# Data Mining Coursework 2021

# CS4850 Data Mining

## Contents

Abstract -------------------------------------------------------------------------------------------------------------------- 2

Introduction --------------------------------------------------------------------------------------------------------------- 3

Exploratory Data Analysis ---------------------------------------------------------------------------------------------- 3

Data Pre-processing ----------------------------------------------------------------------------------------------------- 9

Feature Selection methods --------------------------------------------------------------------------------------------- 10

Machine Learning models ---------------------------------------------------------------------------------------------- 13

Conclusion --------------------------------------------------------------------------------------------------------------- 15

## Abstract

This report provides a comprehensive data mining solution to predicting the properties of protein sequences and their ability to provoke an immune response. The data was provided by IEDB and GenBank. The goal of this data mining was to develop a classification model to predict classes of new observations if Class will predict -1 or 1.

Two datasets were provided: a training dataset to train the models and a test dataset to predict classes. The training dataset contained 4907 rows and 451 columns and the label column had class imbalance. The volume of columns meant high dimensionality for machine learning model training. The data was then profiled with pandas\_profiling to gain insights into each column and most columns had a fairly bell-shaped distribution.

Due to the size of the data, it was divided into nine subgroups with each sub-data frame to allow for data exploration. From the exploration, there were no missing values and no duplicate rows and most of the columns had a normal distribution. From the 451 columns, 327 were of float data type, 120 columns of int64 data type, and 4 columns of string data type. Out of 450 feature columns, 95 were chosen after the preliminary Exploratory Data Analysis (EDA) because they had fewer zeros which are noises.

Outliers were treated using the .clip data frame property of keep all values between the 5th percentile and 95th percentile. In addition, columns with higher values were logged before scaling all columns. PCA was used to reduce the dimensions and number of components chosen based on the output of the scree diagram. XGBoost algorithm appeared to do give the best classification report with an accuracy of 90.84% and a Mathews Correlation Coefficient (MCC) of 77 %.

## Introduction

This report presents a fully developed data mining solution and the steps taken to arrive at the various conclusion and elaboration on the dataset. The aim is to develop a classification model to predict the class of new, previously unseen observations with the best possible classification performance. Two datasets were provided by IEDB and GenBank for the and task: a “training” dataset called CW\_data.csv for model development and performance assessment, and a “test” dataset called CW\_test.csv to make predictions on a new data observations. All observations in the data are related to linear B-cell epitopes, which are protein regions of interest for a variety of applications in immunology. The goal at hand is to predict as accurately as possible, whether or not a given observation is a positive epitope (-1) or a negative epitope (1). This will be useful in diagnosing protein epitopes.

## Exploratory Data analysis

The data exploration process was conducted using python and python libraries and jupyter notebook. The dataset was uploaded in to the notebook using the libraries. Due to the number of columns, the pandas data profiling was used to profile the data. Initial Data exploration revealed the following:

* There were 4907 total number of observations
* There were 450 feature columns and 1 label column
* There were no missing values
* There was class imbalance in the label column: of the 4907 records, 3419 had -1 label classes representing 69.66% and 1489 rows had 1 label class representing 30.34% of total rows and this shows a class imbalance
* Most of the numerical columns have a normal distribution
* 4 columns were of string data type
* 119 columns were of int64 data type
* 327 columns were of float data type

## **Data visualization and data profiling**

To get a vivid look at the data, the pandas profiling library was used to get insight into the data, summary statistics and distribution of the data. Due to the size of the columns, the figures below are samples of the results from the data profiling results and the complete profiling.

The profiling gives a summary statistics of each column by describing the number of distinct values, percentage of distinct values, missing values, the percentage of distinct values, mean, maximum, minimum, zeros, percentage of zeros and the size of memory each column occupies, etc.

