Report

Bank Subscriber Identification

Ganesa Rukmani Ramdas Pillai
pillairamdas@gmail.com
(213) 322-9285

Table of Contents

Table of Figures

1 Abstract

The ability of humans to recognize patterns is one of the key aspects that separates us from machines. It is quite natural that we would seek to design and create machines that could also perform pattern recognition. The applications of such a design is vast ranging from automated speech recognition, fingerprint identification, optical character recognition, DNA sequence identification many more. Here, we present the Bank Marketing Dataset of the University of California, Irvine [11]. Here we will be estimating if a customer will subscribe to the bank or not based on the number of features collected. This is a more practical and real-life pattern classification problem that has huge market implications.

2 Introduction

Pattern Recognition is the act of taking in raw data and take actions based on the category of the pattern. As seen in figure below (Source [13]), pattern recognition process can be split into various paradigms. The most striking ones are as stated below. Let us accompany each step along with an example of a practical use-case like music genre recognition.

- **Data Acquisition:** This is one of the most primitive steps in the process of recognizing a pattern. Here, the phenomena from the physical is captured and generally converted into a digital data. This is said to be in measurement space. In our example, this step can be accounted for as capturing the audio into a digital media file.
- **Pre-Processing:** Pre-processing of data is one of the key steps in simplifying the subsequent operations. Here the complex data obtained in the data acquisition step is simplified so that each data point collected can be represented by a set of variables. Here the data is said to be in “pattern space”. We can visualize this step as involving processing like noise reduction and downmixing the audio data into a single channel decoded PCM file in our music genre recognition example.
- **Feature Extraction:** In this step, the pre-processed data is analyzed and useful features are extracted from the data. This process is one of the most important steps as the classification process completely depends upon how and which features are extracted. Once the features are extracted. Now we will have the data points and their corresponding set of features, this is called the “feature space”. For our example, the feature extraction stage would be gather key aspects like mean beats per minute, overall distortion in the music. These key features will help us in identifying the genre of the music.
- **Classification:** Once we have the data in the feature space, we use it to train different classifier models like SVM, artificial neural network. Once these models are trained, we can now pass our test data and if the training data sample was good enough to get good distinctive features, then the we would get the right classification. In our example, we will get to know if our music is Rock, Hip-Hop, classical or any other genre presented in the training data set.

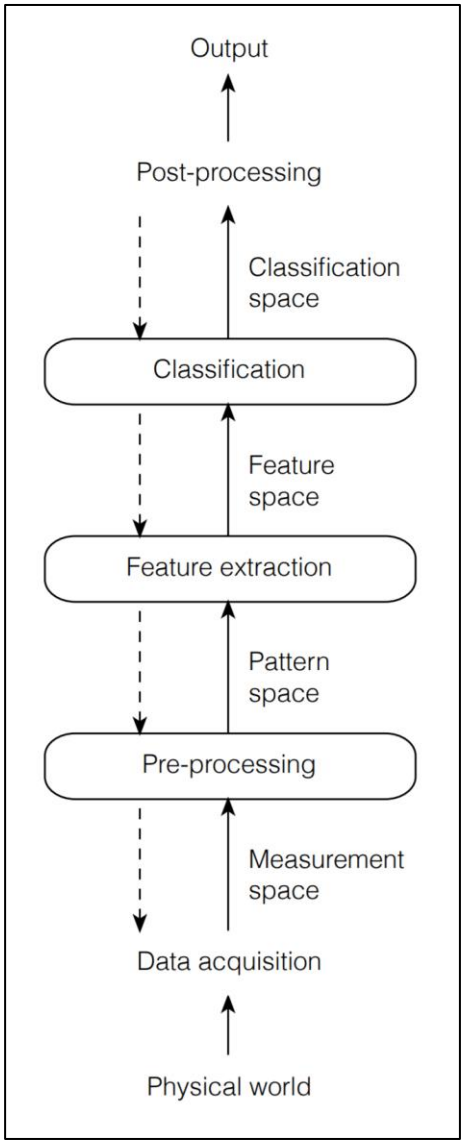


Figure 1: Pattern recognition paradigm

3 Environment

The below environment is used to execute this project. It is tested to work without any issues on below setup. The project is not tested on any other setup; however, it is expected to work on other setups as well provided the relevant packages are installed.

Name	Value
OS	Windows 10
Python Version	3.6.4
IPython Version	6.2.1
seaborn	0.8.1
scikit-learn	0.19.1
prettytable	0.7.2
pandas	0.22.0
numpy	1.14.0
matplotlib	2.1.2
imblearn	0.0.0

4 Data Pre-processing

- Pandas Data Frame: The first step
- **Data Cleaning:** One of key first steps while working with the Bank Marketing dataset is data cleaning. We can see in the below figure that most of the features have some unknown values. It is important to deal with them before we proceed to perform any kind of training.
- **Categorical String attributes:** As we can observe from the figure below, some features like "job" have attributes that are strings. This becomes an ambiguous value for most classifiers. Hence, we use data splitting methods like one hot key encoding. This will be discussed in detail for this use-case.
- **Normalization:** Normalization plays a very important role in pattern recognition and machine learning in general. Normalization helps in avoiding scenarios wherein a particular feature ends up getting more weightage than other features dues to its higher values. Here, we perform zero-mean unit-variance normalization.

	age	job	marital	education	default	housing	loan	contact	month	day_of_week
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri
1	39	services	single	high.school	no	no	no	telephone	may	fri
2	25	services	married	high.school	no	yes	no	telephone	jun	wed
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon

Figure 2: Snippet of the data set

4.1 Data Cleaning

Here we will analyze each of the 17 features that we have and check where it need any data cleaning to be performed.

- **Age:**

  This feature has below histogram:

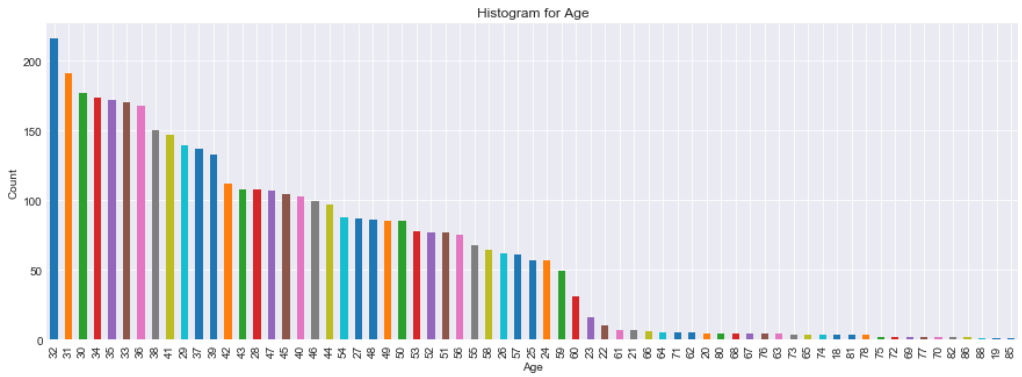


Figure 3: Histogram for feature: Age

Here, no data cleaning is required as there no unknown or missing values for this feature.

- **Job:**

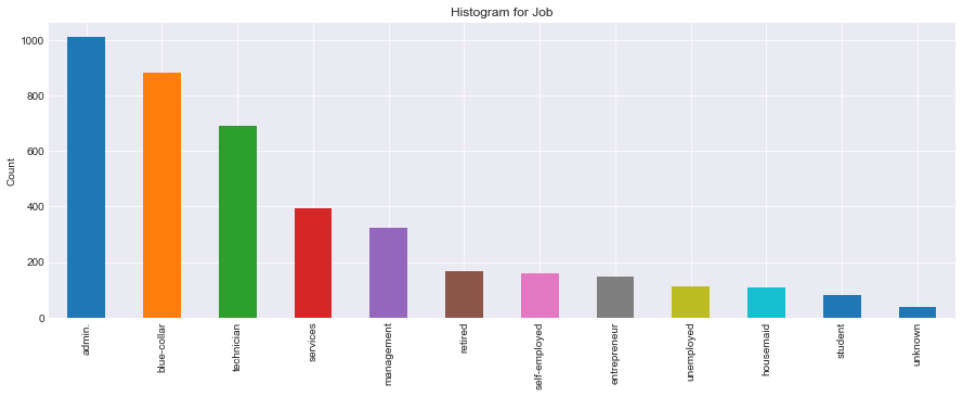


Figure 4: Histogram for feature: Job

Here, we can see that this feature has very few data points where the attribute is “unknown”. Since the number of unknown is very less. We can remove the data points which correspond to this.

- **Marital:**

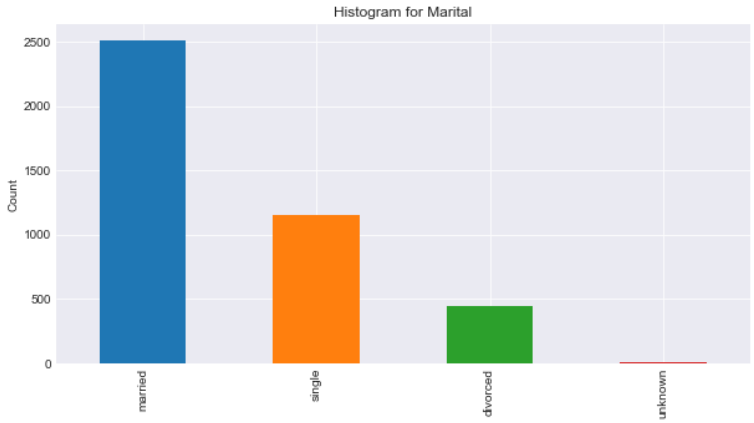


Figure 5: Histogram for feature: Marital

In this feature, we can observe that a very minute number of data points have the feature attribute as “unknown”. These data points can be removed from the dataset.

- **Education:**

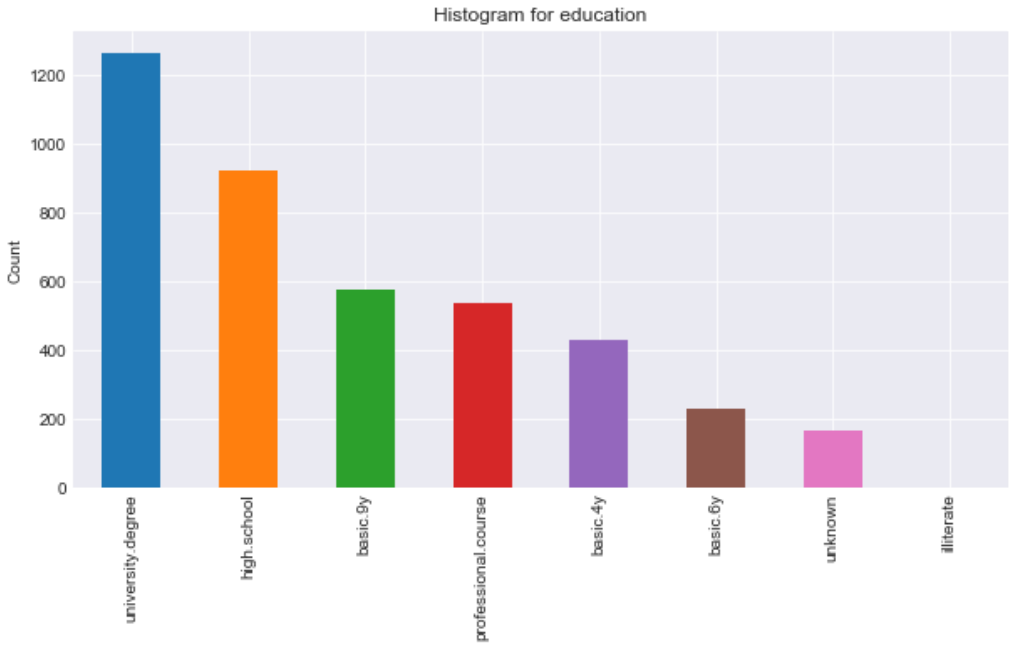


Figure 6: Histogram of feature: Education

Here we can observe that the attribute “illiterate” has almost negligible number of data points; hence, we can remove the data points corresponding to this attribute. We must understand that we will be treating unknown as a valid attribute due to that fact that many data points have this attribute.

