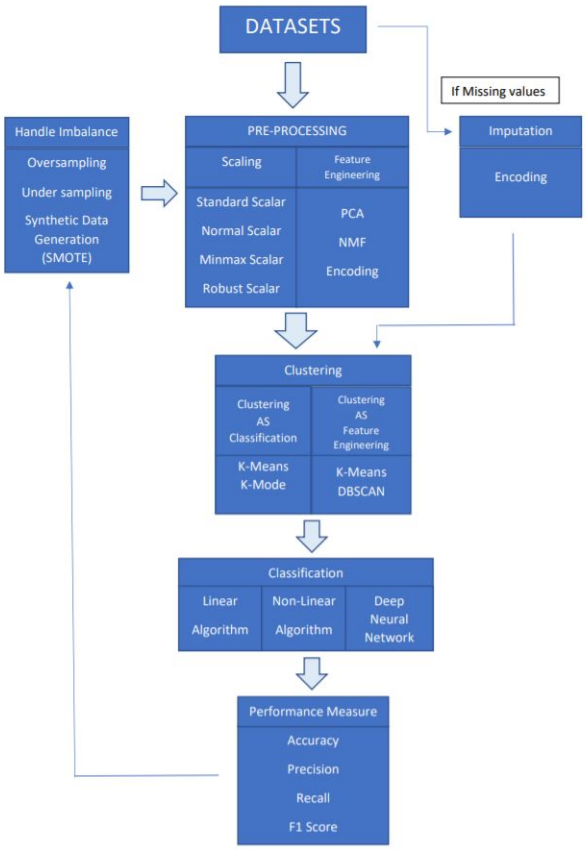
Applied Data Mining Strategies & Evaluation with Automated Report

Pritam Mishra
ID 35390350

***Abstract*— The state of data mining is eager to improve as organizations of all shapes and sizes are focusing on digging deeper into organized data to secure future investments as well as the customer experience. Although, data mining is still in its infancy, organisations in a wide range of industries - including but not limited to retail, finance, heath care, manufacturing, transportation, aerospace - are already applying data mining applications and strategies to take advantage of historical data. Although the idea seems fairly simple, the entire process of recognizing patterns in data to applying mining techniques in order to produce significant facts, relationships, trends, detection of anomaly & predictions, can be challenging from a pragmatic standpoint considering the randomness of data in real world. In this project, An application has been developed to address such challenges, while considering all possible tools & techniques that can be applied on real world data & reverse engineer the process if required to produce improvement in evaluation. This application not only implements the most commonly used pre-processing(PCA, NMF, scaling, imputation, handling imbalance), clustering(K-Means, DBSCAN, K-Mode) and classification(KNN, Decision Tree, Deep Neural Network etc) techniques; rather it also compares between these techniques & custom developed version of these techniques. Further, the application generates a report that compares performance metrics of all the techniques to decide the best possible solution for that data-set. This application could be really useful to generate insights from an unknown data-set without any prior knowledge about the data in few minutes.**

I. INTRODUCTION

Data mining, or knowledge discovery, is the computer-assisted process of digging through and analyzing enormous sets of data and then extracting the meaning of the data. Data mining tools predict behaviors and future trends, allowing businesses to make proactive, knowledge-driven decisions. Data mining tools can answer business questions that traditionally were too time consuming to resolve. Likewise, the purpose of this project is to save enormous amounts of time of the user by producing auto generated report comparing between most popular techniques in data mining. The insight can help decide the user best technique to deal with that particular data-set without much knowledge about data mining or behavior/patterns of data in the context. The application is developed considering further improvement in future & therefore if more classification algorithm or clustering algorithm needs to be added to this application, only that class needs to be updated within the array of objects because of the class object modular implementation of this application. This application has the scope to reverse engineer the entire mining process after the first run if there is class imbalance in the data, which translates to the flexibility & scope of this project. In addition, the application also deals with missing data if required & even if most of the data is categorical. This application also explores & provides performance metrics for every possible combination of different stages of data mining. For example, classification algorithms applied directly on data or applied classifiers after scaling or classification algorithms after scaling with pre-processing or classification after clustering as feature engineering & so on. The workflow of this application can be explained as,



