EE 559 Project

Pattern Recognition System for Bank Dataset

Name: Deepika Kanade

Email ID: kanade@usc.edu

USC ID: 5369288614

Table of contents

A) Abstract

This project deals with a pattern recognition problem that operates on the bank dataset to decide whether a person will subscribe to a deposit after he receives a marketing call from the bank. The output of this pattern recognition system is a yes or a no depending upon what is predicted given the test data. To solve this problem a lot of features have been provided namely the duration to the last attempt to the person, his personal history such as loan, education, housing history and many such features. The given dataset is first divided into testing and training data and the training data is used to train the classifiers. As the data is improper and imbalanced i.e. it has a lot of unknown values and different proportion of data with respect to the two classes, pre-processing has to be performed. The data is first visualized to decide the preprocessing steps to be used. The pre-processing steps include replacing unknown values, dropping some features, removing outliers and converting categorical data to numerical data, normalization of data. The training data is then given to classifiers such as Support Vector Machines, Random Forest, K Nearest Neighbors and Naïve Bayes. The performance of these classifiers is studied using performance evaluation metrics such as F1 score, accuracy, confusion matrix, ROC curves, Area under the curve etc. The best results were obtained for **Random Forest Classifier** which gives an **F1 score of 0.869** and **AUC score of 0.698**.

B) Goal of the project

The main goal of the project was to provide a robust pattern recognition system that will accurately predict if the client will subscribe to the bank for its deposits. To do so, maximum accuracy is desired which is obtained by processing the data properly and using classifiers that will properly classify unknown data.

The language used by me for this project is Python. I used some inbuilt libraries of python like sklearn.

C) Loading the dataset

The bank dataset provided to us was in the form of a csv file. I loaded the data given to us in a pandas data frame. Csv parsing had to be done on the data which I did using the pd.readcsv() command in python.

D) Visualization of data

I visualized each and every feature given to us to study if that feature is useful for discriminating. My main goal to do this part was to see if the features are discriminant and to keep only those features which were discriminant and eliminate those features which were not. To visualize the data, I used box plots and histograms for every feature.

E) Preprocessing of data

The preprocessing of data is a very important step in Pattern recognition problem. This step helps in replacing the unknowns in the data. Also, the features can be normalized in a particular range so that one feature does not over power the other features.

- *Splitting of train and test data*

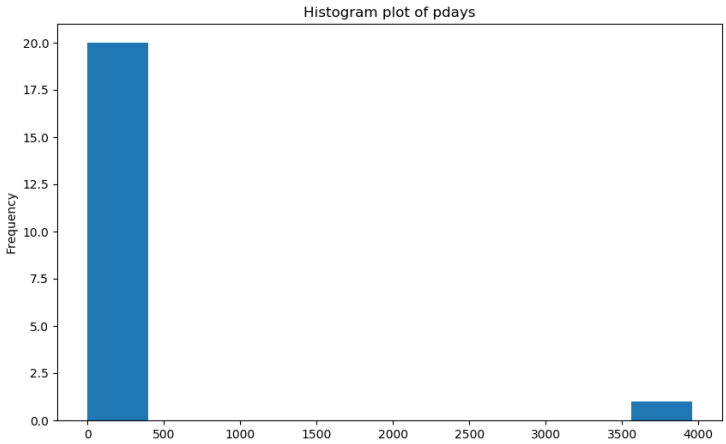
The preprocessing step is done after splitting the data into training data and test data. I am separating the data into 80% training data and 20% testing data. The original dimension of data is 4119x19.The dimension of the train data becomes 3295x19 and that of the test data becomes 824x19. The train_test_split() from sklearn library is used to separate the training and testing data. While splitting the data there was a glitch as in the education column there is just one data point having 'illiterate' category. Hence, when the splitting of data was done, this point went in either the test or train data which was later one hot encoded. This created dimension mismatch which is why that datapoint had to be discarded prior to splitting.

