EE 559

MATHEMATICAL PATTERN RECOGNITION

FINAL PROJECT

DATE : MAY / 1 /2018

SUBMITTED BY
SIYAL SONARKAR
USC ID : 9965116981
USC EMAIL : sonarkar@usc.edu

FINAL PROJECT

Abstract :

Pattern Recognition focuses on the recognition of the pattern and regularities. It is mainly of two types : "supervised learning" which has labeled training data and "unsupervised learning" where no labels are available. The goal of this project is to develop a pattern recognition system which deals with the real world datasets. The dataset is unprocessed and can have categorical features and some might be unknown as well. In this project we are supposed to do preprocessing of the data, then extract important features and lastly, classify them using a classifier.

Main body :

DataSet : The dataset which I am using here is 'Bank Marketing Data Set'. This dataset contains the information about people which are targeted by the bank marketing campaign. The dataset contains 19 features and 1 label column. The data has categorical features and it have some missing data as well. For dealing with the missing data, we need to be careful while pre-processing.
The following operations are performed on the dataset :

Data Preprocessing :

In data preprocessing, we will be transforming the raw data into an understandable format. This is one of the most important task while developing a pattern recognition system. Here, we are dealing with a real world data and so it may have many incomplete or missing data and thats why this step is most crucial. Data preprocessing is achieved by performing following steps :

- **Identifying the 'unknown' :** In this dataset, there are some features which have unknown values. We cannot deal with the data which has unknown values in it. For this unknown value, we will have to find the proper value and fill it in its place. But for now we can just identify them and I have assigned them to be '0'. For replaying the 'unknown' value by '0', I used a panda data frame function. Now, we have all '0' in the place of all 'unknown'.

- **Labelling the data :** The data which we have have some categorical features as well. For this we have to label the data which is in categorical format. The LabelEncoder() from sklearn helps in labelling the data according to their alphabetical order and the digits are placed first. I applied this only to the categorical features and labelled the data. So the features having missing data are unknowns as '0' and we can easily fill in now as we know its label.

- **Separating the class label :** The preprocessing before this was done on the complete dataset. But now we have to separate the class label for further preprocessing. So the last feature is considered as the label and rest all are still the features of the dataset. It is also important to convert the datatype from 'DataFrame' to 'float' as it is easy to deal with the float type data and some functions don't accept the 'DataFrame' type data. So we changes the data type and then separated the class label.

- **Splitting the data :** Now, the dataset which we have after performing the above steps needs to be split into training and test data. For this, I am using an inbuilt sklearn function train_test_split( ). As mentioned in the problem, we have to keep atleast 10 % testing data. I considered 75% training data and 25% testing data from the dataset. So according to this the data was split into train and test data and the labels as well.

- **Fill the missing data :** It is important to fill the missing data in features otherwise the performance of the classifier reduces and it doesn't give good accuracy as well. So the filling the missing data, I considered the most frequently occurring strategy to find the best fit for the missing data in each feature. This means, each feature is checked for its own most frequently occurring data and this is fill in the missing data. The features : 'job', 'marital', 'education', 'default', 'housing' and 'loan' have missing data in it. So the data is filled in only these features and rest all features are ignored as they don't have any missing data. A 'imputer( )' function is used to find the most frequently occurring data in a feature and fit it to that feature and then transform it. The test data is also transformed.

- **Oversampling :** The data we have in each class is not balanced, that is, the data in one class is more than in the other class. This is actually doesn't give good results when trained the model with this data. So oversampling is used to generate new samples in the class which is under-represented. In this problem, I used the Synthetic Minority Oversampling Technique (SMOTE). This probably considered to increase the performance of the classifier.

- **One Hot Encoder :** The categorical features need to be converted to features that can be used with scikit-learn estimator. We can implement by using the one hot encoding technique. In this technique, each category of the feature is reforested in a binary format. And this is included as a feature to train the model. The one hot encoder can be of following format : [0 1 0] for any category. This is important as we will have all the features in a format which can be used to train the model. One hot encoder places an important role in binarising the categorical feature data.

- **Standardization :** The mean and the variance of the training data is found out using the StandardScaler( ) function. This value of mean and standard deviation is used to fit the train data and the training data is normalised using these values. It is useful to find the mean and standard deviation of the training data as it is used for normalising the test data as well. The test data is unknown to us and so we have to normalise the test data with respect to the train data. The formula is given as :

$$Z = \frac{X - \mu}{\sigma}$$

where Z is standardised value.

