associated with one of the layers of the network. Hyperparameters are those parameters, that determine the overall learning process of the network and

influence the convergence speed and accuracy. For any deep neural network architecture to perform well, it is absolutely essential to tune its hyperparamters. For finding the best model that gives the highest accuracy and the lowest cross-entropy loss, different values of the hyperparameters have bee tried and some of these hyperparameters and model characteristics are provided in Table 1.

Hidden Layers	3 (L1, L2, L3)
Hidden Layer Neurons	L1: 1024 , L2: 1024, L3: 512
Kernel Initializer	HeNormal, LecunNormal
Minibatch Size	64, 128
Regularization	Dropout, AlphaDropout
Activations	ReLU, SELU, Sigmoid
Cost Function	BinaryCrossEntropy (with label smoothing)
Optimization	Adam, AdamW
Learning Rate	Adapted (ReduceLROnPlateau)
Resampling Method	Multilabel-StratifiedKFold

Table 1. Deep neural network characteristics and hyperparameters

Several step by step procedures were followed before training the model. The first step is the data preprocessing where the dataset is prepared before fitting into the model. Among 876 features in the training and testing set, the first feature is the `sig_id` which denotes the ID for each drug sample. This feature is not required and thus dropped from the dataframe. There are three categorical variables `cp_type`, `cp_time` and `cp_dose` which denote the treatment type (control/treatment), time length and the dosage amount respectively. Since 92% of the drug samples received a treatment, the `cp_type` variable was dropped as well. The `cp_time` and `cp_dose` variables were mapped into dummy variables as {24: 0, 48: 1, 72: 2} and {'D1': 0, 'D2': 1} respectively. This preprocessing step was done for both the training and testing subsets. Then each of the remaining 874 features of the dataframe were scaled between 0 and 1. This procedure is called input feature scaling and helps the optimization algorithm to perform better.

Once the data preprocessing and the input feature scaling is done, then the model characteristics have to be determined to build the model. There are 874 input features which means there are 874 neurons in the input layer. The examples were divided into mini batches with a batch size of 64 or 128. Batch normalization was performed on the input features. Regularization is another important aspect of a DNN for avoiding the risk of overfitting the model on the train set. 20% dropout regularization was performed on the input layer which indicates that 20% of the input features were randomly dropped in each iteration step of optimization. For the first hidden layer (L1), there were 1024 neurons with the ReLU (rectified linear unit) activation function. That means that all the linear combinations of inputs, weights and biases were passed through the ReLU activation and they will work as the input for the first hidden layer L1. Similar to input layer, batch normalization was done for the neurons of L1 layer and the associated weights for L1 were normalized as well using the 'HeNormal' kernel initializer. 20% dropout regularization were applied for L1. The second hidden layer L2 was implemented in a the same way as L1.

For the third hidden layer, L3, there were 512 neurons and the activation function used was SELU (scaled exponential linear unit). Batch normalization was performed as usual and 20% alpha-dropout regularization was applied for this layer. The associated weights were normalized with the 'LecunNormal' kernel initializer. For the output layer, there were 207 neurons since the number of target classes in the dataset are 207. Batch normalization was performed like before but no regularization was required for the output layer. However, the associated weights were normalized using 'HeNormal' kernel initializer and the activation function used was sigmoid since this is a multi-label classification problem. The Adam optimization algorithm with weight decay (AdamW) was used to train the model where the following paramters: learning rate, `lr = 1e-3`, `weight_decay = 1e-5` and `clipvalue = 756`. To determine th model efficacy, binary crossentropy loss of the network was taken as the evaluation metric with a label smoothing rate of `1e-15`.

To cross validate the possible models, a resampling method called MultiLabel-StratifiedKFold was used. This method is an expansion from the stratified k-fold validation method. The entire dataset (both training

and testing) was split into 7-folds to perform cross validation. Seven seed values were chosen and for each seed, for each fold within a seed, the model was trained for 60 epochs. Several callback methods were used to enhance the model perfomance. The first one is the adaptive learning rate called the `ReduceLROnPlateau` whose task is to reduce the learning rate when an evaluation metric (for this case, the validation loss) has stopped improving. The factor by which the learning rate was set to be reduced was `0.2`, the patience or the number of epochs with no improvement (after which learning rate will be reduced) was set to `4` and the lower bound on the learning rate was set as `1e-6`. `ModelCheckpoint` callback approach was used to to save the model or model paramters (weights) by monitoring the validation loss after each epoch. The third callback is the `EarlyStopping` method which stop training the model (thus, reduced the iterations/epochs) when a monitored metric (the validation loss in this case) has stopped improving. The patience used for early stopping was `10` epochs. The model validation loss was tracked through each epoch to monitor the model performance.