The preprocessing techniques I have used are summarized as follows:

- *Preprocessing of data*

a) Dropping features

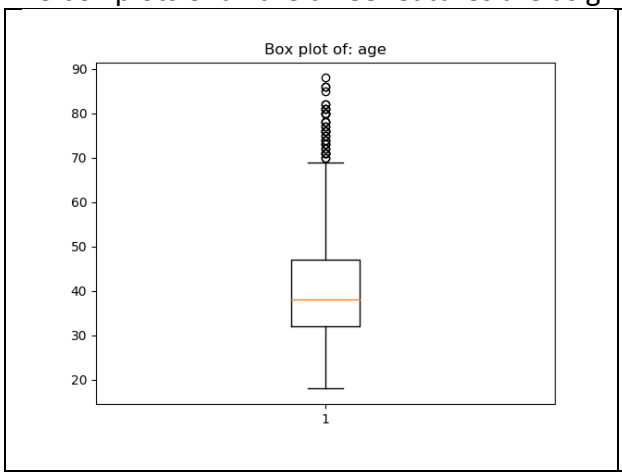
After observing the histograms of all the features in the dataset, I could observe that some features have their data clustered in one bin. For instance, in pdays feature, maximum data points have the value as 999 which indicates that the person was not called before. Hence, this does not become a discriminating feature according to me. This can be clearly observed from the histogram shown below:

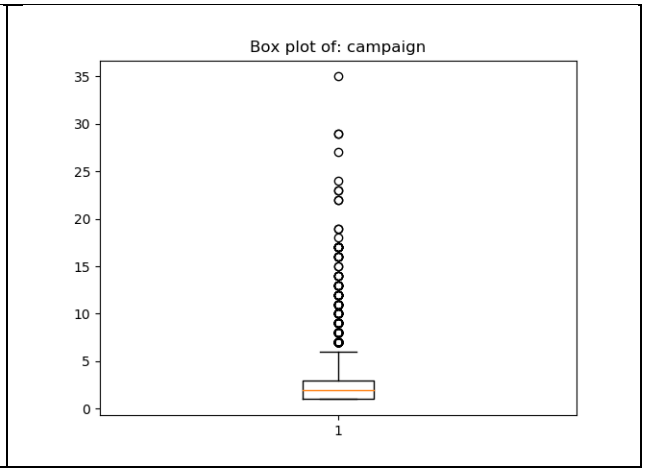


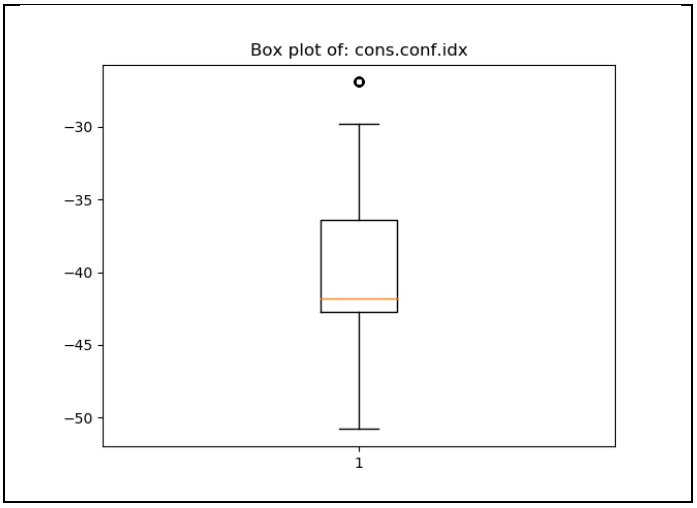
Hence, I discarded this feature from the dataset. Also, another feature that won't be discriminative according to me is the default feature. This feature has 3 categories namely unknown, yes and no. 3315 data points have the value no and 803 data points have the value as unknown. Only 1 data point has values yes. Hence, when we replace the unknowns in the categorical feature with mode value, all the unknown values will be replaced by no. Hence, the distribution of data points having a value 'no' will be high. Therefore, this feature won't be useful and can be dropped.
To summarize, I dropped the 'pdays' and 'default' features from the training and testing data.