X is the data which is being standardised.

μ is the mean.

σ is the standard deviation.

<u>**Feature Selection and Feature Extraction :**</u>

Feature Selection is quite necessary as it is important in reducing the overfitting of the data and it simplifies the model. Feature selection is basically selecting the best features amongst all the features. It is a subset of the original feature. Feature selection is done when we have many features and comparatively less data. Whereas, feature extraction means completely new features are build using the original features. The dimension of the dataset is reduced using these methods. The most common is using PCA for feature selection and feature extraction. There is an inbuilt function for this. I used the PCA technique for this step.

The following figure 1 shows the plot of all the numerical data. The diagonal shows the histogram plot for each numeric feature. Now, when one numeric feature is compared with the other numeric feature, we get a scatter plot as seen in the non-diagonal boxes. The pink and yellow colour specifies each feature. From the plot, we can see that at some places the plot is highly correlated, while at some places it is lightly correlated. As the features in lightly correlated regions are discriminant, they have high discriminant power. We usually need uncorrelated or lightly correlated data, but as we can see both correlated and uncorrelated data is almost equally present, so we cannot eliminate either of them. Thats why, I used all the numerical features.

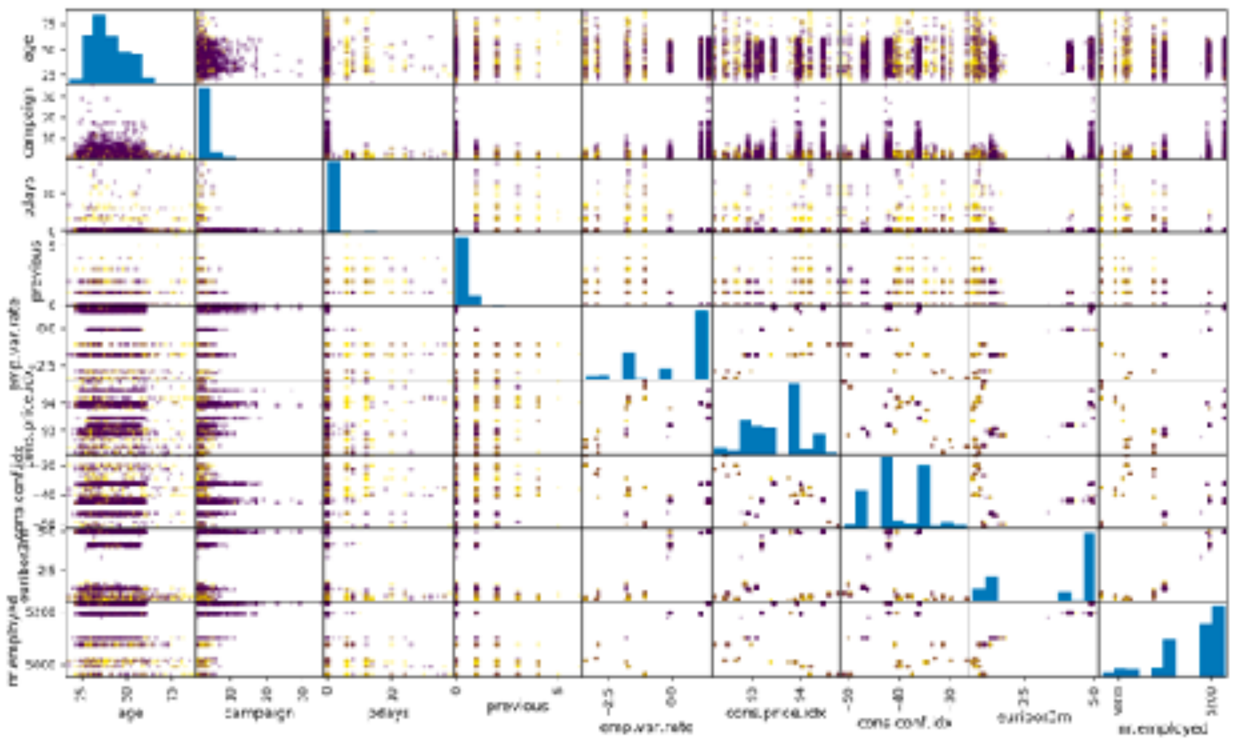


Figure 1 : Correlation plot for numerical data

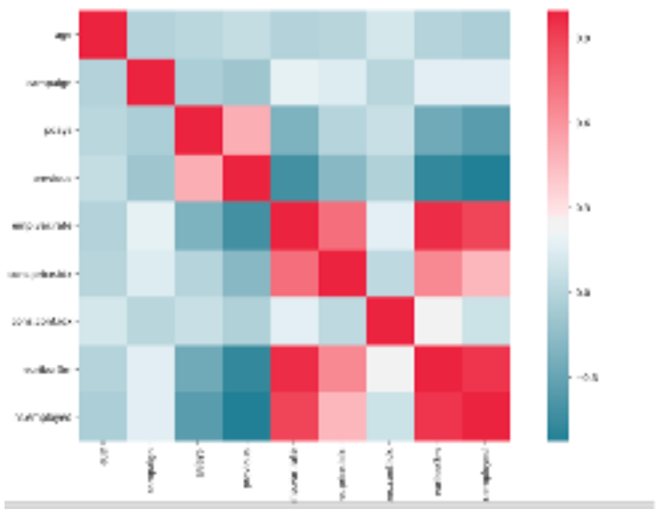


Figure 2 : Correlation plot for categorical data

In figure 2, we can see the same thing which we observed for the numerical data. The red indicates highly correlated and blue indicates uncorrelated or lightly correlated. As both type of features are there, I used all the categorical features as well.

In short, I used all the 19 features and then performed feature extraction.

<u>Classifiers :</u>

I used four classifiers and those are : SVM classifier, Naive Bayes' classifier, Random Forest and Perceptron. They are described below and there pattern recognition names are also given :