4. Results and Discussions

4.1 Exploratory Data Analysis (EDA)

4.1.1 Categorical Features

Before any statistical analysis or machine learning algorithms are applied on a dataset, acquiring important insights about the data is customary. This is becuase having a proper idea about the dataset can steer not only towards the data wrangling approaches but also towards the selection of the model. This procedure is commonly termed as exploratory data analysis or EDA. EDA differs from the traditional confirmatory data analysis or CDA due to the fact that there are no preexisting ideas about the dataset before EDA. Due to its proven benefits, exploratory data analysis has been performed on the MoA dataset to find interesting characteristics that would be helpful for data wrangling and bulding the model.

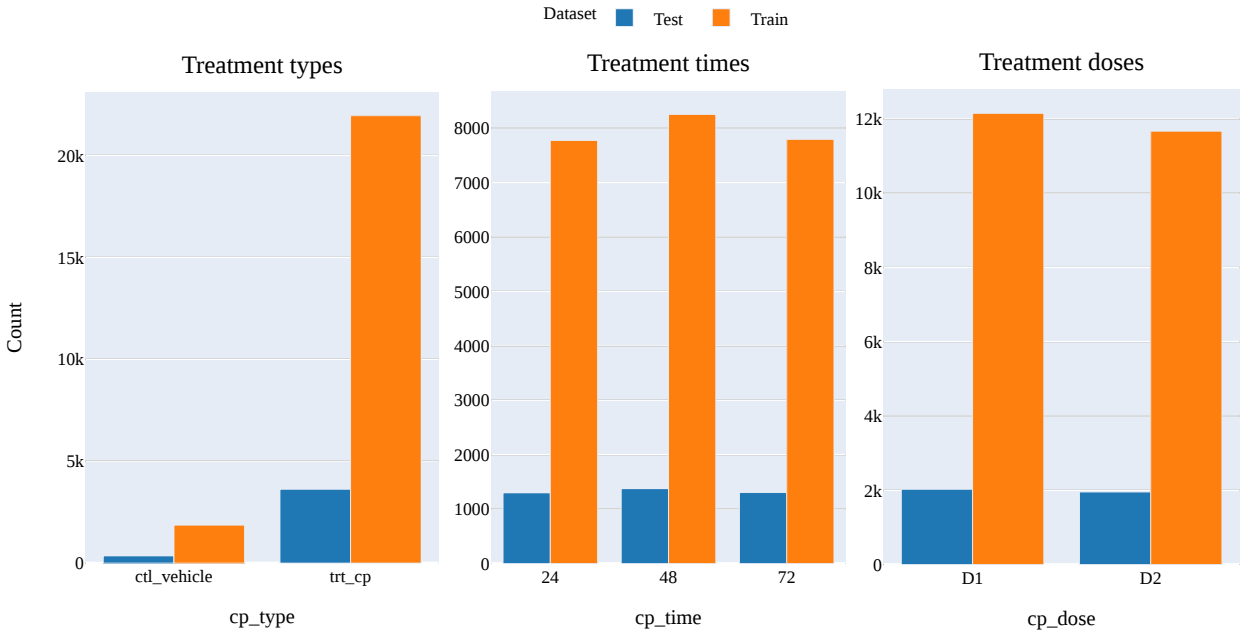

Fig 2. Frequency counts for all the categorical variables.