b) Outlier removal

An outlier is a data point that is far away from the distribution of the data. Therefore, this data point will not be useful in classification, rather it can lead to wrong training of the classifier. These data points can be dropped. After studying the box plots of all the features, I concluded that the 'age', 'campaign', 'cons.conf.idx' features have outliers in them which can be removed by setting thresholds.
The box plots of all the three features are as given below:







The thresholds for all the three features are as follows:

Age	69
Campaign	20
cons.conf.idx'	-29

Hence, all the values less than this threshold are retained and the values greater than the threshold are discarded.

c) Label encoding of ordered categorical features

Some features in the data are label encoded. The reason for encoding is that some features in the data has string values which have to be converted into numerical values for further processing. The data which is in ordered manner can be encoded using label encoding. In label encoding, the feature space is not expanded as the LabelEncoder() function in python calculates the unique values and assigns them separate labels in the same columns itself.
The following features were Label Encoded:

- Month
- Day_of_week
- y

The y feature is the label feature which cannot be a part of the training procedure.

d) Dealing with missing data

The bank dataset given to us has a lot of missing values. These missing values are only in the categorical features. The values are in the form of 'unknown' in the dataset. Hence, these 'unknown' data points have to be imputed. I am using the mode imputing procedure in which the mode of the feature column is computed and then the unknowns are replaced by this mode value. Before replacing the missing data, the features have to be label encoded so that we identify the label number of the unknown data and then this label number is replaced by the mode. This entire process is done by the Imputer() function in python which is a part of the skelarn preprocessing toolbox of python. The label number is to be replaced and the method of replacement i.e. mode has to be specified in this function.
The missing data replacement has been done on the following features:

- Job
- Marital
- Housing
- Loan
- Education

e) One Hot Encoding of Unordered categorical features

The unordered categorical features i.e. features which have no predefined sequence but are string features have to be converted to one hot vectors. By using one hot encoding, the feature

space is expanded. One hot encoding is done to all the features that are filled with missing values. The pd.getdummies() function from python is used to do the encoding.
The features that are encoded using the method are:

- Job
- Marital
- Education
- Housing
- Loan
- Poutcome
- Contact

After the one hot encoding is done, these vectors have to be joined to the data frame and the original column has to be dropped from the data frame as that feature has already been encoded.

f) Oversampling using SMOTE

Due to imbalanced data in the dataset, under sampling or oversampling operations have to be performed. The oversampling operation will increase the data points and will result in providing more data for the classifier to classify. To increase the data points, weights are given to the classes so that the data points of both the classes are increased. This definitely increases the accuracy. The imblearn library has a lot of options for SMOTE wherein I am using the SMOTENN function. The Synthetic Minority Over-sampling Technique(SMOTE) samples a lot of noisy data points. Hence, this data has to be cleaned which is done by using Edited Nearest Neighbors. The inclusion of SMOTEENN function definitely increased out accuracy. The dimension of data after application of Smote is 5597x30 which is considerably high.

g) Feature Dimensionality Reduction

Feature Dimensionality reduction is done on the given data so as to get the data with maximum variance. The main thing to be taken care of while reducing the dimension is that the reduced data should contain maximum amount of information as that of the original data. That is, the loss of information should be very less.
I tried using PCA as a dimensionality reduction method. I tried changing the principal components to values of 5,10,15,20 but the accuracies reduced. This was probably because reducing the number of features from 39 to some lesser value won't be beneficial as the classifier will have fewer values to train with. Hence, I decided to drop the method of PCA.

```
---------K Nearest Neighbors Classifier--------
K Nearest Neighbors Train Accuracy: 1.000
K Nearest Neighbors Test Accuracy: 0.362
K Nearest Neighbors Classification Report:
             precision    recall  f1-score   support

          0       0.87      0.34      0.49      1084
          1       0.09      0.57      0.16       126

avg / total       0.79      0.36      0.45      1210

K Nearest Neighbors Training F1 Score: 1.000
K Nearest Neighbors Testing F1 Score: 0.452
K Nearest Neighbors Training AUC Score: 1.000
K Nearest Neighbors Testing AUC Score: 0.455
```

The testing accuracy obtained by using PCA decreases and also overfitting of training data takes place which can be seen by observing the training accuracy. Hence, PCA is avoided.

h) Normalization

Normalization of data leads to transforming all the data to have values between 0 and 1. Due to this, all the features will have equal importance while classifying rather than letting a feature with higher value dominate the classification process. The formula used for Normalization is as follows:

$$X_{new} = \frac{X - X_{\min}}{X_{max} - X_{\min}}$$

The sklearn preprocessing toolbox has an inbuilt function MinMaxScaler() which does the normalization.