SVM classifier : This is a Support Vector Machine which is commonly known as SVM. The SVM classifier comes under supervised learning. As it is supervised learning, we know the labeled training data and the output of the algorithm is given by optimal hyperplane known as decision boundary which is used for categorizing new samples. Basically, it separates the classes by calculating a decision boundary and determining the region. SVM classifier have different kernels like linear, rbf, etc.

Python function name : SVC (support vector clustering)
model = sklearn.svm.SVC( )

In this problem, I have done cross-validation for the SVM classifier. In the cross validation method, I have generated 10 values of each gamma and C and computed the best amongst them and used for the classifier. I have also calculated without using the cross-validation technique. The results were reported.

Naive Bayes Classifier : This is a type of distribution free and statistical classification. This is a probabilistic classifier which is based on the Bayes' theorem. There are many types of naive Bayes classifier, for e.g. : Gaussian naive Bayes, Multinomial naive Bayes, etc.
I have used a Gaussian naive Bayes classifier for this problem. It deals with the continuous data and treats it like it is distributed according to gaussian distribution. The mean and the variance of the data is found out and the probability distribution is applied to the data. This is how gaussian naive Bayes classifier works.

Python name : naive_bayes (GaussianNB)
model = sklearn.naive_bayes.GaussianNB ( )

Perceptron : Perceptron is used for supervised learning. It is a binary linear classifier which uses linear functions for partition. The perception function calculates the weights sum of the inputs by associating the weights with each feature.

Python name : Perceptron
model = sklearn.linear_model.Perceptron ( )

Random Forest (RF) Classifier : Random forest is an example of ensemble learning. It creates a set of decision trees from randomly selected subset of the training data. I then averages the votes from different decision trees and decides which class the test data will belong to. Random forest always controls the over-fitting of the data and improves the accuracy of the model.

Python name : RandomForestClassifier
model = sklearn.ensemble.RandomForestClassifier( )

Performance Evaluation techniques :

- **F1 score :** It is considered as a measure of test's accuracy. It considers the precision and recall which is given as follows :

$$Precision = \frac{TP}{TP + FP}$$

where, TP is true positive
and FP is false positive.

$$Recall = \frac{TP}{TP + FN}$$

where, TP is true positive
and FN is false negative.