II. PRE-PROCESSING & FEATURE ENGINEERING

In this project several pre-processing techniques have been implemented like four different scaling techniques & imputation have been applied from scratch to handle missing data within the data-set. Additionally, encoding techniques have been implemented to deal with categorical data within the data-set. Since most of the data-sets usually have more than two columns in real life, feature engineering is one of the most important techniques that needs to be implemented for dimensionality reduction. Although, the best way to

visualise the data is to reduce it to two dimensions; it might not be the best possible way considering the nature of the data. Even if the nature of the data is in favor of reducing the dimensions into two dimensions, there will always be loss of information compared to data in it's original state. Therefore, this project divides feature engineering techniques into two subgroups, one group of techniques will reduce the dimensions into two dimensions while the other group will determine all the components without any loss of information & finally both the groups will pass the components to the next stage of data mining process.

A. Scaling

Usually, most data-sets contain features highly varying in magnitudes, units & range. However, since most of the machine learning algorithm determines euclidean distance between data points, this creates erroneous computation of distance which leads to inaccurate results. The results can vary greatly between different units for example, 1000 metres & 1 km. If not scaled 1000 metres will weight a lot more than 1 km for this example.

1) Standardisation: The result of standardization (Z-score normalization) is that the features will be re-scaled so that they'll have the properties of a standard normal distribution with

$$\mu = 0 \quad \sigma = 1 \tag{1}$$

where μ is the mean (average) and σ is the standard deviation from the mean; standard scores (also called z scores) of the samples are calculated as follows:

$$z = \frac{x - \mu}{\sigma} \tag{2}$$

Standardizing the features so that they are centered around 0 with a standard deviation of 1 is not only important for measurements that have different units, but also a general requirement for many machine learning algorithms.

2) MinMaxScaler: MinMaxScaler subtracts the minimum value from the feature and then divides by the range. The range is the difference between the original maximum and original minimum. MinMaxScaler preserves the shape of the original distribution. It does not change the information embedded in the original data.The MinMaxScaler works well for cases when the distribution is not Gaussian or when the standard deviation is very small. However, it is sensitive to outliers — though this can be rectified by RobustScaler, which we will see soon.The equation for MinMax scalar can be given as,

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3}$$

3) RobustScaler: RobustScaler transforms the feature vectors by subtracting the median and then dividing by the interquartile range. RobustScaler uses a similar method to the Min-Max scaler. However, it uses the interquartile range instead of the min-max, which makes it robust to outliers. The formula for RobustScalar can be given as,

$$X_r = \frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)} \tag{4}$$

4) Normalizer: The Normalizer scales each value by dividing each value by its magnitude in n-dimensional space for n number of features.For example, if features of the data-set were x, y, and z, the scaled value for x would be:

$$X_N = \frac{x_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}} \tag{5}$$

Instead of scaling column wise rather normalizer scales data for each row. Each sample (i.e. each row of the data matrix) with at least one non zero component is re-scaled independently of other samples so that its norm (l1 or l2) equals one.

B. Handling Class Imbalance

Imbalanced classes are a common problem in machine learning classification where there are a disproportionate ratio of observations in each class.Most machine learning algorithms work best when the number of samples in each class are about equal. This is because most algorithms are designed to maximize accuracy and reduce error. However, real world data might not have balance between the classes which could be deceiving for users as machine learning algorithm will produce a very high accuracy not considering the importance of a minor class in the data-set. Applying inappropriate evaluation metrics for model generated using imbalanced data can be misleading. Therefore, several performance metrics(confusion matrix, precision, recall, f1 score) need to be considered that could provide a better insight on an imbalanced data-set. These performance measures will be explained in details in the evaluation of model section. The techniques which were implemented in this project to deal with imbalanced classes can be explained as follows,

1) Balanced Over-sampling & Under-sampling: Under-sampling balances the data-set by reducing the size of the abundant class using random selection of data to be removed. However this makes the entire data-set very small which is loss of a lot of information.For example, if the ratio of major class to minor class is $10 : 1$ in a data-set when under-sampled to match with the size of the minor class; the size of entire data-set becomes one fifth. Over-sampling on the other hand balances the data-set by increasing the size of minor class samples through random sampling among the minor class data. For the same example if the minor class needs to produce 9 times the data it holds, it'll always create duplicate data points, which is again not a perfect technique although deals with class imbalance. The balanced technique that has been used in this project utilizes both the techniques together to create a balance between over-sample & under-sample. In a balanced approach the mean class size of both the major & minor class has been taken into consideration. Therefore, the major class is under-sampled to mean class size & minor class is over-sampled to mean class size which produces exactly the same amount of rows that the original data initially had. This approach is more robust & less biased towards sampling data.