F) Classifiers

Four classifiers are used to classify the unknown data namely Random Forest, K Nearest Neighbors, Support Vector Machine and Naïve Bayes. The best accuracy is given by the Random Forest classifier giving a high f1 score as well. It is quite intuitive to get a high F1 score on the Random Forest classifier as this classifier works well on imbalanced data. I also tried some other classifiers such as Perceptron, Stochastic Gradient Descent, Adaboost, Multilayer Perceptron but they did not give the best accuracies as compared to the above-mentioned classifiers.

1) Random Forest Classifier

Random forest creates decision trees from randomly selecting datasets of training data and then taking the vote of all these decision trees to make a prediction. Instead of considering equal voting from all the trees, the weights on the trees can be changed. Hence, trees having low error can be given more weight whereas trees having high error will be given less weight. Due to this, the classification will be proper. The parameters in random forest are the total number of decision trees, the minimum split etc. I used the Randomclassifier() function of sklearn library to hyper tune the parameters using 5 fold cross validation. The most important

parameters are the total number of trees to be used, the maximum features which decide the split of data etc. The random forest uses random features while deciding to split a node or not which brings in a lot of randomness while creating the trees. This in turn helps in classification of unknown data points. The main reason why Random forests are used is because they prevent the problem of overfitting as they create smaller trees and combine them together to generate an ensemble. The best accuracy and F1 score for the bank dataset was obtained using Random Forest which is intuitively correct as RF works very well for imbalanced datasets. The **F1 score achieved was 0.869** and the **AUC score was 0.698**.

2) Support Vector Machines

Support Vector Machines (SVM) are very popular in supervised machine learning problems due to their special characteristics to map point from a non-linear feature space to linear feature space. In support Vectors, the hyperplane is constructed between two classes in such a way that the separation between them is optimized with respect to the support vectors. Support vectors are the data points helping the SVM to construct an optimal decision Boundary. For using an SVM, constraints are added to the learning problem which help in optimizing the decision boundary. By adding such constraints, the decision boundary created is able to classify the data properly.

The mapping from non-linear space to linear feature space is done by using a kernel. I used the svm.SVC() function of sklearn to implement SVM. A lot of parameters can be fine-tuned in this function. First of all the kernel to be used has to be specified. For each kernel, some parameters have to be changed. I used cross validation to decide the parameters in the SVM classifier. For an RBF kernel, **C and gamma parameters** have to be specified which give the regularization and the extent to which a sample training sample reaches. The accuracy and F1 score obtained for SVM is quite decent. The **F1 score obtained was 0.774** and **the AUC score was 0.688.**

3) K Nearest Neighbors

K-nearest neighbors is a non-parametric technique of classification which gives better classification in cases where the decision boundary is irregular. The KNN uses the concept of finding the K nearest neighbors of the data point and assigning a label to the data point based on the distance of the data point with its neighbors. The distance can be calculated using L1 distance, L2 distance etc. The most challenging part in using this classifier is to decide the value of k. For deciding upon k, a range of values can be given provided and 5-fold cross validation can be used to obtain the best result from them. As KNN is sensitive to irrelevant features in the dataset, so proper processing has to be done before we use KNN so as to get the desired accuracy. The **F1 score** I got for KNN is **0.806** and the **AUC score is 0.645**. The best value of K obtained after cross validation was 10.