Therefore, F1 score is the harmonic mean of precision and recall.

$$F1 = \frac{2}{1/precision \ + \ 1/recall}$$

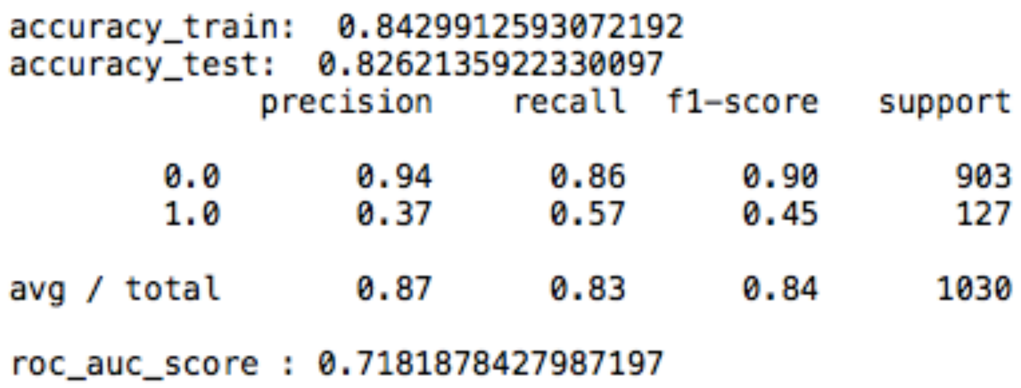
Higher the F1 score, better is the performance. So, high F1 score is desired.

- **Receiver Operating Characteristic (ROC) :** It is used to evaluate the classifier output quality. ROC curves are plotted with false positive rate on X- axis and true positive rate on Y-axis. It is considered that where the false positive is zero is the ideal case. But in general practice, we don't achieve this. So its is interpreted using the area under the curve factor. The larger is the area under curve, better will be the performance.

- **Accuracy :** Accuracy of the predicted train and test data with the actual train and test data is calculated. The higher the accuracy, the better is the model performance. Higher value of accuracy is desired.

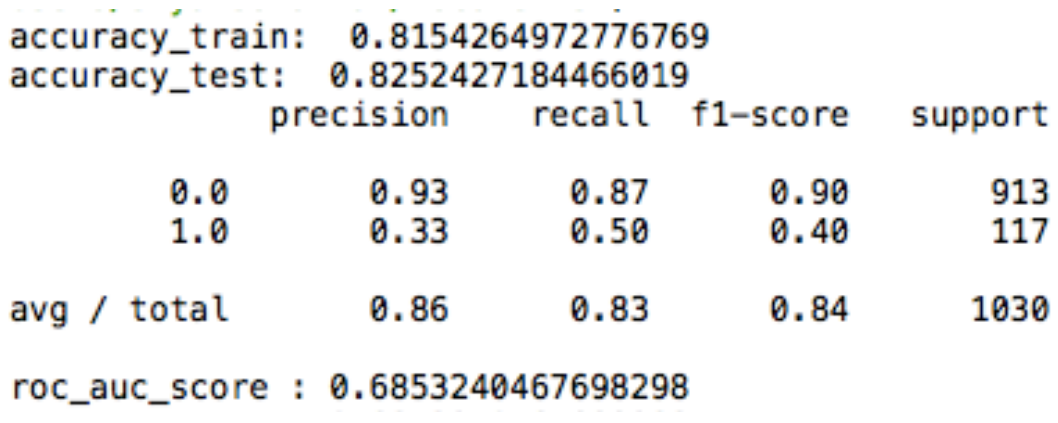
Results :

Model M1 : SVM with PCA feature extraction(gamma =1, C= 1) (oversampling)