2) Balanced Under-sampling & oversampling with SMOTE: The alternate approach to randomly over-sample

data which is susceptible to duplicate entries, is called SMOTE or Synthetic Minority Over-sampling Technique. In this approach any minor class data is chosen randomly & then it's nearest neighbors were found. Now one of it's nearest neighbors is chosen randomly & any point between distance of this point & the original point is the synthetic data for over-sampling. In this project balanced mean based under-sampling is combined with SMOTE for over sampling to produce a robust & balanced data-set to deal with class imbalance.

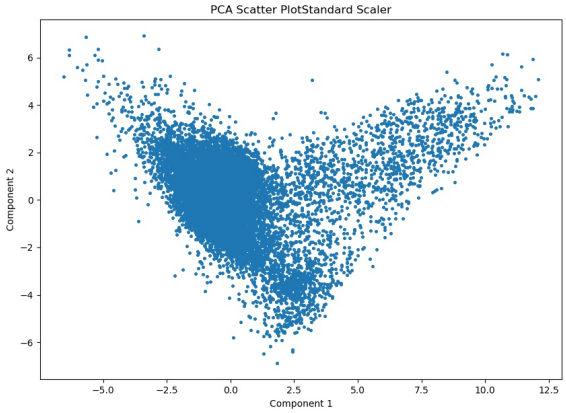
C. Handling Missing Data

Many real-world data-sets may contain missing values for various reasons. They are often encoded as NaNs, blanks or any other placeholders. Training a model with a dataset that has a lot of missing values can drastically impact the machine learning model's quality. Some of the algorithms such as scikit-learn estimators assume that all values are numerical and have and hold meaningful value.

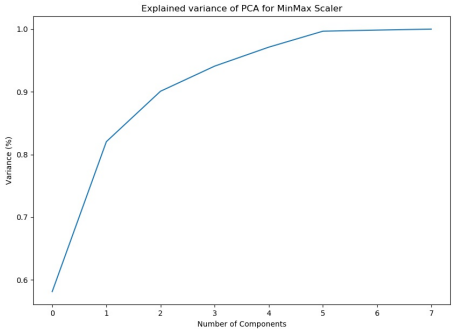
1) Imputation: In order to deal with this challenge mean/median based imputation techniques are generally applied. Usually the missing data or Nan values are replaced by the mean or median of all numerical data of that particular feature vector. In this project, challenges with missing data was found in mushroom data-set which contains categorical data for the entire data-set. Since, it is not possible to calculate mean or median of categorical variables, mode based imputation technique has been implemented. In this technique the missing data is replaced by the mode or most frequently used category of that feature vector or column of the data-set.

D. Principal Component Analysis

Principal-component analysis proposed by Hotelling (1933) is one of the most familiar methods of multivariate analysis which uses the spectral decomposition of a correlation coefficient or covariance matrix. It is a method that rotates the dataset in a way such that the rotated features are statistically uncorrelated. This rotation is often followed by selecting only a subset of the new features, according to how important they are for explaining the data. Therefore this technique is very useful in reducing the dimensions of the data-set, as it creates new components based on the covariance of all the features & the most important components can then be extracted to produce useful insights or patterns from the data-set.



In general PCA is extensively used to reduce the dimensions of the data-set & extract two most important components which could then be visualised in a two dimensional space. However, if the first two components with highest explained variance does not correspond to most of the explained variance of all the features, this technique of reducing to two dimensions or three dimensions might not be that useful as there is loss of information to not consider the other components. Generally if the first two components has a total explained variance about $90 - 95$ percent, then visualisation of those components in two dimensional space & further feeding it to supervised algorithms can be very useful. Therefore, in this project application of PCA is divided into two subgroups, first PCA is applied to extract only two components to visualise the components in two dimensional space & further fed into supervised algorithms. Secondly another PCA is being implemented to return all the components that explains a total of 95 percent variance across the entire data-set & all those components were supplied to machine learning classifiers for training & performance evaluation. Finally both the techniques have been compared with each other for every supervised algorithms applied on this project. This methodology eliminates any scope of information loss which can be misleading if only two or three components were considered. This methodology also eliminates human intervention to check for each data-set how many components could be useful to extract.