As mentioned earlier, there are three categorical features in the training and testing subsets which are the `cp_type`, `cp_time` and `cp_dose`. Fig 2 shows the frequncy of these categorical variables in both training and testing datasets. The feature `cp_type` indicates whether the drug samples were treated with a compound

(trt_cp) or with a control perturbation with no MoAs (ctl_vehicle). As seen from the left panel, almost all of the samples (92%) were treated with a compund. The remaining 8% didn't have any MoA associated with them. This information is useful because it might be considered to drop these 8% of the drug samples since they won't help train the algorithm to detect MoA for new drugs. Again, the drug samples were treated with a compound for three different timespans- 24 hours, 48 hours and 72 hours which are indicated by the cp_time feature. The panel in the middle shows that the timespans are almost evenly distributed both in the training and testing subsets. This is also an important information for implementing machine learning algorithms because the training and the validation set have to be from the same distribution or the learning process would be at jeopardy. The feature cp_dose indicates the dosage amount of the drug samples which can be either high (D1) or low (D2). The feature values D1 and D2 are almost evenly distributed among the training and testing sets as shown by the rightmost panel in Fig 2.

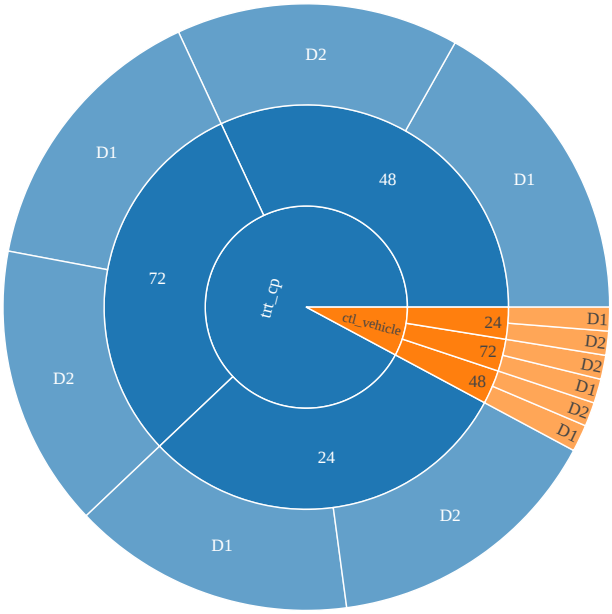

Fig 3. Sunburst chart for all the categorical variables in training dataset.

The information presented by Fig 2. can be summarized in one single plot in the form of a sunburst chart which is given as Fig 3 (only for training dataset). A sunburst chart is an excellent way to plot categorical features that basically shows the percentage of the levels of each categorical feature. One feature level can then again be subdivided into other feature levels and their percentage can be shown as an inward-to-outward manner. For example, in Fig 3, the innermost circle represents the cp_type feature and divides it into two levels- trt_cp (92%) and ctl_vehicle (8%). The circle in the middle shows the levels of time length in hours (24, 48, 72) for the feaure cp_time. This is shown for both levels of the feature cp_type and it's evident that the timespans are almost evenly distributed. The outermost circle indicates the feature cp_dose and its two levels- high (D1) and low (D2). The doses are equally divided among all levels of the other categorical features. The results provided by the sunburst chart is in line with the barplots given in Fig 2. A sunburst chart for the testing dataset will have similar ditribution for the levels of these categorical features.

4.1.2 Cell Viability Features

In the datasets, 100 different cell types were analyzed for MoA and their cell viability features are indicated by the columns c-0 to c-99. Drug samples were experimented on different cell samples to see their adaptibility as well as the impact on different tissues. Since the machine learning model would be trained on all the cell viability features alongside the gene expression features, it might be interesting to check for possible outliers or highly correlated features among the cells. The Pearson correlation coefficient, denoted by r,

is one of the most commonly used metric to determine the strength of correlation between two variables. The value of r ranges between $[-1, 1]$. A positive r value indicates there is a positive relationship between the variables and for negative, it's the opposite. Usually, a r value larger than 0.8 is considered as high correlation. For many statistical models, the multicollinearity issue is a major concern but for most of the machine learning models, it's not an issue since the main purpose is to make prediction on the unseen data. Thus, for largely data-driven algorithmic approaches, the correlation issue is rarely a big concern. Yet, very often it's interesting to investigate the correlation among features since it helps explain model behavior and performance.