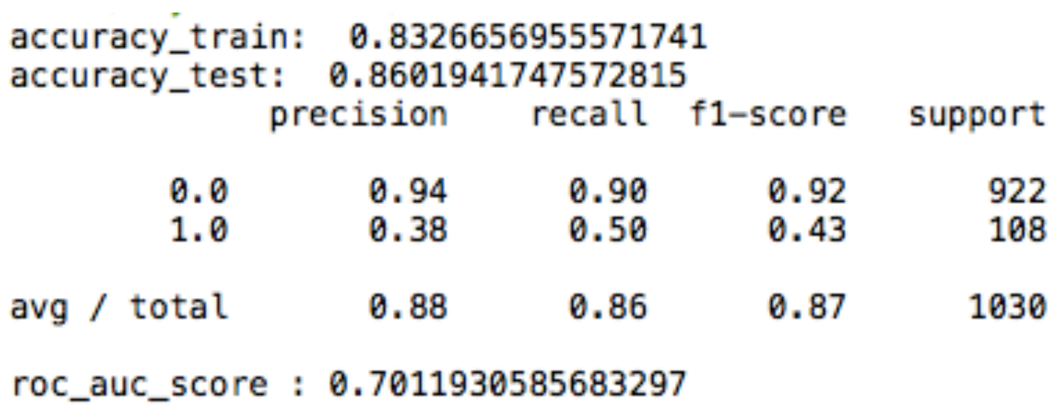
```
accuracy_train:  0.8429912593072192
accuracy_test:  0.8262135922330097
             precision    recall  f1-score   support

        0.0       0.94      0.86      0.90       903
        1.0       0.37      0.57      0.45       127

avg / total       0.87      0.83      0.84      1030

roc_auc_score : 0.7181878427987197
```

Model M2 : SVM without PCA feature (gamma =1, C=1) (oversampling)



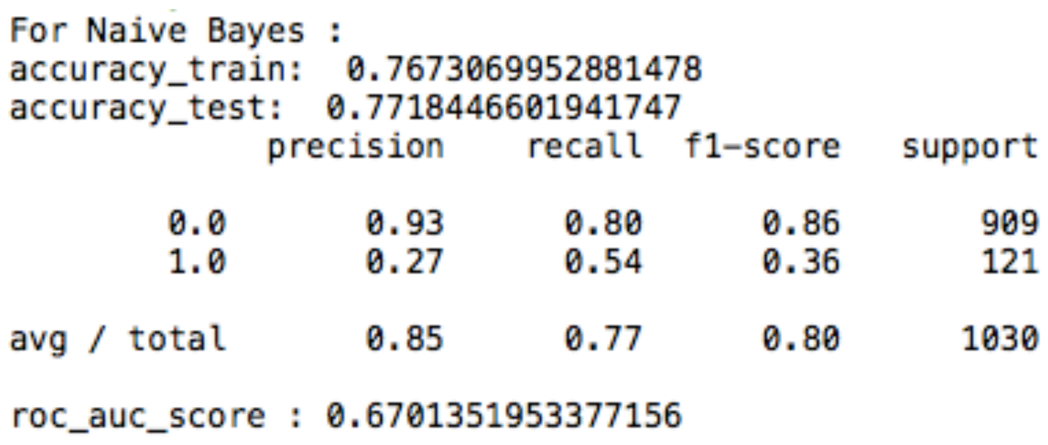
```
accuracy_train:  0.8154264972776769
accuracy_test:  0.8252427184466019
             precision    recall  f1-score   support

        0.0       0.93      0.87      0.90       913
        1.0       0.33      0.50      0.40       117

avg / total       0.86      0.83      0.84      1030

roc_auc_score : 0.6853240467698298
```

Model M3 : SVM with PCA feature extraction without oversampling



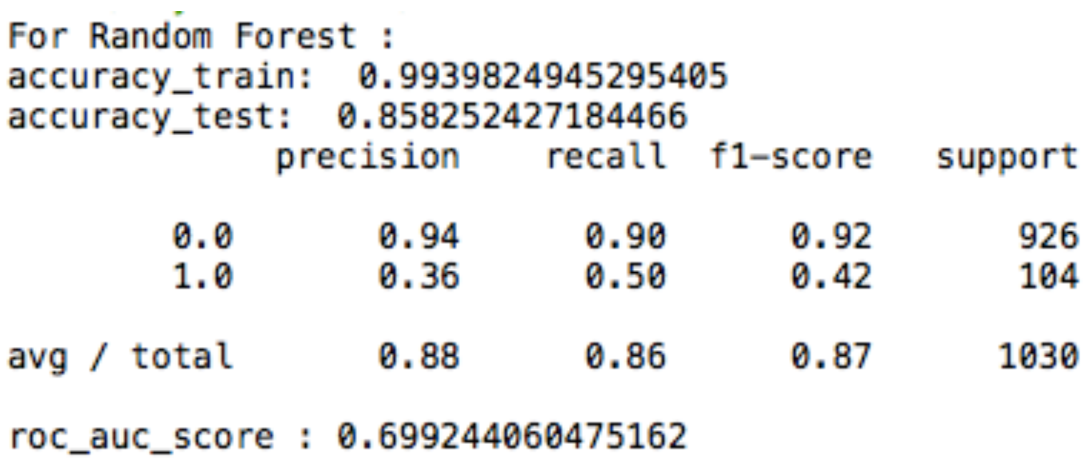
```
accuracy_train:  0.832665695557174l
accuracy_test:  0.8601941747572815
             precision    recall  f1-score   support

        0.0       0.94      0.90      0.92       922
        1.0       0.38      0.50      0.43       108

avg / total       0.88      0.86      0.87      1030

roc_auc_score : 0.7011930585683297
```