Chart, histogram

Description automatically generated

Fig 1: *Data profiling of feat\_Perc\_A column. From the profiling, the column has 9 distinct values and has no missing values. From the distribution, the mean in 0.108, the minimum is 0 and the maximum is 0.533*

Chart, histogram

Description automatically generated

Fig 2: *Data profiling of feat\_Perc\_K column. The column has 7 distinct values, has no missing values and zeros make 21.1% of the total rows. The column has a mean of 0.106, a minimum of 0 and maximum of 0.4*

Chart, timeline

Description automatically generatedChart, funnel chart

Description automatically generated

Fig 4: *Plot of feat\_Perc\_R column. This has 5 distinct values, no missing values and dominated by zeros.*

Fig 3: *Plot of feat\_Perc\_I column. This has 5 distinct values, no missing values and dominated by zeros.*

## **Data decomposition**

The data had 450 feature columns which made it daunting to analyze each individual column, but to allow for easier analysis, the data was further decomposed into nine dataframes with each dataframe having about 50 columns which excludes the following columns: Info\_PepID, Info\_protein\_id, Info\_center\_pos, Info\_AA, and Info\_window\_seq. The decomposition was as follows in order of appearance:

* cw\_1 (49 columns): from feat\_seq\_entropy to feat\_VHSE4
* cw\_2 (50 columns): from feat\_VHSE5 to feat\_CT022
* cw\_3 (50 columns): from feat\_CT023 to feat\_CT123
* cw\_4 (50 columns): from feat\_CT124 to feat\_CT224
* cw\_5 (50 columns): from feat\_CT225 to feat\_CT325
* cw\_6 (50 columns): from feat\_CT326 to feat\_CT426
* cw\_7 (50 columns): from feat\_CT430 to feat\_CT530
* cw\_8 (50 columns): from feat\_CT531 to feat\_CT631
* cw\_9 (46 columns): from feat\_CT632 to feat\_Perc\_Y

## **Label**

The label in the data is the Class column which indicates whether a protein sequence returns -1 or 1. A bar plot was used to visualize the column which indicate a class imbalance. In all, of the 3419 protein sequences were -1 representing 69.66% 1489 were 1representing 30.34% of total rows protein sequence.

Bar chart

Description automatically generated

Fig 5: *A bar plot of the label classes. This represents a class imbalance with -1 having close to 70% of rows and 1 having about 30%*

## **Outliers**

From the preliminary exploratory data analysis, more than half of the data had outliers. The outliers were detected by using the box plot from the seaborn library. Below are samples of outliers detected in some columns. All outliers are presented in the Jupyter notebook.

Chart, box and whisker chart

Description automatically generated

Fig 6*: feat\_seq\_entropy demonstrates outliers after the 75th percentile. The box and whisker plot considers values the fall between the 25th and the 75th percentile and any value outside this boundary are considered outliers*

Chart, box and whisker chart

Description automatically generated

Fig 7: From the box plot in the seaborn library, feat\_KF8 has outliers both below the 25th and 75th percentiles.

## Data Pre-processing

To prepare the data for the machine learning process, the following steps were taken:

## **Outliers**

From the preliminary exploratory data analysis, more than half of the data had outliers. The outliers were detected by using the box plot from the seaborn library. From the detected outliers, the .clip pandas dataframe property was used to set all values below the 5th percentile to the 5th percentile and all values above the 95th to the 95th percentile.

## **Columns selection**

After the preliminary EDA, all columns with zeros were dropped leaving 95 columns for the data preparation process. The columns with zeros were dropped because they will introduce noise into the machine learning process.

## **Column transformation**

feat\_C\_atoms, feat\_H\_atoms, feat\_N\_atoms, feat\_O\_atoms, feat\_molecular\_weight columns had larger values which are likely to have influence the model, to allow for fairness, the columns were logged using NumPy.

## Feature selection methods

The aim of performing feature selection was to limit model training to columns that are relevant and will help to better separate the classes. Some of the methods used were:

## Principal Component Analysis (PCA)

Principal Component Analysis is a technique for reducing dimensionality and yield a better training results. PCA relies on the number of components of the training set to reduce dimensionality. To determine the number of components to use, a scree plot was used.

According to Cattell (1966), the scree plot focuses on the point where there is a sharp decline in size of the eigenvalues. When the eigenvalues drop dramatically in size, an additional factor would add relatively little to the information already extracted. From the graph below, the graph sharply declines where x = 18 which informs the number of components for the PCA to be 18. Fig 2 shows the scree diagram.

Chart

Description automatically generated

Fig 2: *A scree plot to determine the optimal number of component to pass as argument for the dimensionality reduction with the PCA. From the plot, there is a sharp reduction from where the x-axis is 18. This means 18 is the optimal parameter*.