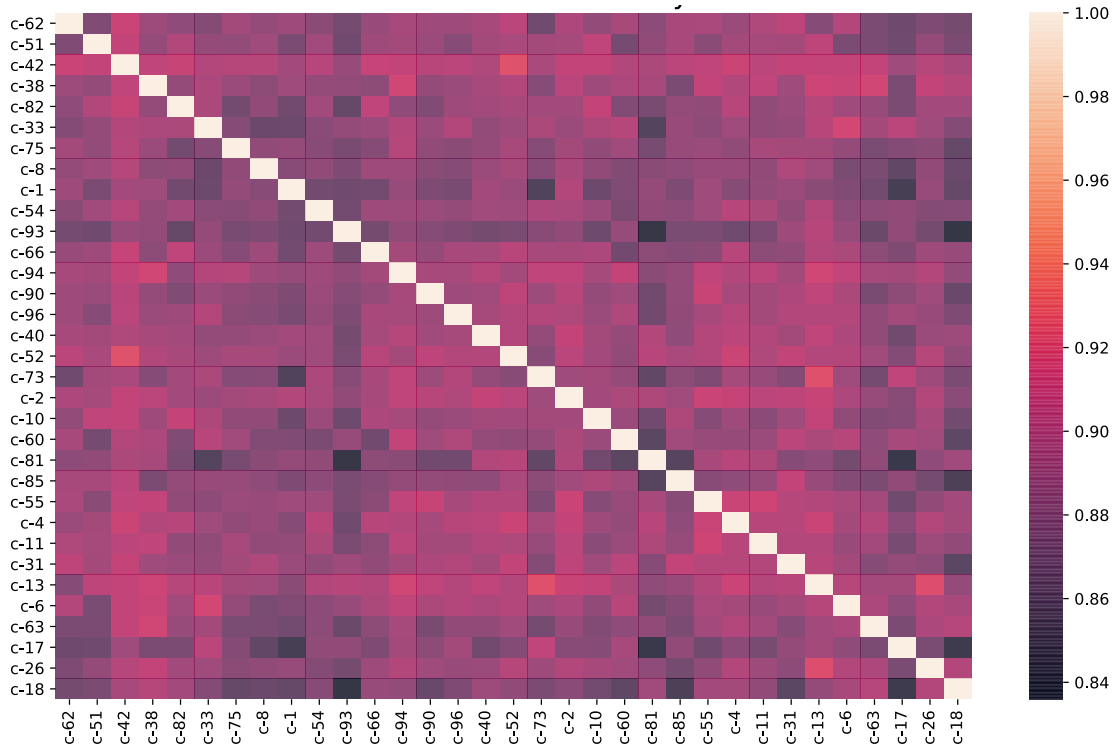

Fig 4. Top correlated cell viability features for Pearson correlation coefficient, $r > 0.8$.

For the training set, the Pearson correlation coefficient r has been calculated for each feature pair taking all the features into account. The top features that have a r value larger than 0.8 have been extracted and plotted in a heatmap which is given in Fig 4. As indicated by the colorbar, lighter the color, higher the value of r. There are top 33 correlated features shown in this heatmap which are highly correlated with the other features. This means that a third of all the cell viability fetures have high correlations among them should be dealt with if necessary for any statistical analysis.

4.1.3 MoA Targets

The binary MoA targets are provided by the file named `train_targets_scored.csv`. The columns serve as the target labels for the drug samples that have been used for the training. There are 207 labels and one drug can belong to 0, 1 or multiple labels. The column names are representative of different mechanism of actions and they apparently contain the biomoecule name and its type (based on the nature of it use). The name of the biochemical family they belong to are seemed to be at the end of the column name string literals. The column names were preprocessed to determine the types of families available among the targets. Six different biomolecule families were most frequently found which are - `inhibitor, antagonist, agonist, activator, agent and blocker` - for which the frequency count percentages are provided in Fig 5. From

Fig 5, it seems that most of the targets belong to a certain MoA type called `inhibitor` which accounts for more than half of all the MoAs. The second most frequent MoA type is `antagonist` which covers roughly above 15% of all the MoA targets. The other four categories account for the remaining 25% of the MoA targets. However, there are targets which don't belong to these six major groups. A few such examples are `dieuretic, steroid and laxative` etc.