Model M4 : Naive Bayes Classifier with PCA feature extraction. (oversampling)



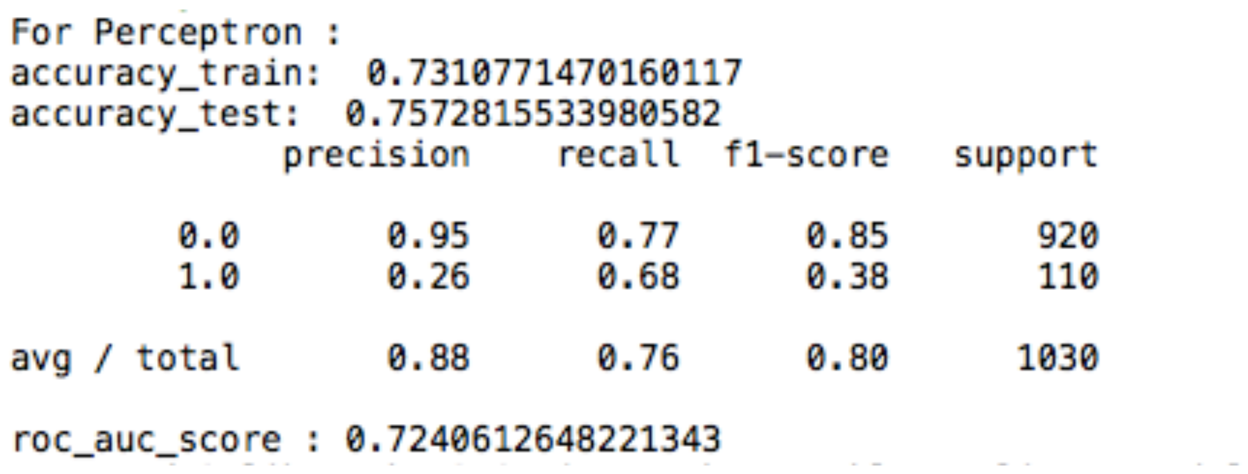
```
For Naive Bayes :
accuracy_train:  0.7673069952881478
accuracy_test:  0.7718446601941747
             precision    recall  f1-score   support

        0.0       0.93      0.80      0.86       909
        1.0       0.27      0.54      0.36       121

avg / total       0.85      0.77      0.80      1030

roc_auc_score : 0.6701351953377156
```

Model M5 : Random Forest Classifier with PCA feature extraction. (oversampling)



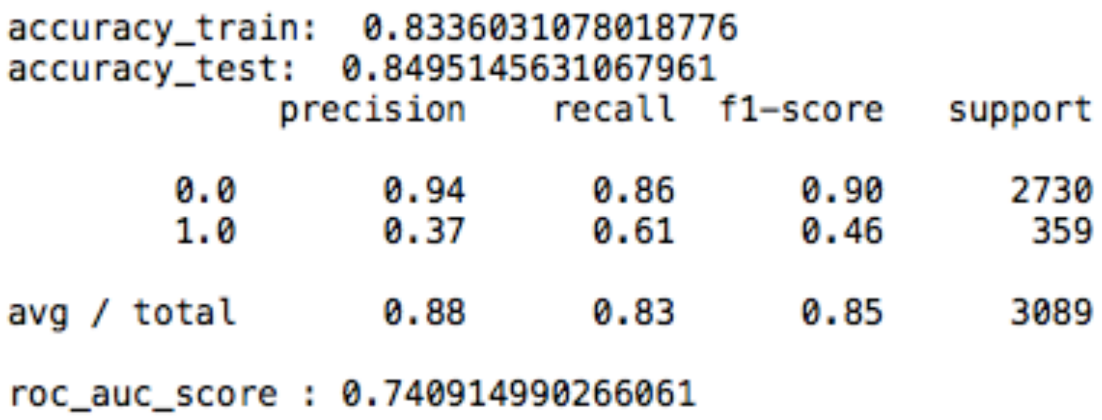
```
For Random Forest :
accuracy_train:  0.9939824945295405
accuracy_test:  0.858252427184466
             precision    recall  f1-score   support

        0.0       0.94      0.90      0.92       926
        1.0       0.36      0.50      0.42       104

avg / total       0.88      0.86      0.87      1030

roc_auc_score : 0.699244060475162
```