- **Default:**

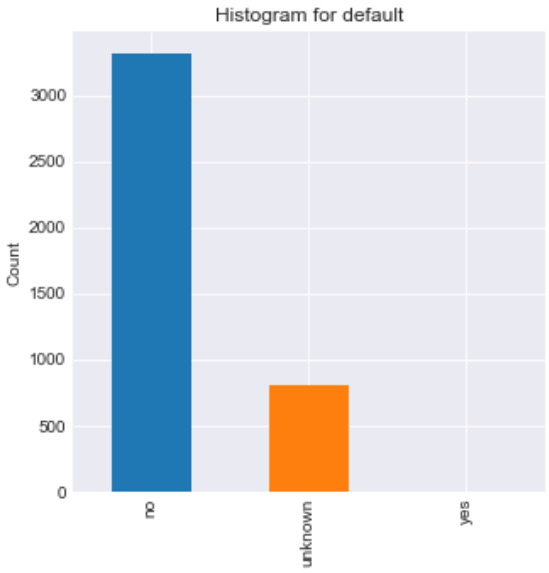


Figure 7: Histogram for feature: Default

In this feature, the number of data points with attribute “yes” are negligible and can be deleted from the dataset. Note that even here, we must not remove the data points corresponding to “unknown” attribute since there are many data points which have attribute as “unknown”.

- **Housing:**

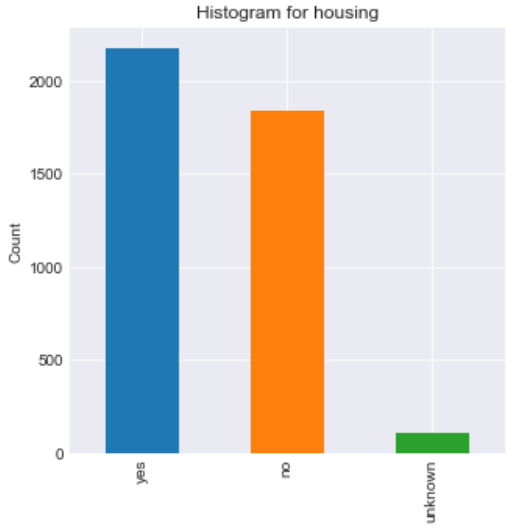


Figure 8: Histogram for feature: Housing

In this feature as we can observe, the number of data points corresponding to attribute “unknown” are very few and thus, these data points can be removed from the dataset.

- **Loan:**

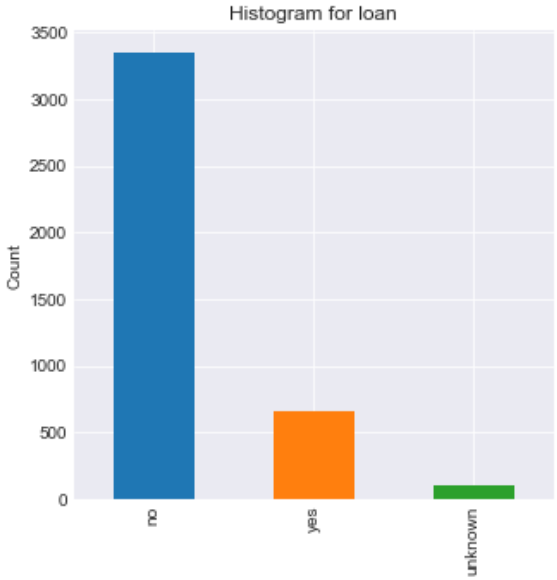


Figure 9: Histogram for feature: Loan

Similar to feature “housing”, even here we have same number of data points which have attribute “unknown”, thus while removing “unknown” data points for feature “housing”, all data points with “unknown” for this feature are also removed.

- **Contact:**

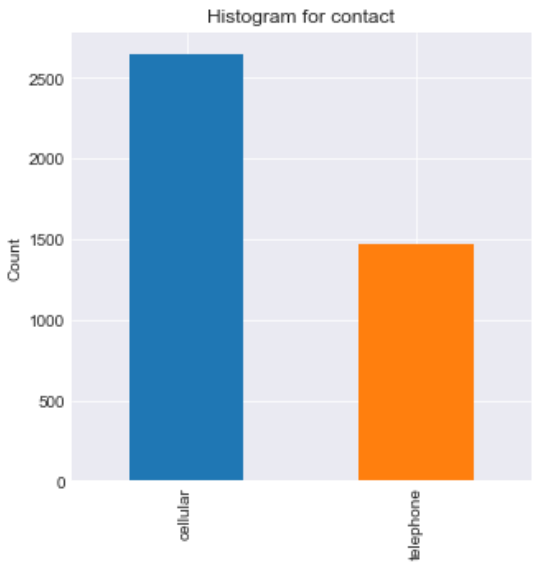


Figure 10: Histogram for feature: Contact

No need to perform any data cleaning here as both attributes have a good number of data points corresponding to them.

- **Month:**

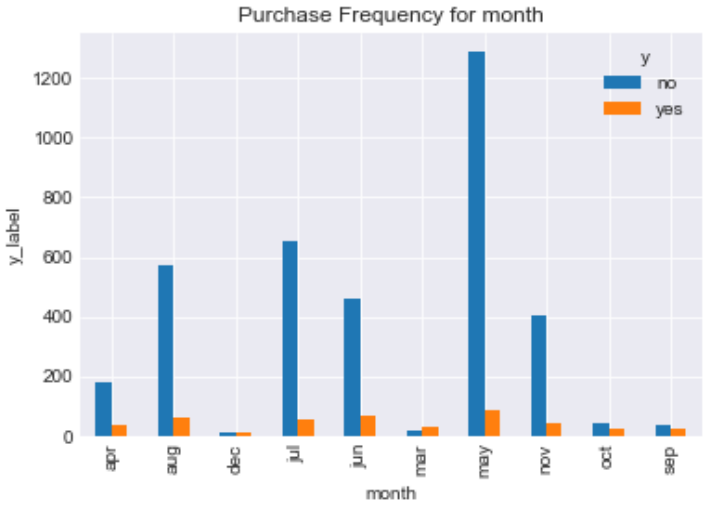


Figure 11: Histogram for feature: Month

No need to perform any data cleaning here as all attributes have a good number of data points corresponding to them. We can see from the figure above that this feature is strong predictor.

- **Day of week:**

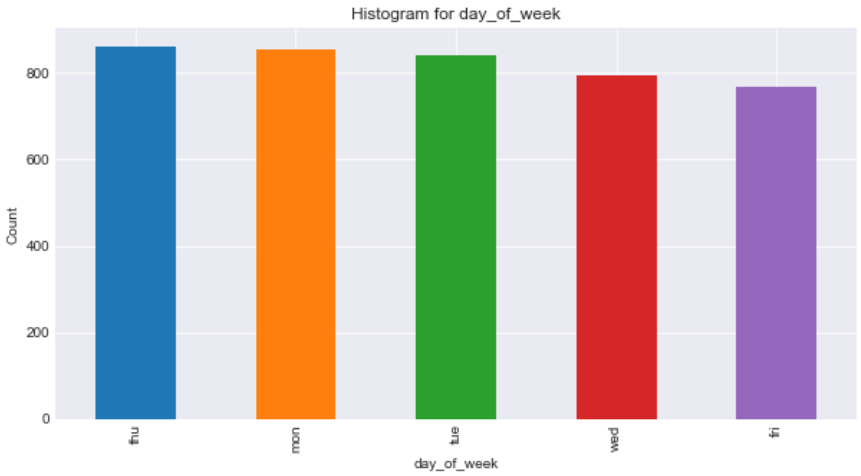


Figure 12: Histogram for feature: Day of Week

No need to perform any data cleaning here as all attributed have good number of data points.

- **Campaign:**

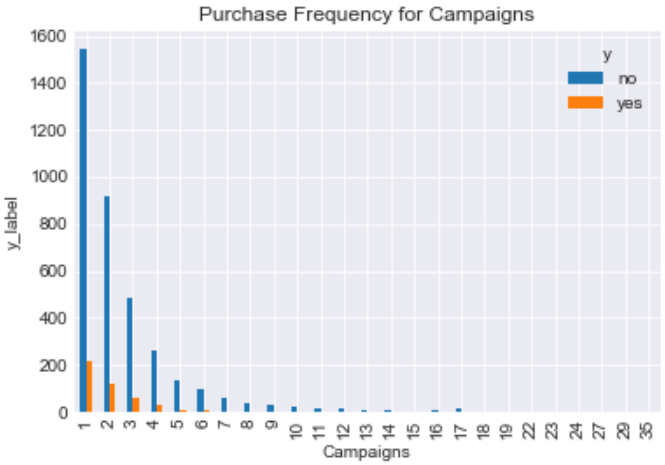


Figure 13: Purchase frequency for Campaigns

No need to perform any data cleaning here as all attributed have good number of data points.

- **PDays:**

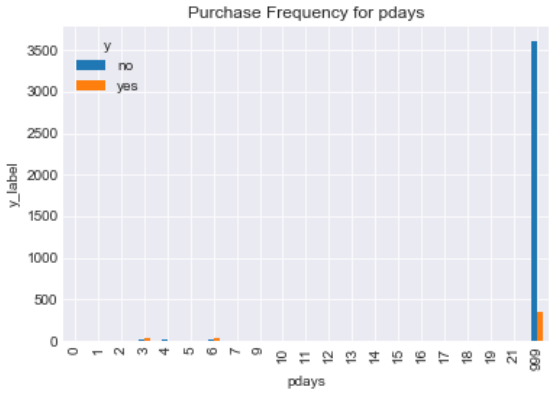


Figure 14: Purchase frequency for feature: pdays

This feature is only one attribute shows both the class output responses, and there is no other significant attribute to make much of a difference. We can drop this feature all together.

- **Previous:**

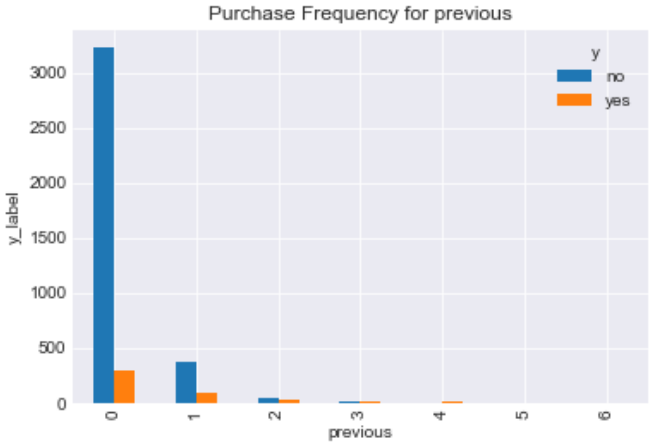


Figure 15: Purchase frequency for feature: previous

This is a good predictor feature, as with increasing values of previous, the probability of 'yes' is increasing as well. No need to perform any kind of data cleaning here.