## Extra Trees Classifier

The sklearn library in python has extra trees classifier for feature selection. This algorithm uses statistical methods to arrive at the features with the specified number of expected features for the model training. From the algorithm, the following columns emerged most important features in descending order of importance as shown in fig 3:

feat\_ProtFP3, feat\_Perc\_N, feat\_F1, feat\_Perc\_Tiny, feat\_F2, feat\_VHSE1, feat\_ProtFP2, feat\_BLOSUM8, feat\_MSWHIM3, feat\_VHSE7, feat\_Perc\_T, feat\_KF4, feat\_MSWHIM2, feat\_Perc\_Acidic, feat\_Perc\_Charged, feat\_Perc\_E, feat\_T2

Chart, bar chart

Description automatically generatedFig 3: *Results from the ExtraTreesClassifier from the scikit-learn library. The classifier returns the most important features and their level of influence in descending order of importance*. *From the plot, feat\_T2 and feat\_Perc\_E are really important in the model training.*

## Random Forest

During the model training, a list of optimal features were presented to the Random Forest Classifier to choose the maximum features necessary for the model training, out of the list, 18 was chosen as the maximum features needed for the machine learning which confirms the results from the scree diagram.

## Machine Learning models

The problem we’re faced with is a classification model. Three classification algorithms were sampled and compared to choose the one that gives the best result. The optimal choice of model will be based on the results of the Mathews Correlation Coefficient (MCC). 70 percent of the dataset was used for training and 30 percent for validation. The training dataset used for the training was the results from the PCA. The models explored were:

## XGBoost Algorithm

Gradient boost is an implementation of gradient boosted decision tree designed for speed and performance in machine learning and greatly helps in regularization. Model parameters were set of which n\_estimators was set to 1200 and max\_depth to 6. This resulted in an Accuracy score of 90.7 percent and an MCC score of 77 percent.

## Logistic Regression

There was no hyperparameter tuning for the Logistics regression, instead, a Grid Search Cross Validation was performed to determine the optimal regularization parameter for the model. The Grid Search was done using the model selection form the scikit-lean library using the inside fold and number of splits to be 10. To handle the class imbalance, class\_weight was set to “balanced”.

A list of regularization parameters were passed to the Grid Search to select the optimal. The parameters passed were [0.1, 1, 10, 100] and after the Grid Search the optimal parameter was 10. This produced an accuracy of 77.60% and an MCC score of 51%

## Random Forest

Following the same procedure for Logistics Regression, no hyperparameter tuning was performed but a Grid Search Cross Validation was performed to determine the optimal max\_features and min\_samples\_leaf for training the model. The max\_features returned was 18 which corroborates with the outcome of the scree diagram and min\_samples\_leaf was 5. To handle class imbalance, class\_weight was set to “balanced”.

The Random Forest classification returned an accuracy of 86.56% and an MCC score of 68%.

## Model Comparison & Selection

After the model training, it is crucial to compare the various performance of the models to select which model is best for the test dataset. The table below tabulates the results.

|  |  |  |
| --- | --- | --- |
| Algorithm | Accuracy | MCC Score |
| XGBoost | 90.7% | 77% |
| Logistic Regression | 77.6% | 51% |
| Random Forest | 86.56% | 68% |

**Table 1**: *A comparison of machine learning algorithms used to train the*

*dataset and the accuracy and MCC scores from the models. All models*

*were trained using Principal Component Analysis and cost-sensitive*

*was factored.*

From the results above, XGBoost gave a better accuracy, and MCC score of 90.7% and 77% respectively followed by Random Forest and then Logistic Regression in that order. XGBoost was chosen as the optimal model since it had the better results and generalized better.

## Conclusion

The dataset we were presented with had 451 columns with 4 columns having no meaning and irrelevant to the model training. The columns were reduced to 95 after all columns with zeros were eliminated because they will introduce noise in the data. The dimensions were further reduced using Principal Component Analysis after a scree plot was used to determine the number of components which was further confirmed by the max\_features from the Random Forest. The XGBoost performed better in the classification out of the three algorithms used in training and thus was chosen for model development and for predicting the classes on the test data.

It is recommended that more bagging algorithms be used to further train the dataset to test if there are better algorithms. Moreover, a data dictionary should be made available as some categorical variables could be essential in training the model.