Model M6 : Perceptron Classifier with PCA feature extraction. (oversampling)



```
For Perceptron :
accuracy_train:  0.7310771470160117
accuracy_test:  0.7572815533980582
             precision    recall  f1-score   support

        0.0       0.95      0.77      0.85       920
        1.0       0.26      0.68      0.38       110

avg / total       0.88      0.76      0.80      1030

roc_auc_score : 0.7240612648221343
```

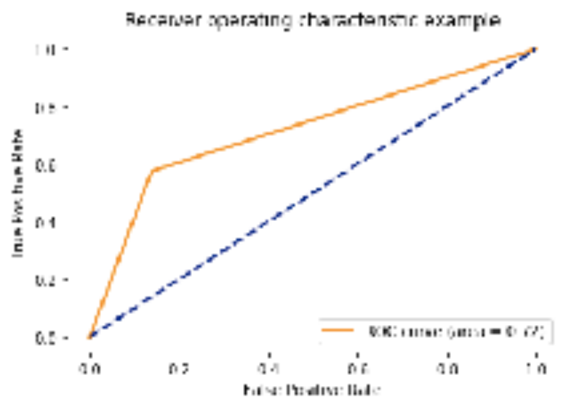
Model M7 : SVM with PCA feature extraction and cross validation . (oversampling)



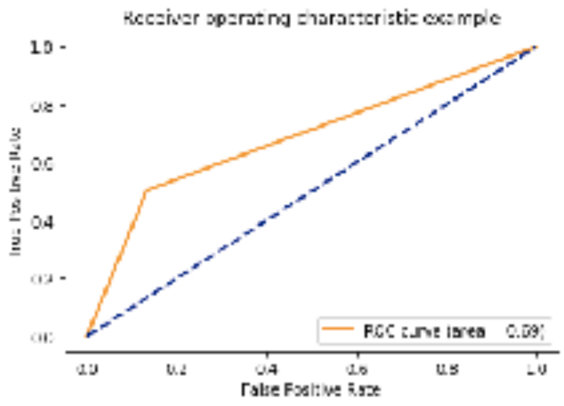
```
accuracy_train:  0.8336031078018776
accuracy_test:  0.8495145631067961
             precision    recall  f1-score   support

        0.0       0.94      0.86      0.90      2730
        1.0       0.37      0.61      0.46       359

avg / total       0.88      0.83      0.85      3089

roc_auc_score : 0.740914990266061
```

ROC curves for all models :

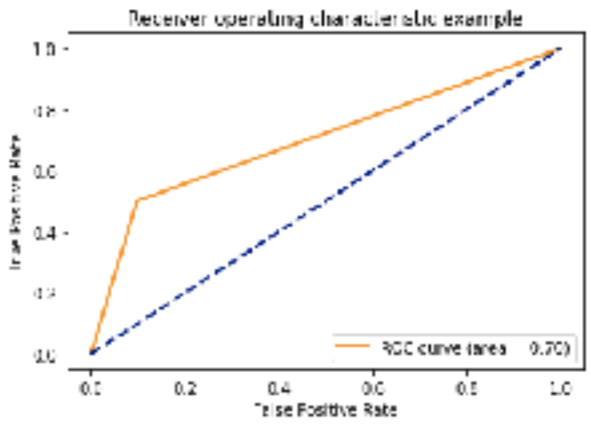
Model 1 :