- **Poutcome:**

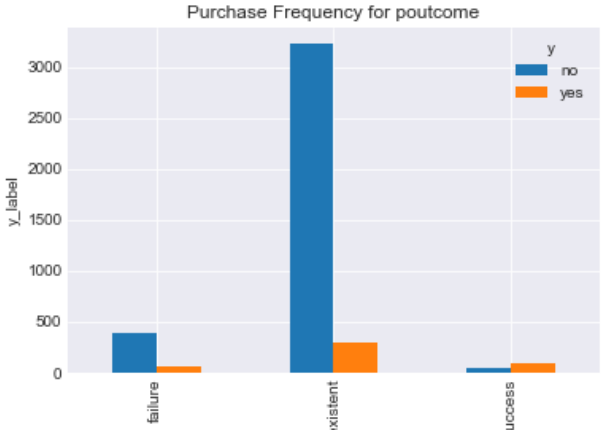


Figure 16: Purchase frequency of poutcome

This is a decent predictor. Here we must note that even though the number of data points corresponding to the attribute "success", we should not remove it. This is because, when the attribute is "success", then the probability of class "yes" is higher than "no" as compared to their probabilities for other attributes.

- **emp.var.rate:**

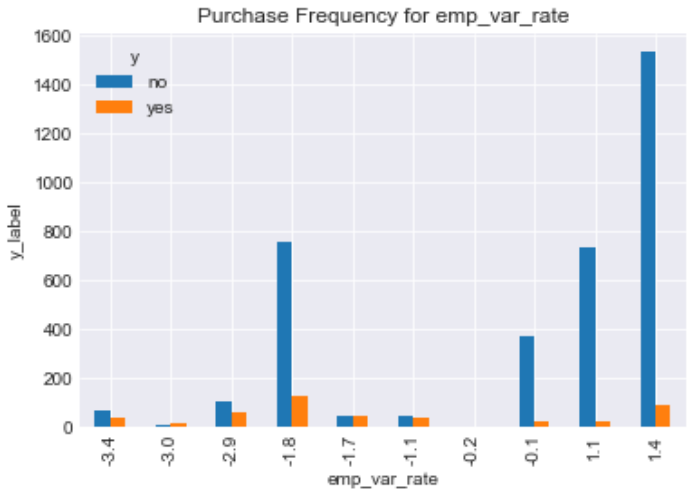


Figure 17: Purchase frequency for feature: emp.var.rate

The feature indicates the employment variation rate. No unknown data points for this feature. No need to perform any data cleaning.

- **cons.price.idx:**

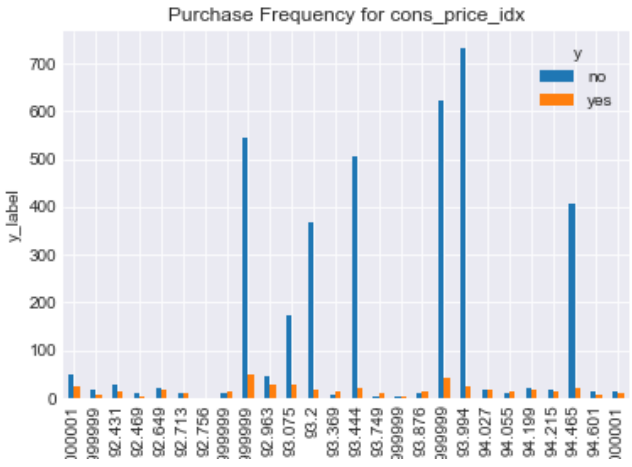


Figure 18: Purchase frequency of feature: cons.price.idx

The feature corresponds to consumer price index. No unknown data points for this feature. No need to perform any data cleaning.

- **cons.conf.idx:**

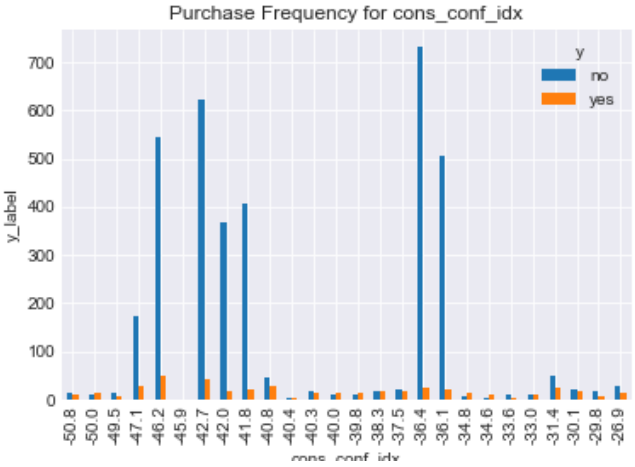


Figure 19: Purchase frequency of feature: cons.conf.idx

This feature corresponds to consumer confidence index. No unknown data points for this feature. No need to perform any data cleaning.

- **euribor3m:**

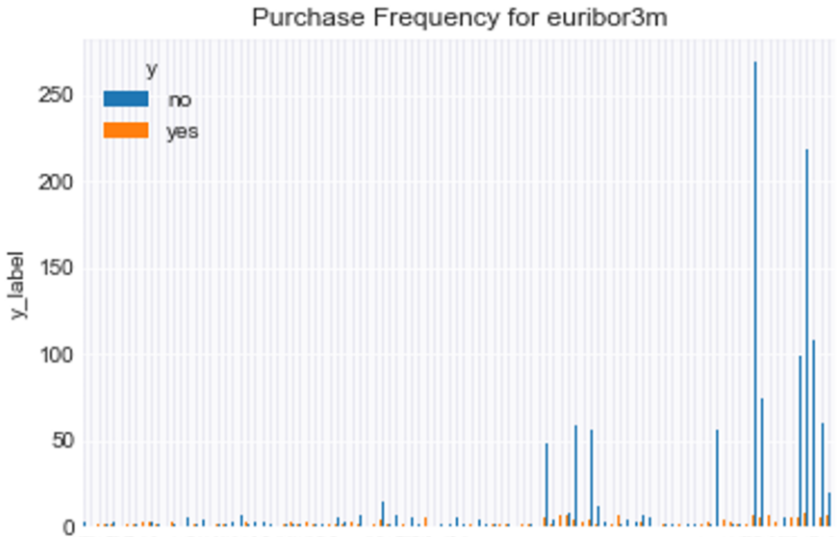


Figure 20: Purchase frequency for feature: euribor3m

This feature corresponds to Euribor 3-month rate. This is an excellent feature and since it does not have any unknown values, we do not need to perform data cleaning for this feature.

- **nr.employed:**

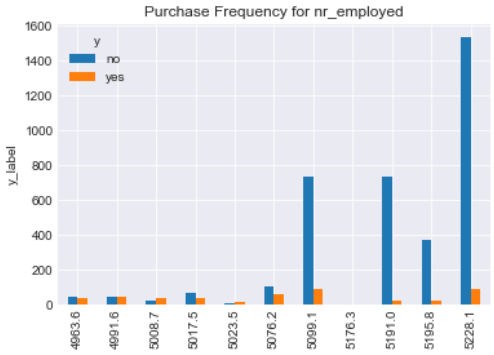


Figure 21: Purchase frequency for feature: nr_employed

This feature represents number of employees. This feature does not have any unknown values and thus, there is no need to perform any data cleaning.

4.2 Encoding categorical string attributes

As was discussed earlier some of the features have string attributes which could not be understood by the classifiers. Thus, it is important to encode this information without corrupting the meaning of the corresponding feature.

- We split the feature “Age” into three categories namely, “young”, “adult” and “old”
- Rest of the features with string attributes are encoded using one-hot key encoding. To make it clear, below is how feature “Marital” looks after one-hot key encoding.

	marital_divorced	marital_married	marital_single	marital_unknown
0	0	1	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	0	0

Figure 22: One-hot key encoding instance

- Here we must appreciate that one-hot key encoding retains the importance of feature without adding any weight to over-emphasize or de-emphasize the feature.

4.3 Normalization

In this step, we normalize the non-categorical features. Before performing normalization, we split the data into training set and testing set. Now, the mean and standard deviation of training is used to normalize the training set and testing set.

5 Feature Selection and Data Sampling Techniques

In this section, we will look upon some of feature selection and data augmentation techniques that we use to increase the performance over classification on the default dataset. Let's look at these in detail.

5.1 Feature Selection

Feature selection is one of the important process wherein we automatically select some features from the data. The features are selected such that they contribute most to the prediction output. Having useless features adds to the dimensionality and corrupts many classifiers especially linear and logistic regression [18].

Some of the major advantages of dimensionality reduction are:

- **Avoid overfitting:** Since the data to work upon gets reduced, this makes the decision boundary to be more linear and thus, avoiding overfitting.
- **Improvement of Accuracy:** Since the irrelevant data will be removed, the accuracy on this set should be theoretically higher than otherwise. Look at results section to see how the performances varies by dimensionality reduction.
- **Less training time:** Since the number of features are reduced, this has a direct impact on the computation time.

5.1.1 Principal Component Analysis (PCA)

PCA makes use of linear algebra concepts to transform the data into a compressed form. For most of the signal processing applications PCA is the go-to approach to reduce dimensionality of the data. Another advantage of using PCA to reduce dimensions is that the features now obtained are uncorrelated to each other. This plays a major role as now the features are orthogonal to each other and this results into covariance matrix being a diagonal matrix.

5.1.2 Select k best features

Select k best is another highly used approach to select the best k features. This is a quite simple and straightforward approach to selecting good features. Here, each of the feature is scored against a scoring function like f_classif() or chi2(). Once the scoring is done, the feature selection model removes all but the 'k' highest scoring features.

5.1.3 Recursive Feature Elimination (RFE)

This algorithm works by removing features recursively and also builds a model on those attributes that remain. It makes use of model accuracy to identify which combination of features contribute to generating a higher accuracy score for prediction. RFE algorithm generates a ranking of features so that the user can select the features based on the requirement. A rank of '1' corresponds to best ranking features.

5.2 Data Sampling

As we would have already noticed from the histogram plots posted earlier, the bank marketing dataset is highly imbalance with a ratio of approximately 1:10. The imbalance and poor features drastically impacts the f-score of the minority class. This begs us to look at techniques to handle this. Let's look at some approaches we take to get around this issue.

5.2.1 Synthetic Minority Over-Sampling Technique (SMOTE)

SMOTE is one of the most commonly used over sampling technique. SMOTE synthesis new synthetic data points into the dataset between the space of existing minority class data points. This results in creating a balanced data set. We must understand that SMOTE must only be applied on the training dataset and not on the testing dataset. This results in a balanced training set with an imbalanced testing set. Below figure (Source: [10]) shows the variants of SMOTE and as we can observe the SVM SMOTE works the best in creating segregated oversampled dataset.