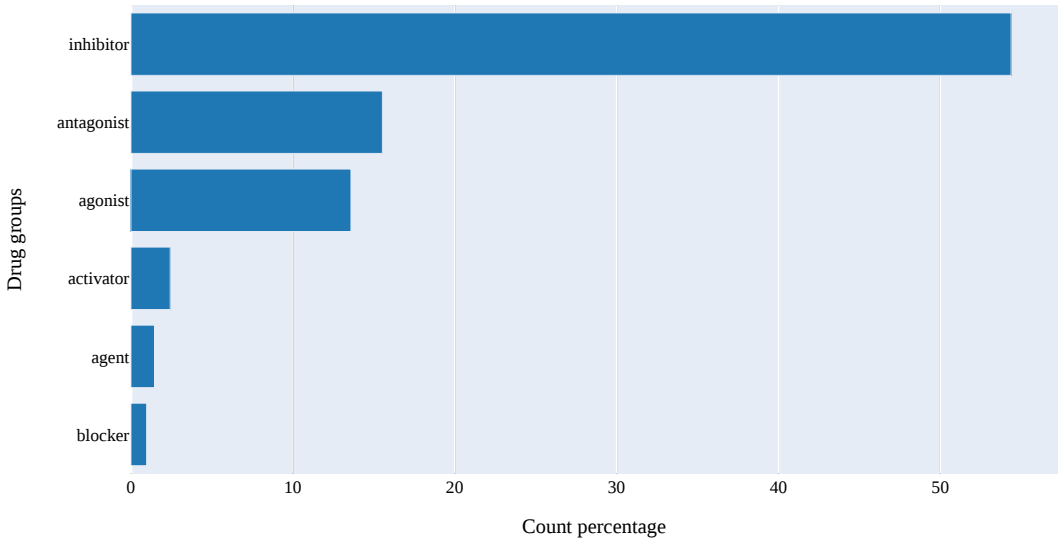

Fig 5. Groups of drugs in the target columns.

Most of the drug samples in the training set belong to at least one or multiple of these 207 MoA targets, although there are a significant number of samples that are not associated with any of these targets. It's worth investigating to find the top MoA targets that are associated with the highest number of drug samples in the trainset. In other words, the drug samples in the trainset can most frequently be related to these top MoA targets. In a similar but opposite manner, the bottom 25 MoA targets can be found as well which indicate that very few drug samples could be associated with these targets. These top and bottom 25 MoA targets are provided in Fig 6.

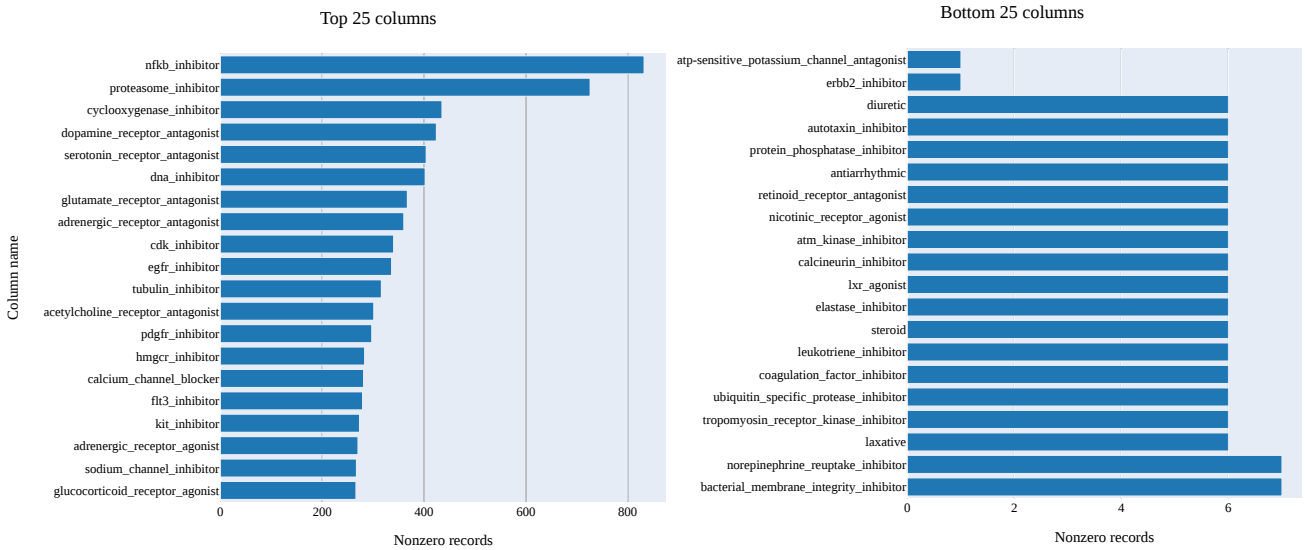

Fig 6. Target columns with the highest and lowest number of activations.