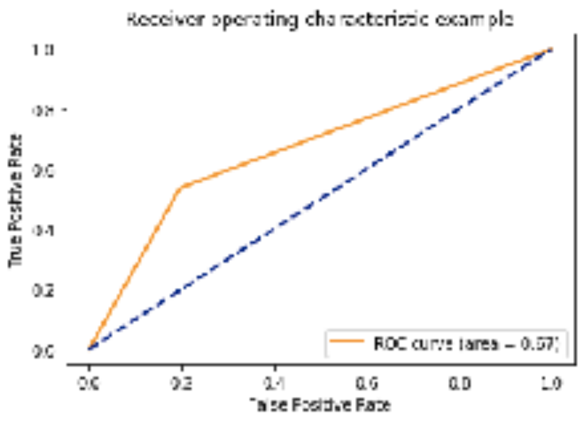
Model 2 :



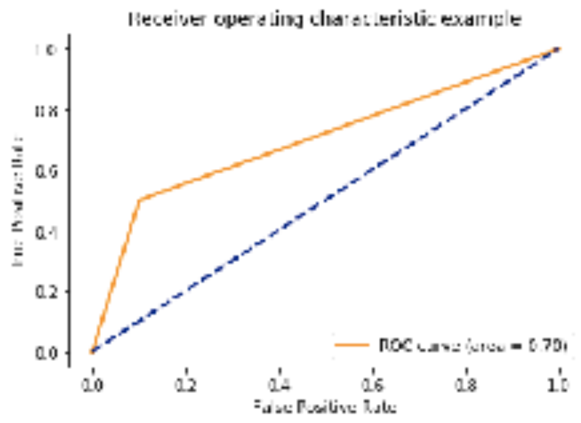
Model 3 :



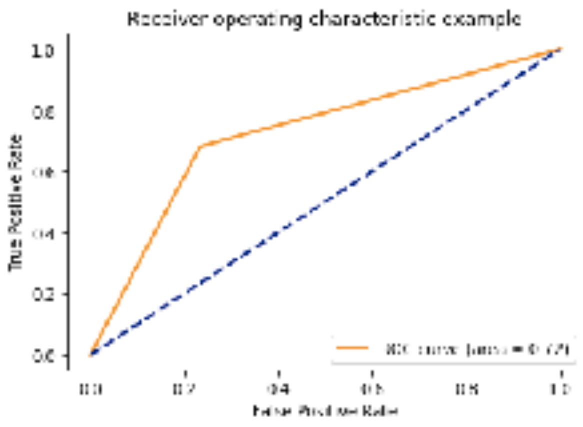
Model 4 :



Model 5 :



Model 6 :



<u>Interpretation of the results :</u>

- I tried running SVM with PCA feature extraction and with oversampling and different combinations of all this. It is found that SVM with oversampling give the best F1score (Ftest). Here, F1-score is only the measure to check if the model is good or not. The F1 score is almost the same for all the cases, therefore, I considered the area under the curve. For the model 1, the area under curve is maximum and it is obvious as it is using oversampling and PCA feature extraction as well.

- In the naive Bayes classifier, it is observed that average F1 score is pretty good but the area under the curve is not that high which is recommended and it is less than the SVM classifier.

- In the Random Forest Classifier, the F1 score is good but the area under the curve is less than the SVM classifier. But it is better than the naive Bayes classifier.

- In the Perceptron model, the F1 score is lower than RF classifier but the are under the curve is much better than RF and naive Bayes classifier. Higher area under curve is always need and the one having is considered to be good.

- Lastly, I performed cross-validation on the SVM model, and I got good F1 score as well as good area under curve. Both the metrics are equally good for this. We can say that cross validation helps in increasing the performance of the model and trains the model properly to give high F1 score.

- I tried solving the problem without using over-sampling, and the results was not that good. The F1 score got reduced and so the area under curve as well. It is better we use over-sampling to balance the data in both the classes. Use of over-sampling gave good results.

- The use of PCA controlled the problem of overfitting which is a plus point for our classifiers.

- Therefore, as observed from the results, we can see that there is no any best classifier amongst them. The use of cross-validation made a difference. In pattern recognition, there is no such best classifier. Every problem has its own best classifier and it varies with different datasets.

Computer Language : Python using anaconda (Sypder).
Toolboxes : Scikit-learn.
Libraries : Numpy, Pandas, Matplotlib.

References :

1] http://scikit-learn.org/stable/
2] https://www.wikipedia.org/
3] Book : Pattern Classification by D.H.S