E. Non-Negative Matrix Factorisation

Nonnegative matrix factorization (NMF) has become a widely used tool for the analysis of high dimensional data as it automatically extracts sparse and meaningful features from a set of nonnegative data vectors. The underlying intuition behind this technique is to factorize a matrix V into two matrices W and H, with the property that all three matrices have no negative elements. The interpretation of W is that each column is a basis element. By basis element it means some component that crops up again and again in all of the n original data points. These are the fundamental building blocks from which approximations can be reconstructed to all of the original data points. The interpretation of H is that each column gives the coordinates of a data point in the basis W. In other words, it shows how to reconstruct an approximation to the original data point from a linear combination of the building blocks in W

$$V = WH \tag{6}$$

Overall, NMF leads to more interpretable components than PCA, as negative components and coefficients can lead to hard-to-interpret cancellation effects.

F. Handling Categorical Data

1) Label Encoder: The label encoder has been applied to convert the categorical variables to numeric variables in this project.

III. Clustering

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different.In other words, it is the process of finding homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as euclidean-based distance or correlation-based distance

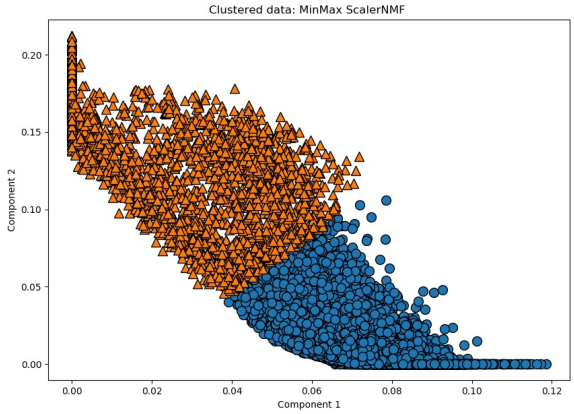
A. Clustering as Classification

Clustering techniques can also be used as classification algorithm, as the data points are being clustered into two or subsequent subgroups. If the clustered data is assigned corresponding labels based on the cluster it belongs to then labels can be compared against the true labels to evaluate the performance of that clustering technique implemented as a classifier.

1) K-Means Clustering: K-means algorithm in data mining starts with k number of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids. it calculates the euclidean distance between each points & centroid & assigns the nearest point to that centroid, further it calculates mean of all the points of that cluster & the centroid then shifts to the mean. This process repeats till there is no change of position for the centroids.

The approach kmeans follows to solve the problem is called Expectation-Maximization. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster.Mathematically the objective function can be represented as,

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik}(x_i - \mu_k)^2 \tag{7}$$



In this project, K-Means clustering is developed with scratch & compared with the scikit-learn's library function to compare the performance between the two.

2) K-Mode Clustering: Huang (1997) presented the K-modes clustering algorithm by introducing a new dissimilarity measure to cluster categorical data. The algorithm replaces means of clusters with modes (most frequent attribute value of an attribute). It uses a frequency based method to update the modes in the clustering process to minimize the cost function which is estimated by computing the normalised sum of within sum errors. This algorithm is experimentally shown to achieve convergence with linear time complexity with respect to the number of data artefacts. Huang (Huang, 1998) also points out that in general, the K-modes algorithm is faster than the K-means algorithm because it takes fewer iterations to converge.

In this project, K-Mode clustering technique has been very useful while working with categorical data-sets like mushroom data-set, where every feature vector contains categorical variables.

B. Clustering as feature engineering

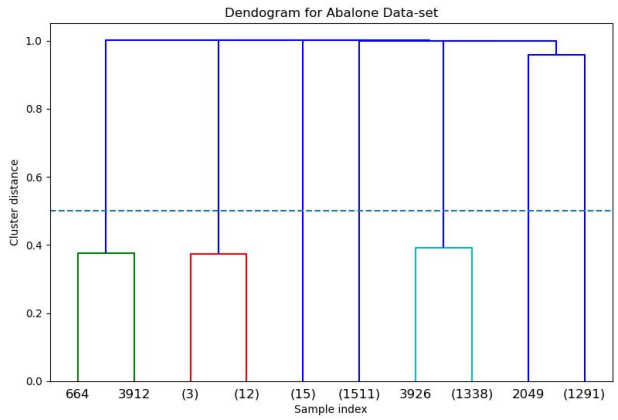
Since clustering is an unsupervised algorithm to segregate similar or dissimilar data points across the feature space, this technique can be implemented to produce distinct features of the data-set based on their similarity. The idea can be simplified as each cluster that it produces can be treated as a feature to machine learning classifiers which can improve the overall performance of the classifiers. In this project a detailed comparison has been done as it explored the performance of several classifiers when clustering is extensively

used as feature extraction compared to the performance without using it.