In the left panel of Fig 6, the top 25 MoA target have been reported. It seems that more than 800 examples in the trainset are associated with the MoA called `nfkb_inhibitor`. It's also noticeable that the top three targets belong to the `inhibitor` group. This is expected since more than 50% of the targets belong to this group of MoA. On the right panel of Fig 6, the bottom 25 MoA targets are reported. The target columns named `atp_sensitive_potassium_channel_antagonist` and `erbb2_inhibitor` sits at the bottom of the list and they both have one non zero entry. This means that each of the MoA targets has at least one non-zero entry and is associated with at least one drug sample in the trainset.

The dataset for this project pose a multi-label classification problem. This means that each drug sample in the training subset can belong to multiple MoA targets/classes. It might be interesting to find the frequency of drug samples belonging to different number of classes or labels, starting from zero and with an increment of one. Filtering the `train_targets_scored.csv` dataset, these frequencies have been calculated and reported in Fig 7. On the left panel, the frequency counts are shown in a bar chart. On the right panel, the percentage of these frequencies are presented in a pie-chart.

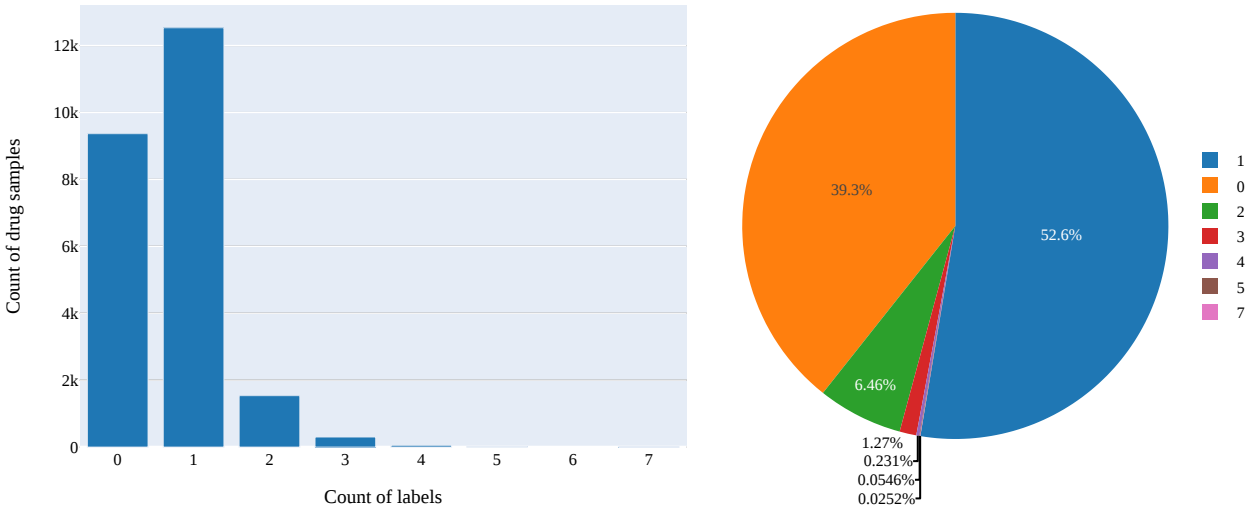

Fig 7. Distribution of number of activated target labels for all drug samples

Fig 7 indicates that above 12,000 drug samples, which is more than half of the trainset, belong to only one class. In other words, they are associated with only one label and account for 52.6% of the entire trainset, as seen from the pie chart in right panel. However, there are more than 9,000 drug samples who don't belong to any of the target classes which account for 39.3% of the trainset data. There are a few drug samples that belong to multiple classes (2, 3 or 4) which account for roughly 8% of the training data. There is no drug sample that belongs to more than 4 target classes as seen from the barplot and piechart.