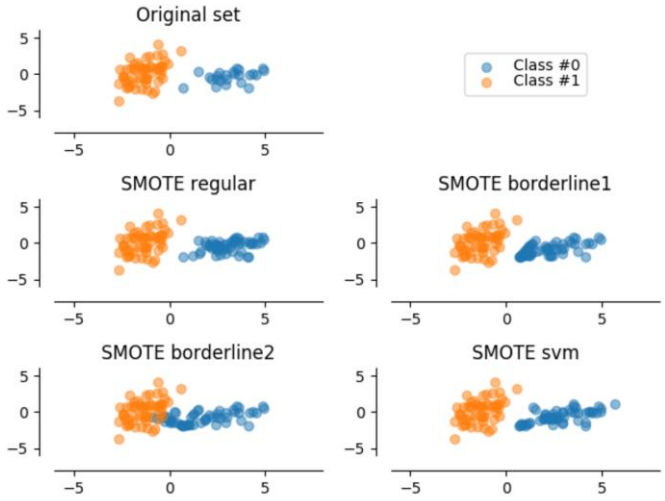


Figure 23: SMOTE variants

We will look at the results on application of SMOTE in the Results Section.

5.2.2 Subsampling to create a balanced test set

As we discussed above, SMOTE is applied only on the training set and not on the testing set. To overcome this issue, a small experiment was tried. Here while splitting the dataset into training and testing, we split the test data such that it is balanced. Then, we apply SMOTE on the training data to make it balanced as well. So what we get is an oversampled balanced training data and balanced test data. The results of this experiment should be interesting and are posted in results section.

6 Cross Validation

6.1 Stratified K-fold Cross Validation

Cross validation is the process of training the learning algorithm by using one set of data for training and testing it using a different set. Parameter tuning is performed to select the values for a classifier's parameters that maximize the accuracy of the prediction. In datasets with a very high imbalance, stratified K-fold cross validation is used. This is a slightly modified adaptation of the K-fold cross validation in the sense that each fold has the same distribution imbalance as the whole dataset. This is shown in below figure.

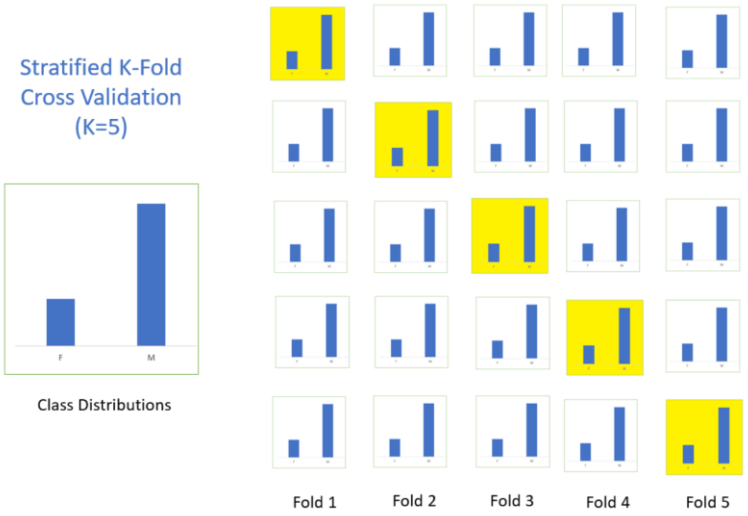


Figure 24: Stratified K-Fold cross validation for imbalanced datasets

The main purpose of validation is to identify the accuracy based on the average performance on each of the K-folds. By doing this, we ensure the robustness of the system to a varied amount of data.

6.2 GridSearchCV

In our code, we make use of sklearn utility called GridSearchCV to perform cross-validation. This internally used stratified K-fold cross validation. GridSearchCV takes in a bunch of classifier parameter categories to test upon. Once the cross validation is completed, the GridSearchCV gives out the best parameters found during the cross validation process.

7 Performance analysis metrics

7.1 F-score

F-score is one of the important measures to understand the performance of a classifier. To understand this, we must first understand the 4 types of predictions that can occur.

- True positive: Data points correctly classified as positive
- True negative: Data points correctly classified as negative
- False positive: Data points which are negative in truth but are wrongly classified as positive.
- False negative: Data points which are positive in truth but are wrongly classified as negative.

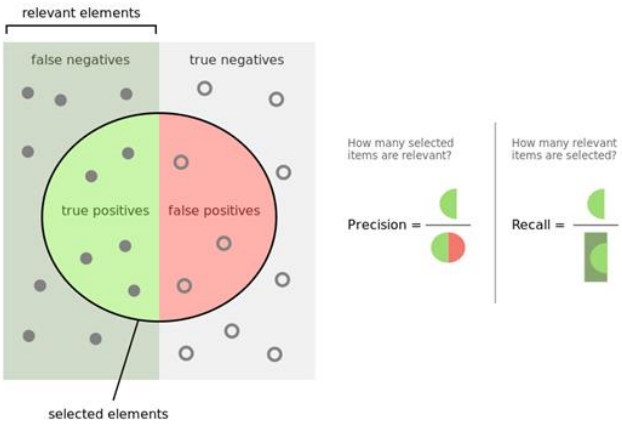


Figure 25: Precision and Recall Computation

The above figure (Source: [14]) shows how precision and Recall are computed. Mathematically, it can be given as:

$$Precision = \frac{True\ Postive}{True\ Positive + False\ Positive}$$

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

$$F - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

In essence, F-score is the harmonic mean of precision and recall. Thus, when a classifier is created, there is a compromise between precision and recall. It is very difficult to compare a model with a high recall and low precision versus a model with high precision and low recall. It is then f-score that can be used to understand the performance of the model.

In python, we use 'weighted' f-score to adapt to the imbalanced data-set.

Function Call: sklearn.metrics.f1_score(y_test, y_pred, average='weighted')

7.2 Classification Report

This builds a text report with the metrics of precision, recall and f-score for individual class. This is quite useful in understanding how the f-score varies if the data is imbalanced. An example of a classification report is shown below.

```
Classification Report
             precision    recall  f1-score   support

          0       0.95      0.90      0.92       895
          1       0.36      0.53      0.43        96

avg / total       0.89      0.86      0.88       991
```

Figure 26: Example of Classification Report output

Function Call: sklearn.metrics.classification_report(y_test, y_pred)

7.3 Confusion Matrix

Confusion matrix is one of the most popular means to understand the performance of a classifier model. It is a very good way to visualize the four possible predictions discussed earlier. Confusion matrix provides a summary of the prediction results on a classification problem. The number of true predictions and false predictions are summarized with count values and broken down by each class. This is fundamental to the confusion matrix.

The confusion matrix portrays the ways in which the classification model is confused which predicting the classes. It helps in understanding the type of errors that the model is making [15].

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	**TP**	**FN**
Class 2 Actual	**FP**	**TN**

Figure 27: Confusion Matrix

Function Call: sklearn.metrics.confusion_matrix(y_test, y_pred)

7.4 Accuracy

Classification accuracy is one of the most primitive measure of the performance of any classification. Classification accuracy tells us how many predictions were correct out of total number of predictions. Mathematically this can be shown as:

$$Accuracy = \frac{Number\ of\ correct\ classified}{Total\ number\ of\ data\ points} \times 100\ \%$$

It is very important to understand the shortcoming of this measurement. The accuracy score performs very poorly if the dataset is imbalanced (like the bank dataset that we have here). This is because, say, if class 1 has 90% data points, and class 2 has remaining 10% data points. Then even if all the data points of class 2 are misclassified, we would still get an accuracy of 90% which implies fake goodness of the classifier.

Function Call: clf.score(X_test, y_test) or sklearn.metrics.accuracy_score(y_test, y_pred)

7.5 ROC plot and ROC-AUC score

Now that we have seen and understood how the confusion matrix is determined, we can go ahead and understand a few more terminologies required for fully making sense of the Receiver Operating Characteristic Curve (ROC Curve).

- True Positive Rate (TPR): This is also known as recall, which as was seen earlier can be defined as below. Intuitively, this metric represents the proportion of positive data are correctly classified as positive with respect to all the positive data points. In other words, higher the TPR, lower is count of positive data point we have misclassified [16].

$$TPR = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- False positive Rate (FPR): This is also known as fall-out, and is defined as below. This metric represents proportion of negative data that are misclassified as positive. Thus, higher the FPR, the more negative data points are misclassified [16].

$$FPR = \frac{False\ Positive}{False\ Positive + True\ Negative}$$

ROC Curve is plot of FPR vs TPR. This curve represents the worthiness of a classifier. Below figure (Source: [17]) shows how to identify the quality of a classifier based on its ROC Curve. We can observe that more the curve tends towards the top-left corner, the better the classifier is.

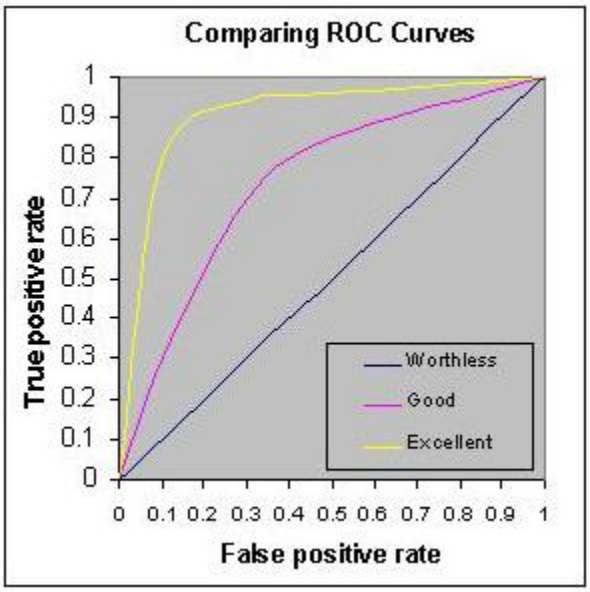


Figure 28: ROC Curves

The area under the ROC curve is an accuracy measure for the ROC. An area of 1 means the classifier test is perfect, and an area of 0.5 means the classifier is worthless and the predictions are essentially random in nature. A rough estimation of understanding the area under a ROC is [17]:

- 0.90 – 1 : Excellent classifier
- 0.80 – 0.90 : Good classifier
- 0.70 – 0.80 : Fair/Average classifier
- 0.60 – 0.70 : Poor classifier
- 0.50 – 0.60 : Worthless/fail classifier

8 Classifiers Explored

8.1 K Nearest Neighbors Classifier

8.1.1 Description

- K Nearest Neighbors or also abbreviated as KNN is one of the most primitive non-parametric learning algorithm. Here, the non-parametric implies that this classifier does make any assumptions of the underlying distribution of the data samples.
- This property of making no assumptions of the underlying probability distribution makes KNN one of the first approaches to try on an unknown dataset, that is where there is no prior knowledge about its distribution.
- KNN is also a class of "lazy" algorithm wherein there is no explicit training phase. This results in the least training time. However, all the training data will be required when testing. This consequences to a high testing time. This is not ideal.
- KNN works based on feature similarity. The closeness of any arbitrary point to samples of training data of any class determines which class this arbitrary point belong to.