4) Naïve Bayes classifier

Naïve Bayes classifier takes into account the prior distribution of class-wise data and the probability density function. Hence, Naïve Bayes method is not distribution free classification method. The main concept used in Naïve Bayes classifier is that each feature is independent. The Bayes decision Rule can be given as follows:

$$P\left(\frac{x}{Si}\right) * P(Si) > P\left(\frac{x}{Sj}\right) * P(Sj)\ then\ x\ belongs\ to\ class\ i$$

I used the GausssianNB() function in python which is there in the sklearn library. The only parameter to be used in the GaussianNB() function is the class priors. So, if the priors are not provided the function chooses them according to distribution of data. **The F1 score** I got for Naïve Bayes Classifier **is 0.764** and the **AUC score is 0.674**.

G) Performance Evaluation

To evaluate the performance of any classifier, certain performance metrics are used. These include the F1 score, Area Under the Curve (AUC), Receiver Operating Characteristics (ROC), Confusion Matrix, Accuracy score etc. The classifier is trained using the training data and the unknown testing data is given to it to check the classification for this unknown data. By providing the unknown test data, the classifier makes certain predictions. As this is a supervised learning method, the test labels are already provided as ground truths. Hence, these test labels are compared to the predicted labels to measure the parameters specified above. Classification Report is generated which gives the precision and recall values which give us an idea of the performance of the classifier.
Depending on the predictions made by the classifier, predicted labels can be divided into 4 types:

- True Positive
- True Negative
- False Positive
- False Negative

The true positive and True negative are the best cases as they coincide with the expected outputs. False Positive and False negative cases should be avoided.
The precision value for the data is given as,

$$Precison = \frac{(True\ Positive)}{True\ Positive + False\ Positive}$$

The recall value for the data is given as,

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

The F1 score is given as:

$$F1\ Score = 2 * precison * \frac{recall}{precision + recall}$$

The functions used in python to evaluate these performance metrics are accuracy_score(), f1_score(), classification_report(), roc_curve(), auc(). All these functions are present in the sklearn metrics library.

H) Results

The result for all the classifiers are given as follows:

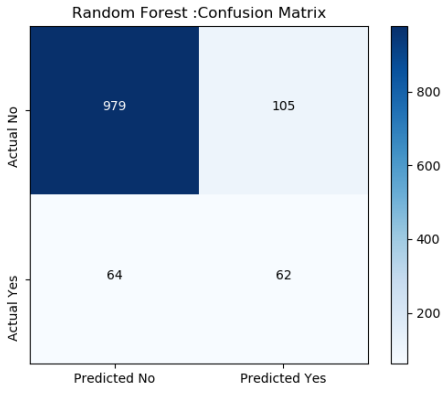
```
---------Normalization of data is done-----------

---------Random Forest Classifier--------
Random Forest Train Accuracy: 0.999
Random Forest Test Accuracy: 0.860
Random Forest Classification Report:
             precision    recall  f1-score   support

          0       0.94      0.90      0.92      1084
          1       0.37      0.49      0.42       126

avg / total       0.88      0.86      0.87      1210

Random Forest Training F1 Score: 0.999
Random Forest Testing F1 Score: 0.869
Random Forest Training AUC Score: 0.999
Random Forest Testing AUC Score: 0.698

---------SVM Classifier--------
SVM Train Accuracy: 0.871
SVM Test Accuracy: 0.724
SVM Classification Report:
             precision    recall  f1-score   support

          0       0.95      0.73      0.83      1084
          1       0.22      0.64      0.33       126

avg / total       0.87      0.72      0.77      1210

SVM Training F1 Score: 0.872
SVM Testing F1 Score: 0.774
SVM Training AUC Score: 0.865
SVM Testing AUC Score: 0.688

---------K Nearest Neighbors Classifier--------
K Nearest Neighbors Train Accuracy: 0.996
K Nearest Neighbors Test Accuracy: 0.772
K Nearest Neighbors Classification Report:
             precision    recall  f1-score   support

          0       0.93      0.81      0.86      1084
          1       0.22      0.48      0.31       126

avg / total       0.86      0.77      0.81      1210

K Nearest Neighbors Training F1 Score: 0.996
K Nearest Neighbors Testing F1 Score: 0.806
K Nearest Neighbors Training AUC Score: 0.997
K Nearest Neighbors Testing AUC Score: 0.645
```

```
---------Naive Bayes Classifier--------
Naive Bayes Train Accuracy: 0.751
Naive Bayes Test Accuracy: 0.711
Naive Bayes Classification Report:
             precision    recall  f1-score   support

          0       0.94      0.72      0.82      1084
          1       0.21      0.63      0.31       126

avg / total       0.87      0.71      0.76      1210

Naive Bayes Training F1 Score: 0.755
Naive Bayes Testing F1 Score: 0.764
Naive Bayes Training AUC Score: 0.762
Naive Bayes Testing AUC Score: 0.674
```