4.2 Model Analysis

4.2.1 Dimensionality Reduction

Principal component analysis or PCA has been performed for dimensionality reduction of the gene expression features matrix. There are 772 gene features which are first scaled to the range $[0, 1]$. Once the preprocessing is done, the feature matrix was fitted within the PCA model to extract the top features. These top features are often called principal components. The number of components to be chosen can be determined by the individual. However, the recommended approach is to generate a cumulative variance plot against the number of selected principal components to visualize the PCA performance on the feature matrix. Such a plot has been provided in Fig 8.

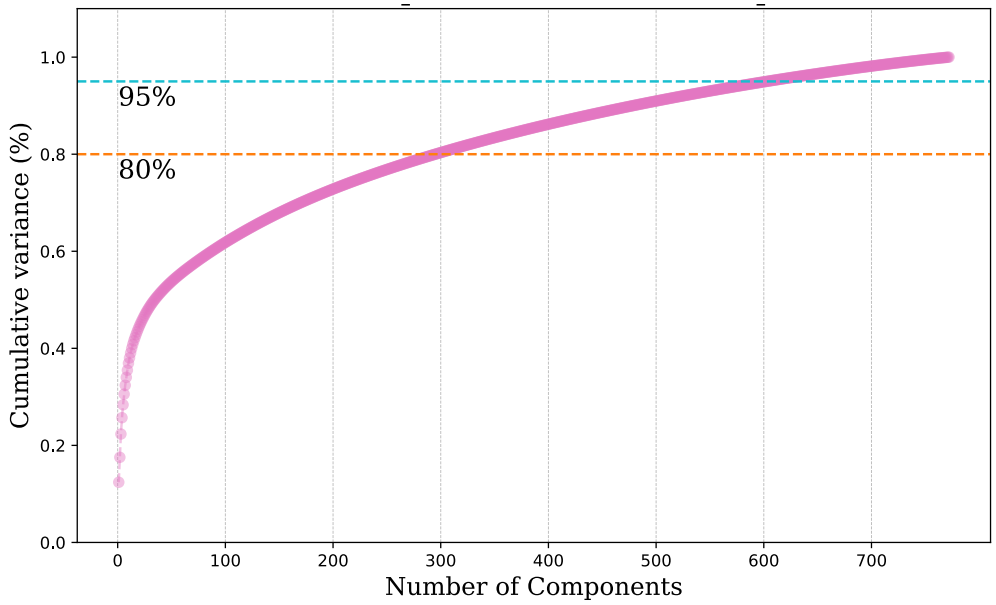

Fig 8. Top principal components determined by PCA.

In Fig 8, the x axis denotes the number of principal components required to explain the cumulative variance percentage denoted by the y axis. This plot is extremely helpful to determine the threshold for the cumulative variance and the number of principal components. Conventionally, at least a cumulative variance of 80% is recommended and 90% is preferred. In Fig 8, the pink line shows the gradual rise in the explained cumulative variance with the increasing number of principal components. The olive dashed, horizontal line indicates that to achieve 80% cumulative variance, top 300 principal components should be taken into account. For 90%, the required number of principal components would be 600 which is almost doubled. There is a trade off to be made between the explained variance and the number of features which, though mostly depends on the choice of the supervised learning model where these features will be used, can often be subjective to an individual's preference.