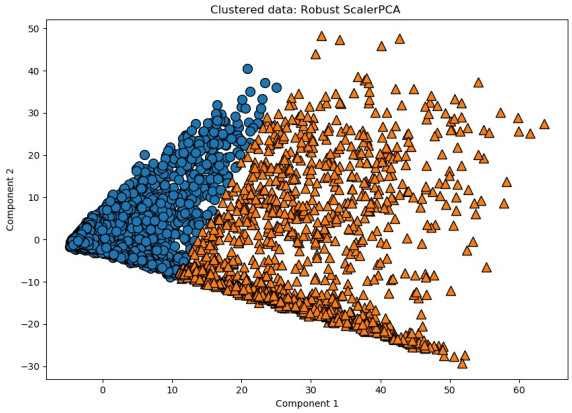
1) Agglomerative Hierarchical Clustering: The agglomerative clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It's also known as AGNES (Agglomerative Nesting). The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects. The result is a tree-based representation of the objects, called dendrogram.

The dendrogram shows data points as points on the bottom as a tree is plotted with these points (representing single-point clusters) as the leaves, and a new node parent is added for each two clusters that are joined to for such a tree structure as part of the hierarchical clustering.

In this project, The dendrogram has been used separately in a program to decide the optimal possible number of clusters that the user can opt. The number of clusters can then be passed as a parameter in the main application which will start feature engineering using the algorithms like K-Means Clusturing where the k is provided by the user based on the insights from dendrogram.



2) K-Means for Feature Extraction: Hierchical clustering techniques can be useful to decide optimal number of clustering required for a specific data-set. Once it is combined with the K-Means algorithm where k number of clusters are provided by the user based on the insights from hierarchical clustering, the combination can be really powerful toward extracting new features which could improve the performance overall for the classifiers. Unfortunately K-Means or Agglomerative clustering techniques fails at separating complex shapes, therefore a more advanced algorithm like DBSCAN can be implemented in those cases. In this project, all of these techniques has been implemented to ensure the comprehensive analysis of the data-sets.



3) DBSCAN: Another very useful clustering algorithm is DBSCAN (which stands for "density based spatial clustering of applications with noise"). The main benefits of DBSCAN are that it does not require the user to set the number of clusters as a parameter compared to K-Means algorithm, it can capture clusters of more complex shapes, and it can identify points that are not part of any cluster which can be termed as noise in the data. DBSCAN is somewhat slower than agglomerative clustering and k-means, but still scales to relatively large data-sets. DBSCAN works by identifying points that are in "crowded" regions of the featurespace, where many data points are close together. These regions are referred to as dense regions in feature space. The idea behind DBSCAN is that clusters form dense regions of data, separated by regions that are relatively empty.

In this algorithm, data points that are within a dense region are called core samples (or core points), and they are defined as follows. There are two parameters in DBSCAN: `min_samples` and eps.If there are at least `min_samples` many data points within a distance of eps to a given data point, that data point is classified as a core sample. Core samples that are closer to each other than the distance eps are put into the same cluster by DBSCAN. In this project DBSACN has been implemented to produce impressive results for feature extraction while working with machine learning classifiers together.

IV. Classification

In machine learning and statistics, classification is a supervised learning approach in which the program learns from the data input given to it and then uses this learning to classify new observations. The types of classification algorithms implemented in this project to work with pulser, abalone & mushroom data-sets, can be explained as follows,

- Linear Classifiers: Logistic Regression [API based and custom algorithm from scratch]
- K Nearest Neighbors
- Decision Trees
- Neural Networks

A. Logistic Regression

Logistic Regression is one of the most used Machine Learning algorithms for binary classification. It is a simple

Algorithm that can be used as a performance baseline, it is easy to implement and it will do well enough in many tasks. Logistic Regression measures the relationship between the dependent variable (labels) and the one or more independent variables (features), by estimating probabilities using it's underlying logistic function.These probabilities must then be transformed into binary values in order to actually make a prediction. This is the task of the logistic function, also called the sigmoid function. The sigmoid function takes a any number as input and map it into a value between the range of 0 and 1. The value in between 0 and 1 can then be transformed into either 0 or 1 using a threshold classifier. This process is repeated using gradient descent or stochastic gradient descent algorithm to minimize the loss function at the time of training the data & the optimal weights or coefficients can be found to predict the data accurately.