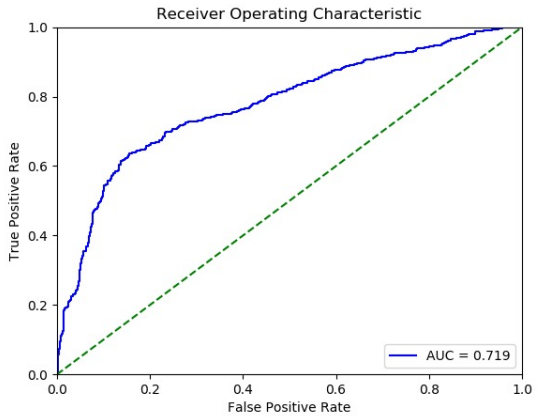
The entire output is summarized below:

Classifier	Testing F1 score	Testing AUC score	Testing Accuracy	Precision score	Recall score
Random Forest	**0.869**	**0.698**	**86%**	**0.88**	**0.86**
SVM	0.774	0.688	72.4%	0.87	0.72
KNN	0.806	0.645	77.2%	0.86	0.77
Naïve Bayes	0.764	0.674	71.1%	0.87	0.71

Also, to get a better understanding of the outputs generated, the confusion matrix and the ROC curve for the best classifier i.e. Random Forest is displayed.



Confusion Matrix



ROC plot

As seen from the figures, the confusion matrix obtained is good as the number of 'No' labels in the data are more which are correctly classified by the Random Forest classifier. The True Positives and the True Negatives are high which is the desired case. Hence, this proves that the Random Forest works well on the given dataset.

I) Interpretation of Results

- Preprocessing was a very important step that helped in getting better accuracies. As some features contain important data as compared to others, every feature has to be preprocessed so that they are all in the same range and help in classification.
- Using cross validation for parameter estimation helped greatly as it increased the accuracies and the F1 score for every classifier.
- The best classifier from the obtained results is the Random Forest Classifier. The RF classifier works well for imbalanced data set which can be proven from the obtained results.
  The best accuracy and F1 score of Random Forest Classifier is as given below:
  F1 score- 0.869
  Testing Accuracy- 86%
  AUC score-0.698
- After observing the ROC curve, as the curve lies above the green line, it can be deduced that the classifier works well. The green line is for AUC of 0.5 and as the Random Forest Classifier has a greater ROC, the plot lies above the green line.
- As the number of 'no' in our labels is greater than the number of 'yes', the confusion matrix has more 'predicted no' and 'actual no'. Hence, the confusion matrix also gives us a better idea about the performance of the classifier.
- The precision and Recall values give a good measure about how many data points were falsely classified or the datapoints that were correctly classified. Hence, monitoring the recall and precision scores is important. The Precision and Recall scores for all the classifiers is pretty decent with Random Forest giving the maximum Precision of 0.88 and Recall of 0.86.

References

[1] Bank dataset - https://archive.ics.uci.edu/ml/datasets/Bank+Marketing#

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

[3] K Nearest Neighbors- http://scikit-learn.org/stable/modules/neighbors.html

[4] EE 559 Lecture notes and Discussion notes

[5] www.wikipedia.com

[6] Naïve Bayes - http://scikit-learn.org/stable/modules/naive_bayes.html

[7] SMOTE - http://contrib.scikit-learn.org/imbalanced-learn/stable/auto_examples/combine/plot_smote_enn.html

[8] Preprocessing - http://scikit-learn.org/stable/modules/preprocessing.html