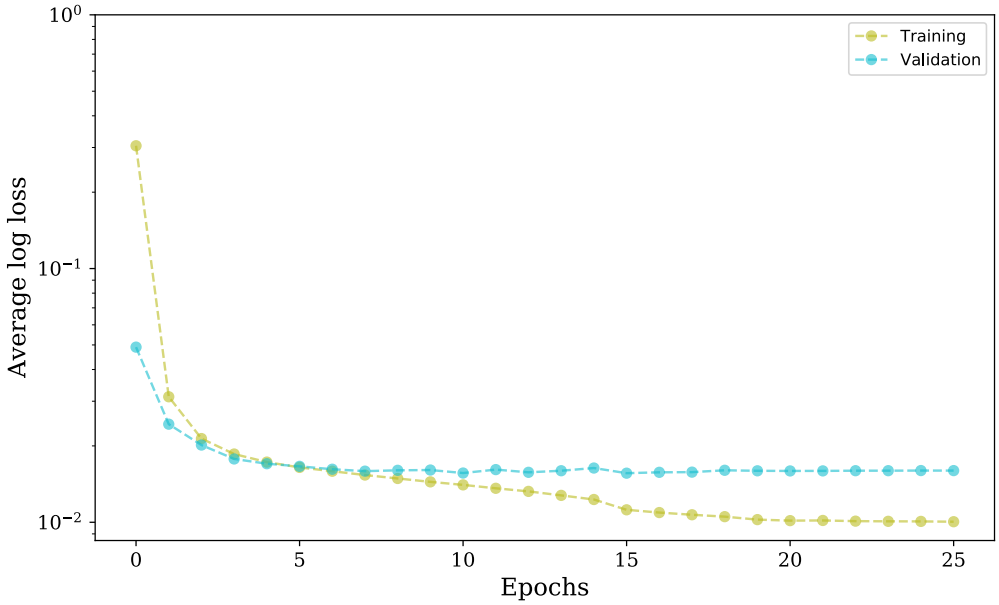

Fig 9. Average log loss across the neural network architecture for the training and testing stage.

4.2.2 DNN Training and Testing

A feed forward deep neural network was implemented for training the dataset on roughly 23,000 examples. Since the trainset is adequately large and the data is available in a tidy, tabular format, neural network algorithm has been preferred over other classifiers. Since one training example can belong to multiple classes, multi-label stratified k-fold with 7 folds has been used as the resampling method for training the algorithm. The model hyperparameters has been tuned multiple times to acieve the best accuracy and minimize the binary cross entropy loss. AdamW (weighted Adam) optimization algorithm used to find the model parameters (the weights and biases). The average log loss has been considered as the evaluation metric for both training and testing and has been plotted against the number of epochs to see how the model is performed. The plot is provided in Fig 9.

The number of epochs for each fold was set to 60 initially, however, since early stopping was incorporated in the model, only 40 epochs are shown on the x axis in Fig 9. The average log loss is shown in log scale on the y axis. It's found that for training, the cross entropy loss starts below 1 and then gradually reduces after each epoch until it reaches the plateau after 25 epochs. The testing loss starts even at a smaller value than the training stage and then plateaus sometime after 15 epochs. However, when the loss reaches the plateau for both the training and testing stage, it seems that the testing loss is slightly higher than the training loss. This indicates that there might be an issue of overfitting the trainset which leads to this small variance.

5. Conclusion

The unprecedented advancement in the field of aritificial intelligence has influenced how scientific research is being carried out in numerous fields. Its impact is noticeable in the pharmacology as well since it is replacing the age old drug discovery approaches. The data derived from one such advanced technique has been worked with for this project which paves the way for classifying new drugs by associating them with the known biological acitivities called the mechanism of action (MoA). The dataset contained both numerical and categorical variables for which interesting insights have been discovered by going through the exploratory data analysis. The data exploration proved to be useful not only for gathering in depth ideas about the features but also for formulating machine learning models suitable for making prediction. PCA, an unsupervised machine learning algorithm, has been used to extract the most important cell viability features. Also, feed forward deep neural network, a popular supervised machine learning algorithm, has been implemented for training and testing the dataset. The cross entropy loss for the model has been reported which indicated towards the model's efficacy for making prediction on new drugs to detect their MoA. The potential new drugs can be used for treating diseases with their MoA being understood. However, a better model can be engineered to work with such datasets by exploring other simpler or more complex multi-label classification algorithms as well as more rigorous tuning of the hyperparameters which can be an extension of the work presented in this project.

Bibliography

[1] The Connectivity Map Project, Broad Institute. https://clue.io
[2] Laboratory for Innovation Science at Harvard (LISH). https://lish.harvard.edu
[3] NIH Common Funds Library of Integrated Network-Based Cellular Signatures (LINCS). https://lincsproject.org
[4] Mechanism of Action Detection, Kaggle. https://www.kaggle.com/c/lish-moa/data
[5] https://alphabold.com/neural-networks-and-deep-learning-an-overview/

Appendix

All the analyses presented in the report has been done in Python. The code has been uploaded to GitHub which can be accessed through the following link. https://github.com/mmbillah/CPTS575_DataScience_Project