B. K Nearest Neighbors

The kNN algorithm is one of the simplest machine learning algorithms where it finds the k number of nearest neighbors by calculating the euclidean distance between this point & all the other points. Once it finds it's k number of nearest neighbors, it uses a voting scheme to assign a label to the test data based on the majority of it's k nearest neighbor's labels. That is the test data is assigned the same label as most frequent or majority class among the k neighbors.

C. Decision Tree

Decision tree builds classification models in the form of a tree structure. It splits the sample into two or more homogeneous sets (leaves) based on the most significant differentiators among the input variables. To choose a differentiator (predictor), the algorithm considers all features and does a binary split on them (for categorical data, split by cat; for continuous, pick a cut-off threshold).The it chooses the one with the least cost (i.e. highest accuracy), and repeats recursively, until it successfully splits the data in all leaves (or reaches the maximum depth). Decision Tree is simple to understand and visualise, requires little data preparation, and can handle both numerical and categorical data.

D. Deep Neural Network

Neural Networks is one of the most popular machine learning algorithms at present. It has been decisively proven over time that neural networks outperform other algorithms in accuracy and speed. As the name suggests, neural networks were inspired by the neural architecture of a human brain, and like in a human brain the basic building block is called a Neuron. Its functionality is similar to a human neuron, i.e. it takes in some inputs and produces an output.The function used in a neuron is generally termed as an activation function. There have been 5 major activation functions which gained popularity & used very often in neural Networks, step, sigmoid, tanh, and ReLU.

1) Step Function: Step function is defined as it produces an output of 1 if the value of independent variable x is greater than equals zero & it is zero if the value of x is less than zero.

2) Sigmoid Function: The sigmoid function is defined as the value of the function tends to zero when z or independent variable tends to negative infinity and tends to 1 when z tends to infinity.

3) Tanh Function: The tanh(z) function is a rescaled version of the sigmoid, and its output range is [-1,1] instead of [0,1].

4) ReLU Function: The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. So, it can be written as $f(x) = max(0, x)$

In a deep neural network, each layer can have same or different activation function. In a deep neural network each input layer generates an output with the help of a nonlinear activation function which is sent to the next layer which again uses activation function until it reaches the output layer. After one forward pass the network calculates the gradients of all the activation functions through chain rule of derivative, which is also known as the back- propagation algorithm. The idea is to find the change in gradient of the entire network for a small change in the parameters or weights of the network; the small change is also known as the learning rate. The goal of the network is to reduce the loss or gradient function of the network by changing the parameter in each iteration. If the algorithm reduces the loss function for each iteration of training the entire data through forward pass & back propagation that is known as Gradient Descent algorithm, where the weights of the network is updated for each epoch. If those weights of the network are updated for each training data or row of the data-set within an epoch, that is known as Stochastic Gradient Descent algorithm.

In this project, custom neural network is designed with one hidden layer containing 16 neurons & and an output layer, where the hidden layer is using a ReLU activation function while the output layer is using a sigmoid activation function. The performance of the neural network can be found later or in the auto-generated report.

V. Performance Metrics

A. Classification Accuracy

It is usually meant by the accuracy of the model provided there is no class imbalance in the data-set. In case there is class imbalance in the data-set, the data-set can be over-sampled & under-sampled together as discussed earlier as the balanced mean based handling class imbalance approach that was applied in this project. The other alternative way to analyze the accuracy of the model is to obtain the other advanced performance measures which will be discussed soon. The accuracy here can be defined as ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{CorrectPrediction}{Total\ number\ of\ predictions} \tag{8}$$

B. Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data

for which the true values are known. There are four sections in a confusion matrix which can be defined by the four terms as follows,

- **True Positives:** The cases in which the predicted output is YES and the actual output was also YES
- **True Negatives:** The cases in which the predicted output is NO and the actual output was also NO.
- **False Positives:** The cases in which the predicted output is YES and the actual output was NO.
- **False Negatives:** The cases in which the predicted output is NO and the actual output was YES.