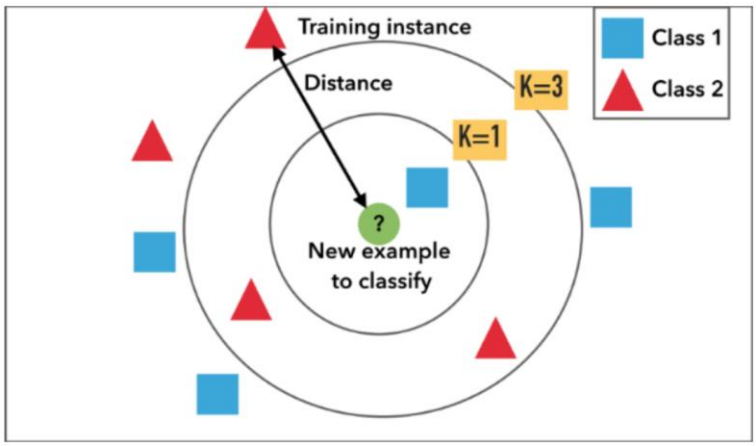


Figure 29: KNN classification

- In the above figure (Source: [1]), when K=1, the sample will be classified as class 1 as there is only 1 training point of class 1 in the region. When K=3, then the sample will be classified as class 2 as there are more data points of class 2 compared to class 1.
- Decision rule for discriminative model of KNN is given as:

$$K_n^{(i)} > K_n^{(j)} \quad : \forall j \neq i$$
$$x \in S_i$$

8.1.2 Results

KNN

```
 Time to train:  0.01282  seconds

 Time to test:  0.06577  seconds

Confusion Matrix
     0   1
0  345  11
1   28  13

Training Accuracy:  92.03 %

Testing Accuracy:  90.18 %

F1 score is:  0.89006

Accuracy score:  90.18 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.97      0.95       356
          1       0.54      0.32      0.40        41

avg / total       0.89      0.90      0.89       397


Area Under the Receiver Operating Characteristic Curve:
 0.64309
```

Figure 30: Results for KNN classifier

8.2 Support Vector Machines Classifier

8.2.1 Description

- Support Vector Machines or SVMs are motivated from the concept of a linear classifier with margins. However, SVMs rely on preprocessing the data to represent patterns in a higher dimension, and with an appropriate non-linear mapping, the data from the two classes can be separated by a hyperplane.
- Let's look at an example to understand a bit more about the concept of SVM. In below figure (Source [2]), (a) shows the how the actual data points look in feature space. As we can clearly observe the two classes are not linearly separable in any possible way. Now if we apply SVM on this data, it will essentially augment the data into a higher dimension such the data points in the higher dimension become linearly separable. This is shown in (b).

Figure 31: Example of SVM advantage

- Training a SVM main involves finding the optimal hyperplane, that is, the one with the maximum distance from the nearest training patterns. Below figure (Source: [3]), shows the support vectors are those (nearest) patterns, a distance b from the hyperplane. The three support vectors are shown in solid dots.

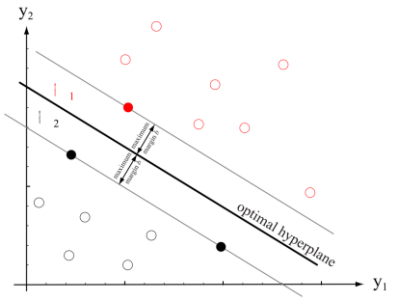


Figure 32: Support vector machine training

- Some common tuning parameters for SVM are:
  - **Kernel:** Kernels determine how the transformation of data points to the hyperplane will take place. Some common kernels are: Linear and Radial basis function.
  - **Regularization:** This parameter is an input to SVM to let know of the desired amount of how less the misclassification should be. This drastically affects the classification time.
  - **Gamma:** This parameter determines how many points to be considered within the margin of training.

8.2.2 Result

SVM

```
 Time to train:  0.30929  seconds

 Time to test:  0.02164  seconds

Confusion Matrix
     0   1
0  352   4
1   30  11

Training Accuracy:  89.85 %

Testing Accuracy:  91.44 %

F1 score is:  0.89599

Accuracy score:  91.44 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.99      0.95       356
          1       0.73      0.27      0.39        41

avg / total       0.90      0.91      0.90       397


Area Under the Receiver Operating Characteristic Curve:
 0.62853
```

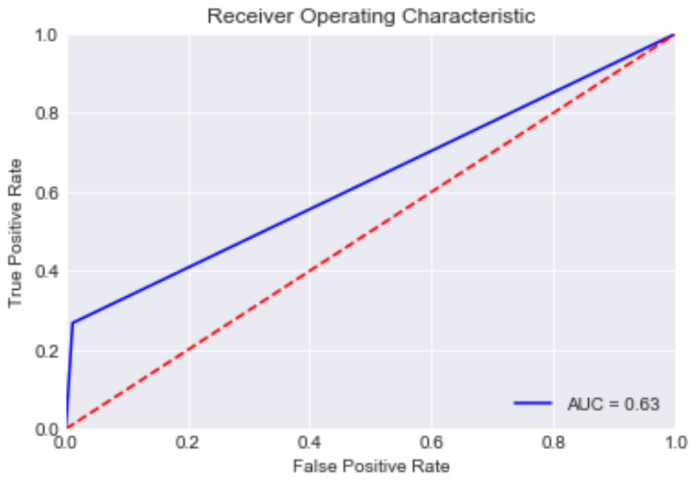


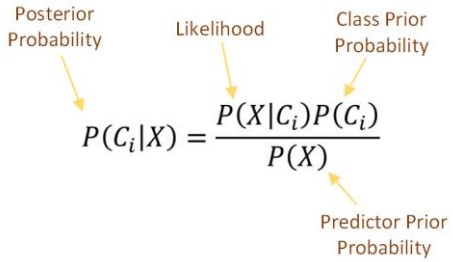
Figure 33: Results for SVM classifier

8.3 Naïve Bayesian Classifier

8.3.1 Description

- Naïve Bayesian classifier is based on the Bayes' Theorem and is highly used when dimension of input dataset is very high. Despite the simplicity of this classifier, it often outperforms other more complicated classifiers. [4]
- One of the key points about a Naïve Bayesian classifier is that it assumes that the features of the data set are independent. Hence, we use the word "naïve".
- Let's look at how the Naïve Bayesian works mathematically.

From Bayes' Theorem,



$$where\ X = \{x_1, x_2, x_3 \ldots . x_d\}$$

Therefore,

$$P(C_i|x_1, x_2, x_3 \ldots . x_d) \propto P(x_1, x_2, x_3 \ldots . x_d|C_i)P(C_i)$$

Since, Naïve Bayesian assumes that the conditional probabilities are independent. We can decompose the likelihood as:

$$P(X|C_i) \propto \prod_{k=1}^{d} P(x_k|C_i)$$

Thus, the posterior becomes:

$$P(C_i|X) \propto P(C_i) \prod_{k=1}^{d} P(x_k|C_i)$$

- One of the main advantage of using a Naïve Bayesian classifier is that it is very easy and fast to predict the class of any testing data.

8.3.2 Result

Naive Bayesian

```
 Time to train:  0.00596  seconds

 Time to test:  0.00033  seconds

Confusion Matrix
     0   1
0  323  33
1   19  22

Training Accuracy:  85.11 %

Testing Accuracy:  86.9 %

F1 score is:  0.87725

Accuracy score:  86.9 %

Classification Report
             precision    recall  f1-score   support

          0       0.94      0.91      0.93       356
          1       0.40      0.54      0.46        41

avg / total       0.89      0.87      0.88       397


Area Under the Receiver Operating Characteristic Curve:
 0.72194
```

Figure 34: Results for Naïve Bayesian Classifier

8.4 Multi-Layer Perceptron Classifier

8.4.1 Description

- A Multi-Layer Perceptron is a deep artificial neural network which is composed of more than one perceptron [5]. MLPs are composed of an input layer that receives the dataset and its corresponding set of features, an output layer to make the decision about the classification of the input data point, and in between these two layers, several hidden layers that act as the computational engine of the MLP. MLP with one hidden layer can approximate any continuous function. A sample MLP with one hidden layer is shown below. [6]

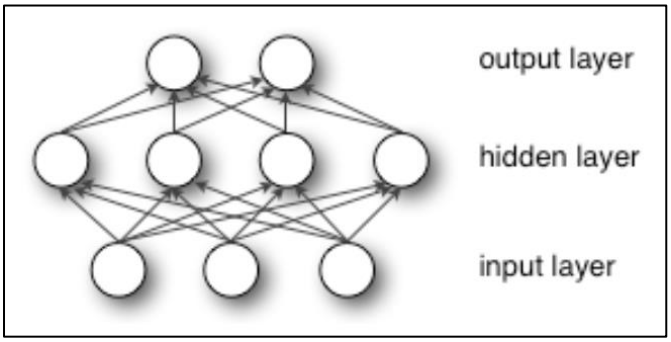


Figure 35: MLP with 1 hidden layer

- MLP are often applied to supervised learning problems. Here the MLP train on the training set and learns the relation between the output labels with their corresponding input features. This is done by adjusting the parameters, weights in order to reduce the loss function. Here, back propagation is used to compute the gradient, so that a stochastic gradient descent (SGD) algorithm can be applied to reduce the loss function. Below figure (Source [6]) shows the basic flow of SGD.

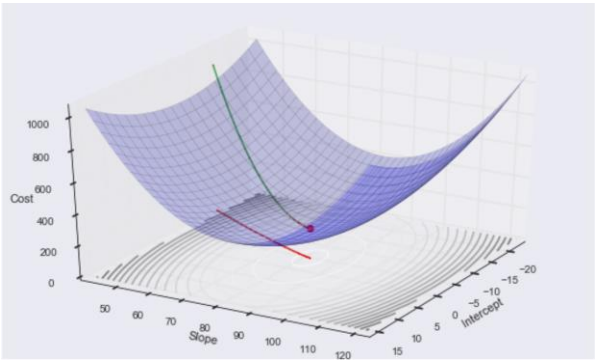


Figure 36: Use of Stochastic gradient descent to minimize cost function

- MLP is widely used by Industry leaders like Google, Microsoft. Microsoft adapted a 150 layer MLP (ResNet) to win the recent ImageNet challenge.

8.4.2 Result

```
MLP

 Time to train:  2.04912  seconds

 Time to test:  0.001  seconds

Confusion Matrix
     0   1
0  339  17
1   27  14

Training Accuracy:  99.02 %

Testing Accuracy:  88.92 %

F1 score is:  0.88224

Accuracy score:  88.92 %

Classification Report
             precision    recall  f1-score   support

          0       0.93      0.95      0.94       356
          1       0.45      0.34      0.39        41

avg / total       0.88      0.89      0.88       397


Area Under the Receiver Operating Characteristic Curve:
 0.64686
```

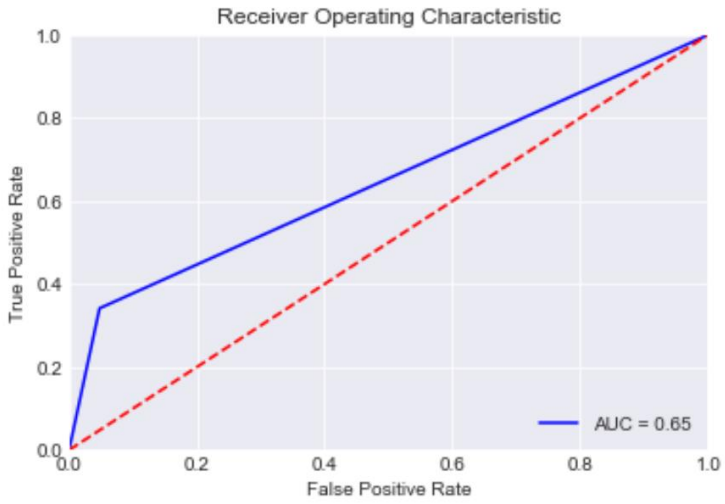


Figure 37: Results for MLP Classifier

8.5 Decision Tree Classifier

8.5.1 Description

- The Decision Tree (DT) Classifier is an easy to understand, non-parametric supervised and widely used machine learning algorithm for classification purposes. The DT Classifier uses tree representation to solve the classification problem. Here, each internal node of the tree represents the attributes and each leaf node represent the class label.
- To visualize how the decision tree solves the classification problem on feature space graph, observe below image (Source: [7]). Here the DT Classifier repeatedly divides the working area into sub-parts by identifying lines.

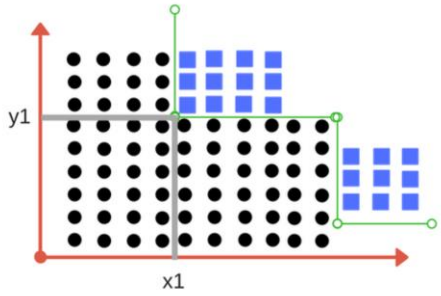


Figure 38: Decision Tree classifier visualization on feature space plot

- Some important terms related to DT classifiers are:
  - **Impurity:** Impurity is scenario when two classes are not separable based on the current set of features. This can be used as trick to fasten the process by selected lesser features.
  - **Entropy:** Shannon's entropy represents the degree of randomness or impurity. This tells about the predictability of a certain event (attribute). If the entropy is 0, then there is no randomness. It is given by [8]:

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

  - **Information Gain:** Information gain gives us an idea about which feature to select for dividing the working set. The feature selected based on information gain has the lowest impurity. Mathematically,

$$IG(S, A) = H(S) - \sum_{i=0}^{n} p(x) H(x)$$

Where $IG(S, A)$ is the information gain on applying feature A. and $p(x)$ is the probability of attribute 'x'.

8.5.2 Result

Decision Tree

```
 Time to train:  0.0067  seconds

 Time to test:  0.00027  seconds

Confusion Matrix
     0   1
0  353   3
1   30  11

Training Accuracy:  90.07 %

Testing Accuracy:  91.69 %

F1 score is:  0.89799

Accuracy score:  91.69 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.99      0.96       356
          1       0.79      0.27      0.40        41

avg / total       0.91      0.92      0.90       397


Area Under the Receiver Operating Characteristic Curve:
 0.62993
```

Figure 39: Results for Decision Tree Classifier

8.6 Random Forest Classifier

8.6.1 Description

- The Decision Tree Classifier we looked above forms the underlying concept for the Random Forest Classifier.
- Random forest classifier created a set of decision trees by a selecting a random set from the training data set. It then merges the results from different decision tree to create a consolidated final class for the test data.
- The random-forest algorithm brings extra randomness into the model when it is growing the trees. Here the random forest classifiers searches for the best feature among a subset of features creating a wide diversity which results in a better model.
- The parameters for a random forest divided into two categories:
    - To increase the prediction
        - **n_estimators:** Number of trees the algorithm builds before averaging. Higher number increases prediction accuracy but badly reduces the speed.
        - **max_features:** Maximum number of features the classifier is allowed to play with in a particular tree.
        - **min_sample_leaf:** This determines the number of leafs (classes).
    - To increase the computation speed
        - **n_jobs:** Tells the engine how many CPU cores to use.

- Random Forest is generally considered as an easy to use algorithm as most of its default hyperparameters often produce a good prediction result. Also, we must note that the number of hyperparameters are not too high and can configured quite easily.
- The main limitation when it comes to using Random forest classifier is that the large number of tree can drastically reduce the computation time making them useless in a real-time scenario. It is important to understand that these algorithms are generally fast to train. However, the prediction time can be very slow once they are trained, leading to its less use in real-time use cases.

8.6.2 Result

Random Forest

```
 Time to train:  0.04126  seconds

 Time to test:  0.00178  seconds

Confusion Matrix
     0   1
0  347   9
1   25  16

Training Accuracy:  98.18 %

Testing Accuracy:  91.44 %

F1 score is:  0.90492

Accuracy score:  91.44 %

Classification Report
             precision    recall  f1-score   support

          0       0.93      0.97      0.95       356
          1       0.64      0.39      0.48        41

avg / total       0.90      0.91      0.90       397


Area Under the Receiver Operating Characteristic Curve:
 0.68248
```

Figure 40: Results for Random Forest Classifier

8.7 Logistic Regression Classifier

8.7.1 Description

- Logistic regression classifier assigns observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.
- Logistic regression makes use of Sigmoid activation to map any real value to a value between 0 and 1. The sigmoid function is given as:

$$S(z) = \frac{1}{1 + e^{-z}}$$

- The decision boundary for logistic regression can be given as:

$$p \underset{S_2}{\overset{S_1}{\gtrless}} 0.5$$

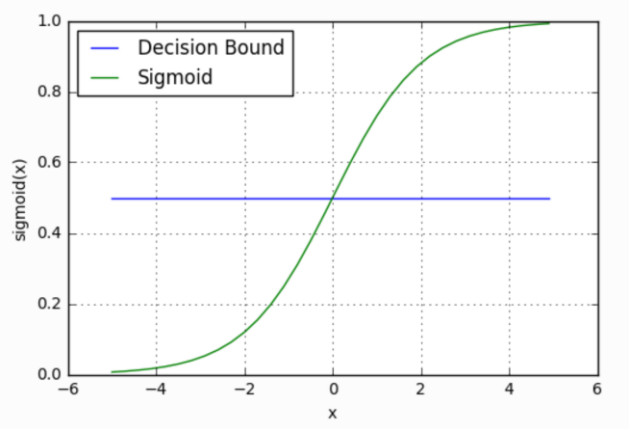


Figure 41: Decision Boundary for Logistic Regression

- Logistic Regression uses Cross-Entropy as the loss function and uses stochastic gradient descent to find the optimal parameters.
- Feature selection plays a key role in logistic regression as selecting the best features plays a major role in reducing computation time for logistic regression classifier.

8.7.2 Result

Logistic Regression

```
 Time to train:  0.02655  seconds

 Time to test:  0.00025  seconds

Confusion Matrix
     0   1
0  350   6
1   29  12

Training Accuracy:  90.1 %

Testing Accuracy:  91.18 %

F1 score is:  0.89603

Accuracy score:  91.18 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.98      0.95       356
          1       0.67      0.29      0.41        41

avg / total       0.90      0.91      0.90       397


Area Under the Receiver Operating Characteristic Curve:
 0.63791
```

Figure 42: Results for Logistic Regression Classifier

9 Other Tricks to improve performance

9.1 Reducing Dimensionality with PCA

9.1.1 Results with Random Forest Classifier

```
Random Forest

 Time to train:  0.06059  seconds

 Time to test:  0.0019  seconds

Confusion Matrix
     0   1
0  343  13
1   30  11

Training Accuracy:  98.23 %

Testing Accuracy:  89.17 %

F1 score is:  0.87879

Accuracy score:  89.17 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.96      0.94       356
          1       0.46      0.27      0.34        41

avg / total       0.87      0.89      0.88       397


Area Under the Receiver Operating Characteristic Curve:
 0.61589
```

Figure 43: Results for Random Forest (PCA) Classifier

9.1.2 Results with Naïve Bayesian Classifier

```
Naive Bayesian (PCA)

 Time to train:  0.00285  seconds

 Time to test:  0.00022  seconds

Confusion Matrix
     0   1
0  343  13
1   31  10

Training Accuracy:  87.85 %

Testing Accuracy:  88.92 %

F1 score is:  0.87495

Accuracy score:  88.92 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.96      0.94       356
          1       0.43      0.24      0.31        41

avg / total       0.87      0.89      0.87       397


Area Under the Receiver Operating Characteristic Curve:
 0.60369
```

Figure 44: Results for Naive Bayesian (PCA) Classifier

9.1.3 Results with Multi-Layer Perceptron Classifier

```
MLP (PCA)

 Time to train:  0.82884  seconds

 Time to test:  0.0006  seconds

Confusion Matrix
     0  1
0  350  6
1   33  8

Training Accuracy:  89.42 %

Testing Accuracy:  90.18 %

F1 score is:  0.87945

Accuracy score:  90.18 %

Classification Report
             precision    recall  f1-score   support

          0       0.91      0.98      0.95       356
          1       0.57      0.20      0.29        41

avg / total       0.88      0.90      0.88       397


Area Under the Receiver Operating Characteristic Curve:
 0.58913
```

Figure 45: Results for MLP (PCA) Classifier

9.2 Select good features using “Select K Best”

9.2.1 Results with SVM Classifier

```
SVM (SelectKBest)

 Time to train:  0.06067  seconds

 Time to test:  0.00652  seconds

Confusion Matrix
     0   1
0  352   4
1   30  11

Training Accuracy:  89.85 %

Testing Accuracy:  91.44 %

F1 score is:  0.89599

Accuracy score:  91.44 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.99      0.95       356
          1       0.73      0.27      0.39        41

avg / total       0.90      0.91      0.90       397


Area Under the Receiver Operating Characteristic Curve:
 0.62853
```

Figure 46: Results for SVM (SelectKBest) Classifier

9.2.2 Results with Naïve Bayesian Classifier

```
Naive Bayesian (SelectKBest)

 Time to train:  0.00217  seconds

 Time to test:  0.00017  seconds

Confusion Matrix
     0   1
0  321  35
1   21  20

Training Accuracy:  83.93 %

Testing Accuracy:  85.89 %

F1 score is:  0.86781

Accuracy score:  85.89 %

Classification Report
             precision    recall  f1-score   support

          0       0.94      0.90      0.92       356
          1       0.36      0.49      0.42        41

avg / total       0.88      0.86      0.87       397


Area Under the Receiver Operating Characteristic Curve:
 0.69475
```

Figure 47: Results for Naive Bayesian (SelectKBest) Classifier

9.2.3 Results with Multi-Layer Perceptron Classifier

```
MLP (SelectKBest)

 Time to train:  0.55113  seconds

 Time to test:  0.00054  seconds

Confusion Matrix
     0   1
0  348   8
1   23  18

Training Accuracy:  89.96 %

Testing Accuracy:  92.19 %

F1 score is:  0.91398

Accuracy score:  92.19 %

Classification Report
             precision    recall  f1-score   support

          0       0.94      0.98      0.96       356
          1       0.69      0.44      0.54        41

avg / total       0.91      0.92      0.91       397


Area Under the Receiver Operating Characteristic Curve:
 0.70828
```

Figure 48: Results for MLP (SelectKBest) Classifier

9.2.4 Results with Random Forest Classifier

Random Forest (SelectKBest)

```
 Time to train:  0.03094  seconds

 Time to test:  0.00147  seconds

Confusion Matrix
     0   1
0  347   9
1   20  21

Training Accuracy:  93.49 %

Testing Accuracy:  92.7 %

F1 score is:  0.92185

Accuracy score:  92.7 %

Classification Report
             precision    recall  f1-score   support

          0       0.95      0.97      0.96       356
          1       0.70      0.51      0.59        41

avg / total       0.92      0.93      0.92       397


Area Under the Receiver Operating Characteristic Curve:
 0.74346
```