C. Area Under Curve

Area Under Curve(AUC) is one of the most widely used metrics for evaluation. It is used for binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example.

- **True Positive Rate (Sensitivity):** True Positive Rate is defined as TP/(FN+TP). True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.

$$True\ Positive\ Rate = \frac{True\ Positive}{False\ Negative + True\ Positive} \tag{9}$$

- **False Positive Rate (Specificity):** False Positive Rate is defined as FP/(FP+TN). False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

$$False\ Positive\ Rate = \frac{False\ Positive}{False\ Positive + True\ Negative} \tag{10}$$

False Positive Rate and True Positive Rate both have values in the range [0, 1]. FPR and TPR both are computed at threshold values such as (0.00, 0.02, 0.04, …., 1.00) and a graph is drawn. AUC is the area under the curve of plot False Positive Rate vs True Positive Rate at different points in [0, 1].

D. F1 Score

F1 Score is defined as the Harmonic Mean between precision and recall.

1) Precision :: It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{11}$$

2) Recall :: It is the number of correct positive results divided by the number of all samples that should have been positive.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{12}$$

Now the F1 score can be defined as,

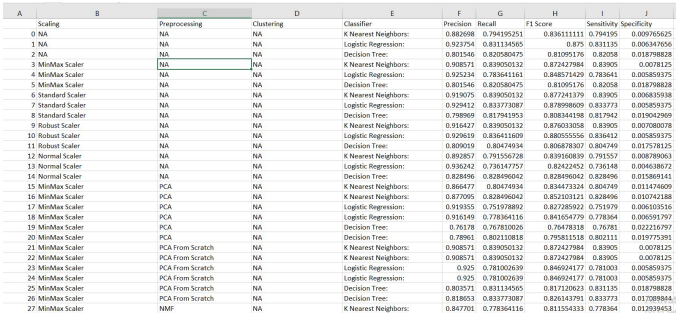
$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{13}$$

F1 Score tries to find the balance between precision and recall. In short, the greater the F1 Score, the better is the performance of the given model.

VI. CONCLUSIONS

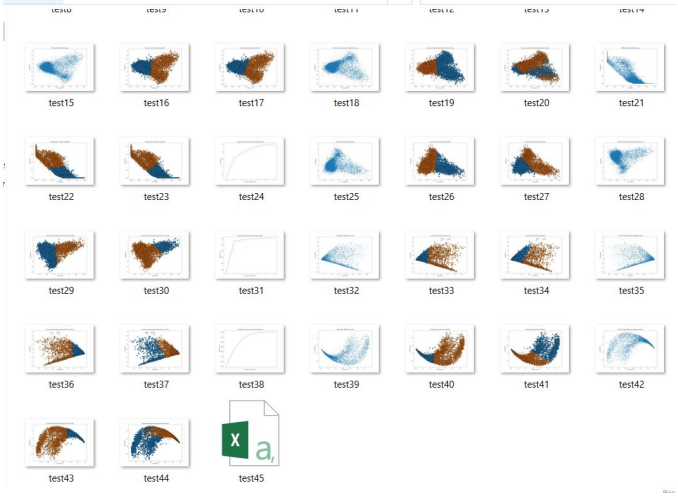
Although, machine learning classifiers when applied directly to the data-set can produce higher accuracy. But the accuracy metric by itself might not be the measure of how well the model will generalise the data. Hence, handling class imbalance producing confusion matrix to determine precision recall of the model is very crucial. Simultaneously, handling the missing data could be a big challenge in real life, where more advanced imputation techniques must be applied. In this project, categorical missing data has been dealt with mode based imputation approach. Therefore the optimal model for any data-set requires combinations mining strategies along with building several models to come to a conclusion.