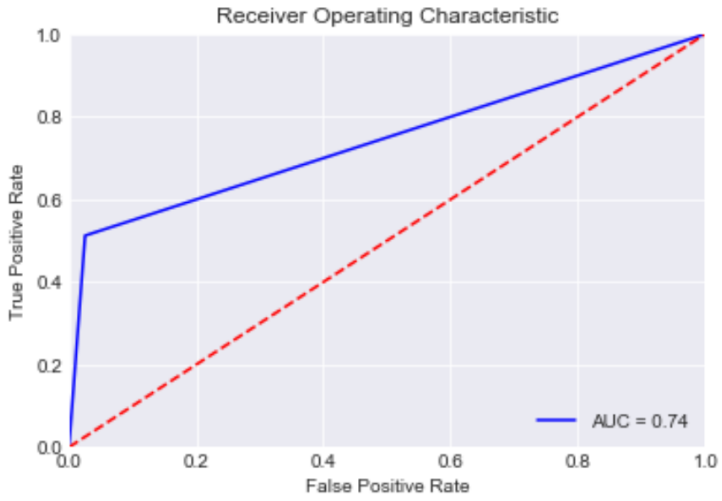


Figure 49: Results for Random Forest (SelectKBest) Classifier

9.3 Data oversampling with SMOTE

9.3.1 Results with SVM Classifier

```
SVM (SMOTE)

 Time to train:  3.05562  seconds

 Time to test:  0.03293  seconds

Confusion Matrix
     0   1
0  322  34
1   15  26

Training Accuracy:  86.1 %

Testing Accuracy:  87.66 %

F1 score is:  0.88649

Accuracy score:  87.66 %

Classification Report
             precision    recall  f1-score   support

          0       0.96      0.90      0.93       356
          1       0.43      0.63      0.51        41

avg / total       0.90      0.88      0.89       397


Area Under the Receiver Operating Characteristic Curve:
 0.76932
```

Figure 50: Results for SVM (SMOTE) Classifier

9.3.2 Results with Naïve Bayesian Classifier

Naive Bayesian (SMOTE)

```
 Time to train:  0.01177  seconds

 Time to test:  0.00036  seconds

Confusion Matrix
     0   1
0  325  31
1   18  23

Training Accuracy:  82.68 %

Testing Accuracy:  87.66 %

F1 score is:  0.88387

Accuracy score:  87.66 %

Classification Report
             precision    recall  f1-score   support

          0       0.95      0.91      0.93       356
          1       0.43      0.56      0.48        41

avg / total       0.89      0.88      0.88       397


Area Under the Receiver Operating Characteristic Curve:
 0.73695
```

Figure 51: Results for Naive Bayesian (SMOTE) Classifier

9.3.3 Results with Multi-Layer Perceptron Classifier

```
MLP (SMOTE)

 Time to train:  2.25731  seconds

 Time to test:  0.00087  seconds

Confusion Matrix
     0   1
0  336  20
1   23  18

Training Accuracy:  99.04 %

Testing Accuracy:  89.17 %

F1 score is:  0.88986

Accuracy score:  89.17 %

Classification Report
             precision    recall  f1-score   support

          0       0.94      0.94      0.94       356
          1       0.47      0.44      0.46        41

avg / total       0.89      0.89      0.89       397


Area Under the Receiver Operating Characteristic Curve:
 0.69142
```

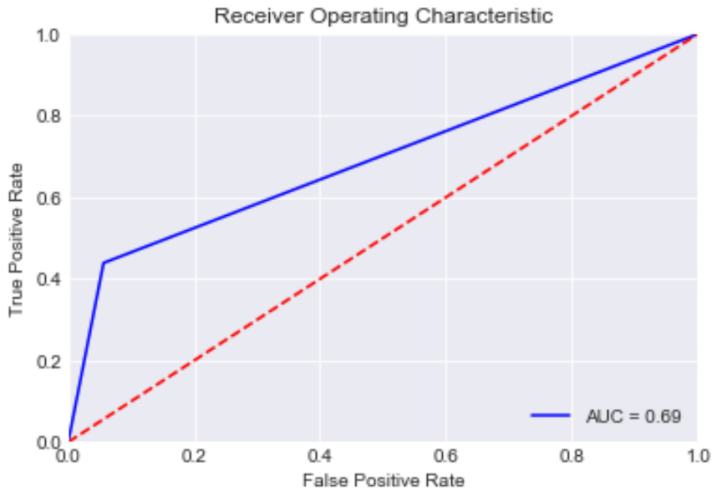


Figure 52: Results for MLP (SMOTE) Classifier

9.3.4 Results with Random Forest Classifier

Random Forest (SMOTE)

```
 Time to train:  0.06294  seconds

 Time to test:  0.00135  seconds

Confusion Matrix
     0   1
0  343  13
1   22  19

Training Accuracy:  99.38 %

Testing Accuracy:  91.18 %

F1 score is:  0.90695

Accuracy score:  91.18 %

Classification Report
             precision    recall  f1-score   support

          0       0.94      0.96      0.95       356
          1       0.59      0.46      0.52        41

avg / total       0.90      0.91      0.91       397


Area Under the Receiver Operating Characteristic Curve:
 0.71345
```

Figure 53: Results for Random Forest (SMOTE) Classifier

9.4 SelectKBest followed by data oversampling and SMOTE

9.4.1 Results with SVM Classifier

SVM (SelectKBest + SMOTE)

```
 Time to train:  0.84211  seconds

 Time to test:  0.00958  seconds

Confusion Matrix
     0   1
0  324  32
1   16  25

Training Accuracy:  85.36 %

Testing Accuracy:  87.91 %

F1 score is:  0.88757

Accuracy score:  87.91 %

Classification Report
             precision    recall  f1-score   support

          0       0.95      0.91      0.93       356
          1       0.44      0.61      0.51        41

avg / total       0.90      0.88      0.89       397


Area Under the Receiver Operating Characteristic Curve:
 0.75993
```

Figure 54: Results for SVM (SelectKBest + SMOTE) Classifier

9.4.2 Results with Naïve Bayesian Classifier

Naive Bayesian (SelectKBest + SMOTE)

```
 Time to train:  0.0032  seconds

 Time to test:  0.00016  seconds

Confusion Matrix
     0   1
0  320  36
1   20  21

Training Accuracy:  71.85 %

Testing Accuracy:  85.89 %

F1 score is:  0.86884

Accuracy score:  85.89 %

Classification Report
             precision    recall  f1-score   support

          0       0.94      0.90      0.92       356
          1       0.37      0.51      0.43        41

avg / total       0.88      0.86      0.87       397


Area Under the Receiver Operating Characteristic Curve:
 0.70554
```

Figure 55: Results for Naive Bayesian (SelectKBest + SMOTE) Classifier

9.4.3 Results with Multi-Layer Perceptron Classifier

```
MLP (SelectKBest + SMOTE)

 Time to train:  1.07854  seconds

 Time to test:  0.00055  seconds

Confusion Matrix
     0   1
0  330  26
1   16  25

Training Accuracy:  86.24 %

Testing Accuracy:  89.42 %

F1 score is:  0.8992

Accuracy score:  89.42 %

Classification Report
             precision    recall  f1-score   support

          0       0.95      0.93      0.94       356
          1       0.49      0.61      0.54        41

avg / total       0.91      0.89      0.90       397


Area Under the Receiver Operating Characteristic Curve:
 0.76836
```

Figure 56: Results for MLP (SelectKBest + SMOTE) Classifier

9.4.4 Results with Random Forest Classifier

Random Forest (SelectKBest + SMOTE)

```
 Time to train:  0.03125  seconds

 Time to test:  0.00128  seconds

Confusion Matrix
     0   1
0  342  14
1   19  22

Training Accuracy:  92.96 %

Testing Accuracy:  91.69 %

F1 score is:  0.91447

Accuracy score:  91.69 %

Classification Report
             precision    recall  f1-score   support

          0       0.95      0.96      0.95       356
          1       0.61      0.54      0.57        41

avg / total       0.91      0.92      0.91       397


Area Under the Receiver Operating Characteristic Curve:
 0.74863
```

Figure 57: Results for Random Forest (SelectKBest + SMOTE) Classifier

9.5 Using Recursive Feature Extraction with Logistic Regression

Logistic Regression (RFE)

```
 Time to train:  0.00664  seconds

 Time to test:  0.00017  seconds

Confusion Matrix
     0   1
0  351   5
1   30  11

Training Accuracy:  90.07 %

Testing Accuracy:  91.18 %

F1 score is:  0.894

Accuracy score:  91.18 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.99      0.95       356
          1       0.69      0.27      0.39        41

avg / total       0.90      0.91      0.89       397


Area Under the Receiver Operating Characteristic Curve:
 0.62712
```

Figure 58: Results for Logistic Regression (RFE) Classifier

9.6 Efforts to increase f-score of minority class

As discussed above, since the dataset is highly imbalance, we observe a low f1-score for the minority class. Several experiments were performed as mentioned earlier. The results are:

9.6.1 Balanced test set, SelectKBest feature selection and SMOTE on training data

9.6.1.1 Results with SVM Classifier

```
SVM (SelectKBest + Balanced Test + SMOTE)

 Time to train:  0.9932  seconds

 Time to test:  0.00732  seconds

Confusion Matrix
     0   1
0  119  13
1   59  73

Training Accuracy:  81.06 %

Testing Accuracy:  72.73 %

F1 score is:  0.71873

Accuracy score:  72.73 %

Classification Report
             precision    recall  f1-score   support

          0       0.67      0.90      0.77       132
          1       0.85      0.55      0.67       132

avg / total       0.76      0.73      0.72       264


Area Under the Receiver Operating Characteristic Curve:
 0.72727
```

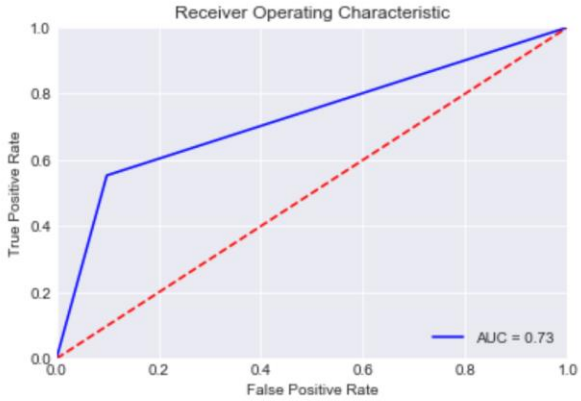


Figure 59: Results for SVM (SelectKBest + Balanced Test + SMOTE) Classifier

9.6.1.2 Results with Naïve Bayesian Classifier

Naive Bayesian (SelectKBest + Balanced Test + SMOTE)

```
 Time to train:  0.00413  seconds

 Time to test:  0.00019  seconds

Confusion Matrix
     0   1
0  117  15
1   68  64

Training Accuracy:  75.36 %

Testing Accuracy:  68.56 %

F1 score is:  0.6724

Accuracy score:  68.56 %

Classification Report
             precision    recall  f1-score   support

          0       0.63      0.89      0.74       132
          1       0.81      0.48      0.61       132

avg / total       0.72      0.69      0.67       264


Area Under the Receiver Operating Characteristic Curve:
 0.68561
```

Figure 60: Results for Naive Bayesian (SelectKBest + Balanced Test + SMOTE) Classifier

9.6.1.3 Results with Multi-Layer Perceptron Classifier

MLP (SelectKBest + Balanced Test + SMOTE)

```
 Time to train:  0.46626  seconds

 Time to test:  0.00051  seconds

Confusion Matrix
    0    1
0  89   43
1  28  104

Training Accuracy:  74.3 %

Testing Accuracy:  73.11 %

F1 score is:  0.73019

Accuracy score:  73.11 %

Classification Report
             precision    recall  f1-score   support

          0       0.76      0.67      0.71       132
          1       0.71      0.79      0.75       132

avg / total       0.73      0.73      0.73       264


Area Under the Receiver Operating Characteristic Curve:
 0.73106
```

Figure 61: Results for MLP (SelectKBest + Balanced Test + SMOTE) Classifier

9.6.1.4 Results with Random Forest Classifier

Random Forest (SelectKBest + Balanced Test + SMOTE)

```
 Time to train:  0.0327  seconds

 Time to test:  0.00114  seconds

Confusion Matrix
     0   1
0  127   5
1   79  53

Training Accuracy:  92.34 %

Testing Accuracy:  68.18 %

F1 score is:  0.65469

Accuracy score:  68.18 %

Classification Report
             precision    recall  f1-score   support

          0       0.62      0.96      0.75       132
          1       0.91      0.40      0.56       132

avg / total       0.77      0.68      0.65       264


Area Under the Receiver Operating Characteristic Curve:
 0.68182
```

Figure 62: Results for Random Forest (SelectKBest + Balanced Test + SMOTE) Classifier

9.6.2 Changing SVM Class Weights

SVM (Class Weight Balanced)

```
 Time to train:  2.34452  seconds

 Time to test:  0.03529  seconds

Confusion Matrix
     0   1
0  307  49
1   11  30

Training Accuracy:  83.53 %

Testing Accuracy:  84.89 %

F1 score is:  0.86854

Accuracy score:  84.89 %

Classification Report
             precision    recall  f1-score   support

          0       0.97      0.86      0.91       356
          1       0.38      0.73      0.50        41

avg / total       0.90      0.85      0.87       397


Area Under the Receiver Operating Characteristic Curve:
 0.79703
```

Figure 63: Results for SVM (Class Weight Balanced) Classifier

9.7 Cross Validation using GridSearchCV

9.7.1 Result for KNN Classifier

```
KNN best k:  4
kNN

 Time to train:  161.45995  seconds

 Time to test:  0.06652  seconds

Confusion Matrix
     0  1
0  348  8
1   33  8

Training Accuracy:  90.83 %

Testing Accuracy:  89.67 %

F1 score is:  0.87583

Accuracy score:  89.67 %

Classification Report
             precision    recall  f1-score   support

          0       0.91      0.98      0.94       356
          1       0.50      0.20      0.28        41

avg / total       0.87      0.90      0.88       397


Area Under the Receiver Operating Characteristic Curve:
 0.58633
```

Figure 64: Results for kNN (GridSearchCV) Classifier

9.7.2 Result for SVM Classifier

```
[SVM] Best C: 100
[SVM] Best Kernel: rbf
[SVM] Best Gamma: 0.001
SVM (GridSearchCV)

 Time to train:  823.15373  seconds

 Time to test:  0.03189  seconds

Confusion Matrix
     0   1
0  352   4
1   30  11

Training Accuracy:  89.85 %

Testing Accuracy:  91.44 %

F1 score is:  0.89599

Accuracy score:  91.44 %

Classification Report
             precision    recall  f1-score   support

          0       0.92      0.99      0.95       356
          1       0.73      0.27      0.39        41

avg / total       0.90      0.91      0.90       397


Area Under the Receiver Operating Characteristic Curve:
 0.62853
```

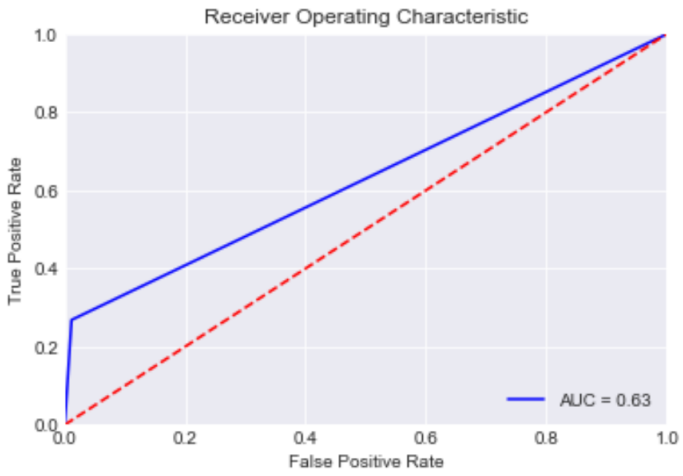


Figure 65: Results for SVM (GridSearchCV) Classifier

10 Usage of datasets

Below is the description of how the dataset was used.

- Training Data: The cleaned and normalized dataset is split such that 90% of data points correspond to the training dataset. Following operations were performed while trying to increase the performance.
    - For PCA: The training dataset was reduced to 2 principal components.
    - For SelectKBest: 11 best features were selected from the 57 total features (57 features were generated after one-hot key encoding of categorical strings)
    - For SMOTE: The training dataset was oversampled to contain same number of data points in majority and minority class.

- Testing Data: The cleaned and normalized dataset is split such that 10% of data points correspond to the testing dataset. Following operations were performed while trying to increase the performance.
    - For PCA: The testing dataset was reduced to 2 principal components.
    - For SelectKBest: 11 best features were selected from the 57 total features (57 features were generated after one-hot key encoding of categorical strings)
    - For SMOTE: No oversampling was performed on the training data set.
    - Balanced dataset experiment: During the balanced test dataset experiment, a balanced subset of the complete dataset was extracted for testing. Rest of the points were assigned to be part of training dataset.

- Validation set and Cross Validation: For purposes of stratified K-Fold cross validation, GridSearchCV was used. This internally segregates the training dataset into K-Folds with equal distribution and computes the average accuracy across all folds. Each of these operations occur for each parameter that we pass to test upon. Finally, the GridSearchCV algorithm return the best parameters.

11 Understanding the Results

Below is a summary of all the results.

Summary of Results						
Classifier	Training Time (in seconds)	Training Score	Testing Time (in seconds)	Testing Score	ROC-AUC	F1-score
KNN	0.01282	92.03%	0.06577	90.18%	0.64309	0.89006
SVM	0.30929	89.85%	0.02164	91.44%	0.62853	0.89599
Naive Bayesian	0.00596	85.11%	0.00033	86.90%	0.72194	0.87725
MLP	2.04912	99.02%	0.001	88.92%	0.64686	0.88224
Decision Tree	0.0067	90.07%	0.00027	91.69%	0.62993	0.89799
Random Forest	0.04126	98.18%	0.00178	91.44%	0.68248	0.90492
Logistic Regression	0.02655	90.10%	0.00025	91.18%	0.63791	0.89603
Random Forest (PCA)	0.06059	98.23%	0.0019	89.17%	0.61589	0.87879
Naive Bayesian (PCA)	0.00285	87.85%	0.00022	88.92%	0.60369	0.87495
MLP (PCA)	0.82884	89.42%	0.0006	90.18%	0.58913	0.87945
SVM (SelectKBest)	0.06067	89.85%	0.00652	91.44%	0.62853	0.89599
Naive Bayesian (SelectKBest)	0.00217	83.93%	0.00017	85.89%	0.69475	0.86781
MLP (SelectKBest)	0.55113	89.96%	0.00054	92.19%	0.70828	0.91398
Random Forest (SelectKBest)	0.03094	93.49%	0.00147	92.70%	0.74346	0.92185
SVM (SMOTE)	3.05562	86.10%	0.03293	87.66%	0.76932	0.88649
Naive Bayesian (SMOTE)	0.01177	82.68%	0.00036	87.66%	0.73695	0.88387
MLP (SMOTE)	2.25731	99.04%	0.00087	89.17%	0.69142	0.88986
Random Forest (SMOTE)	0.06294	99.38%	0.00135	91.18%	0.71345	0.90695
SVM (SelectKBest + SMOTE)	0.84211	85.36%	0.00958	87.91%	0.75993	0.88757
Naive Bayesian (SelectKBest + SMOTE)	0.0032	71.85%	0.00016	85.89%	0.70554	0.86884
MLP (SelectKBest + SMOTE)	1.07854	86.24%	0.00055	89.42%	0.76836	0.8992
Random Forest (SelectKBest + SMOTE)	0.03125	92.96%	0.00128	91.69%	0.74863	0.91447
Logistic Regression (RFE)	0.00664	90.07%	0.00017	91.18%	0.62712	0.894
SVM (SelectKBest + Balanced Test + SMOTE)	0.9932	81.06%	0.00732	72.73%	0.72727	0.71873
Naive Bayesian (SelectKBest + Balanced Test + SMOTE)	0.00413	75.36%	0.00019	68.56%	0.68561	0.6724
MLP (SelectKBest + Balanced Test + SMOTE)	0.46626	74.30%	0.00051	73.11%	0.73106	0.73019
Random Forest (SelectKBest + Balanced Test + SMOTE)	0.0327	92.34%	0.00114	68.18%	0.68182	0.65469
SVM (Class Weight Balanced)	2.34452	83.53%	0.03529	84.89%	0.79703	0.86854
kNN (GridSearchCV)	161.45995	90.83%	0.06652	89.67%	0.58633	0.87583
SVM (GridSearchCV)	823.15373	89.85%	0.03189	91.44%	0.62853	0.89599

Figure 66: Summary of Results

We can make some key conclusions from all the experiments we have performed in order to identify the best classifier.

- In general, MLP classifiers take a lot more time for training compared to other classifiers.
- SVM classifier with a “balanced” class weight seems to perform the best based on the area under ROC curve. The AUC here is 0.8 with below ROC

Figure 67: ROC curve for SVM with balanced class weights

- Keeping in view of f1-score for the minority class, we can choose MLP with a balanced dataset for testing. On top of this we also apply SelectKBest feature selection and SMOTE data oversampling on the training dataset. Here we get a f-score of 0.75 for the minority class. Below is the classification report for the same.

Classification Report				
	precision	recall	f1-score	support
0	0.76	0.67	0.71	132
1	0.71	0.79	0.75	132
avg / total	0.73	0.73	0.73	264

Figure 68: Classification Report for best f1-score of minority with MLP in experimental setup

- If a faster classifier is required, then Naïve Bayesian Classifier is observed to be one of the best. Its training and testing time is much lesser than other classifiers while maintain a good accuracy, f-score and AUROC.
- We can observe that PCA does not help things much, rather other feature selection technique: SelectKBest seems to be very fruitful. This could be because PCA considers only linear relationships between variables and completely ignored the possibility of a potential multivariate nature of the dataset variables.
- The worst classifier here can be credited to KNN due to the fact of its high testing time, low AUROC, average f1-score for minority class.

12 References

[1] https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

[2] https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. 2000. Pattern Classification (2nd Edition). Wiley-Interscience.

[4] http://www.statsoft.com/textbook/naive-bayes-classifier

[5] https://deeplearning4j.org/multilayerperceptron

[6] http://deeplearning.net/tutorial/mlp.html

[7] https://medium.com/machine-learning-101/chapter-3-decision-trees-theory-e7398adac567

[8] https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html

[9] http://inmachineswetrust.com/posts/building-logistic-regression/

[10] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[11] [Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

[12] https://stackoverflow.com/questions/45963174/what-is-f1-score-and-what-its-value-indicates

[13] EE 559 Paradigm and course flow chart S18

[14] Wikipedia contributors. Precision and recall. Wikipedia, The Free Encyclopedia. April 8, 2018, 13:11 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=835396495. Accessed May 1, 2018.

[15] https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

[16] https://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it

[17] http://gim.unmc.edu/dxtests/roc3.htm

[18] https://machinelearningmastery.com/feature-selection-machine-learning-python/