APPENDIX



	Scaling	Preprocessing	Clustering	Classifier	Precision	Recall	F1 Score	Sensitivity	Specificity
0	NA	NA	NA	K Nearest Neighbors:	0.882698	0.794195251	0.836111111	0.794195	0.009765625
1	NA	NA	NA	Logistic Regression:	0.923754	0.831134565	0.875	0.831135	0.006347656
2	NA	NA	NA	Decision Tree:	0.801546	0.820580475	0.81095176	0.82058	0.018798828
3	MinMax Scaler	NA	NA	K Nearest Neighbors:	0.908571	0.839050132	0.872427984	0.83905	0.0078125
4	MinMax Scaler	NA	NA	Logistic Regression:	0.925234	0.783641161	0.848571429	0.783641	0.005859375
5	MinMax Scaler	NA	NA	Decision Tree:	0.801546	0.820580475	0.81095176	0.82058	0.018798828
6	Standard Scaler	NA	NA	K Nearest Neighbors:	0.919075	0.839050132	0.877241379	0.83905	0.006835938
7	Standard Scaler	NA	NA	Logistic Regression:	0.929412	0.833773087	0.878998609	0.833773	0.005859375
8	Standard Scaler	NA	NA	Decision Tree:	0.798969	0.817941953	0.808344198	0.817942	0.019042969
9	Robust Scaler	NA	NA	K Nearest Neighbors:	0.916427	0.839050132	0.876033058	0.83905	0.00708078
10	Robust Scaler	NA	NA	Logistic Regression:	0.929619	0.836411609	0.880555556	0.836412	0.005859375
11	Robust Scaler	NA	NA	Decision Tree:	0.809019	0.80474934	0.806878307	0.804749	0.017578125
12	Normal Scaler	NA	NA	K Nearest Neighbors:	0.892857	0.791556728	0.839160839	0.791557	0.008789063
13	Normal Scaler	NA	NA	Logistic Regression:	0.936242	0.736147757	0.82422452	0.736148	0.004638672
14	Normal Scaler	NA	NA	Decision Tree:	0.828496	0.828496042	0.828496042	0.828496	0.015869141
15	MinMax Scaler	PCA	NA	K Nearest Neighbors:	0.866477	0.80474934	0.834473324	0.804749	0.011474609
16	MinMax Scaler	PCA	NA	K Nearest Neighbors:	0.877095	0.828496042	0.852103121	0.828496	0.010742188
17	MinMax Scaler	PCA	NA	Logistic Regression:	0.919355	0.751978892	0.827285922	0.751979	0.006103516
18	MinMax Scaler	PCA	NA	Logistic Regression:	0.916149	0.778364116	0.841654779	0.778364	0.006591797
19	MinMax Scaler	PCA	NA	Decision Tree:	0.76378	0.767810026	0.764783218	0.76781	0.022216797
20	MinMax Scaler	PCA	NA	Decision Tree:	0.78961	0.802110818	0.795811518	0.802111	0.019775391
21	MinMax Scaler	PCA From Scratch	NA	K Nearest Neighbors:	0.908571	0.839050132	0.872427984	0.83905	0.0078125
22	MinMax Scaler	PCA From Scratch	NA	K Nearest Neighbors:	0.908571	0.839050132	0.872427984	0.83905	0.0078125
23	MinMax Scaler	PCA From Scratch	NA	Logistic Regression:	0.925	0.781002639	0.846924177	0.781003	0.005859375
24	MinMax Scaler	PCA From Scratch	NA	Logistic Regression:	0.925	0.781002639	0.846924177	0.781003	0.005859375
25	MinMax Scaler	PCA From Scratch	NA	Decision Tree:	0.803571	0.831134565	0.817120623	0.831135	0.018798828
26	MinMax Scaler	PCA From Scratch	NA	Decision Tree:	0.818653	0.833773087	0.826143791	0.833773	0.017089844
27	MinMax Scaler	NMF	NA	K Nearest Neighbors:	0.847701	0.778364116	0.811554333	0.778364	0.012939453

.The comparison of several machine learning models as a result is extracted in an auto-generated csv file in this project within a program generated directory which will contain all the program generated plots of several models.



REFERENCES

[1] (1, 2) Huang, Z.: Clustering large data sets with mixed numeric and categorical values, Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, Singapore, pp. 21-34, 1997..

[2] https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e14

[3] https://towardsdatascience.com/predict-malignancy-in-breast-cancer-tumors-with-your-own-neural-network-and-the-wisconsin-dataset-76271a05e941

[4] https://medium.com/datadriveninvestor/classification-algorithms-in-machine-learning-85c0ab65ff4

[5] https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

[6] Andreas C. Müller Sarah Guido: Introduction to Machine Learning with Python