[Date]

MACHINE LEARNING PROJECT

BANK MARKETING CLASSIFICATION

GUIDED BY:
DR. VINAY KULKARNI
PROF. VIPIN DUBEY

TITLE OF THE PROJECT:

BANK MARKETING CLASSIFICATION

GROUP NO.: 15

PARTICIPANT LIST:

1). SARANG BAGUL

2). RAHUL SINGH

3). MOHNISH LAVANIA

PROBLEM STATEMENT:

This project utilizes different types of Machine Learning algorithm, using the Bank Marketing dataset, to check if the client has subscribed for a term deposit depending on various bank marketing attributes like age, type of job, education level, if the client has housing loan or not, last date of contact etc.

We have chosen Logistic Regression as our Benchmark model for this model and that will be compared to following classifiers:

- Decision Tree Classifier
- Random Forest Classifier
- Support Vector Machine

Feature engineering has been done that also include Transformation (Using Principal Component Analysis) to see if there is any improvement over the classifiers(classifiers are trained separately for different feature set obtained). Finally, we compare all above mentioned classifiers with the benchmark classifier (Logistic Regression Classifier).

Metrics :

The most common used metric to measure and compare classifiers is the Accuracy.

But in the dataset, there are almost 89% example of “no” and nearly 11% example of “yes”. So due to the nature of the dataset, accuracy is not a good measure as the dataset id unbalanced. So, for the purpose of this project, F1-Score and Area Under ROC Curve has been used as the metric for the evaluation and comparison.

F1-Score considers both the precision p and recall r of the test to compute the score:

P is the number of correct positive result divided by the number of all positive result returned by the classifier, and R is the number of correct positive results divided by the number of all relevant samples (all sample that have been identified as positive). The F1-score is the harmonic average of precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. That’s the reason why it is considered a good evaluation metric when dealing with imbalanced classification because it takes into account for the false positives and false negatives.

On the other hand, Area under ROC is the area under ROC (Receiver Operating Characteristics) curve. An area of 1 represents a perfect test; an area of 0.5 represents a worthless test. In the ROC curve, the true positive rate (Sensitivity) is plotted in function of false positive rate (100-specificity) for different cut-off points of the parameter. Each point on the ROC curve represents a sensitivity / Specificity pair corresponding to a particular decision threshold, the area under the ROC curve (AUC) is the measure of how well a parameter can distinguish between two classes.

Following is the dataset that have been used for this project:

1. Bank_full.csv
    a. This dataset has 45211 entries, each with 17 inputs ordered by date. First 16 input for each entry are attributed and 17th input is the class to be predicted.
    b. Out of 16 attributes, 7 attributes (age, balance, day, duration, campaign, pdays and previous) are numeric. Rest are categorical (including binary attributes).

Following are the attributes of this dataset with brief description:

1. Age (numeric).
2. Job: type of job (categorical)

   Types: admin, unknown, unemployed, management, housemaid, student, blue-collar, self-employed, retired, technician, services)
3. Marital: marital status (categorical) married, divorced, single.
4. Education: (categorical) unknown, secondary, primary, tertiary
5. Default: has credit in default: yes or no
6. Balance: (numeric) average yearly balance (in euros)
7. Housing: has housing loan ? yes or no
8. Loan: has personal loan ?
9. Contact: contact information: unknown, telephone, cellular
10. Day: last contact day of month
11. Month: last contact month of client (jan, feb, mar,…………,dec)
12. Duration: last contact duration in seconds

Other attributes:

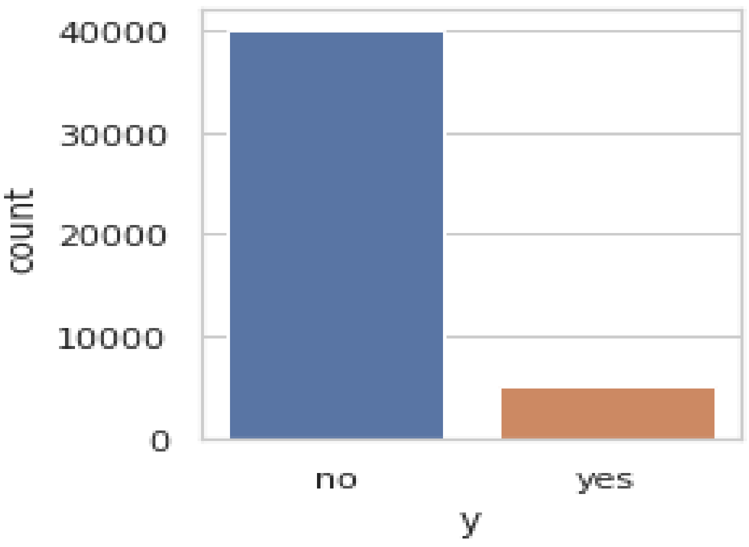
13. Campaign: number of contact performed during this campaign and for this client (numeric)
14. Pdays: number of days passed by after the client was contacted from previous campaign (numeric, -1 means client was not previously contacted)
15. Previous: number of contact perfomed before this campaign and for this client
16. Poutcome: outcome of previous marketing campaign(unknown, other, failure, success.

Output variable (desired target):

17. Has the client subscribed a term deposit ? yes or no

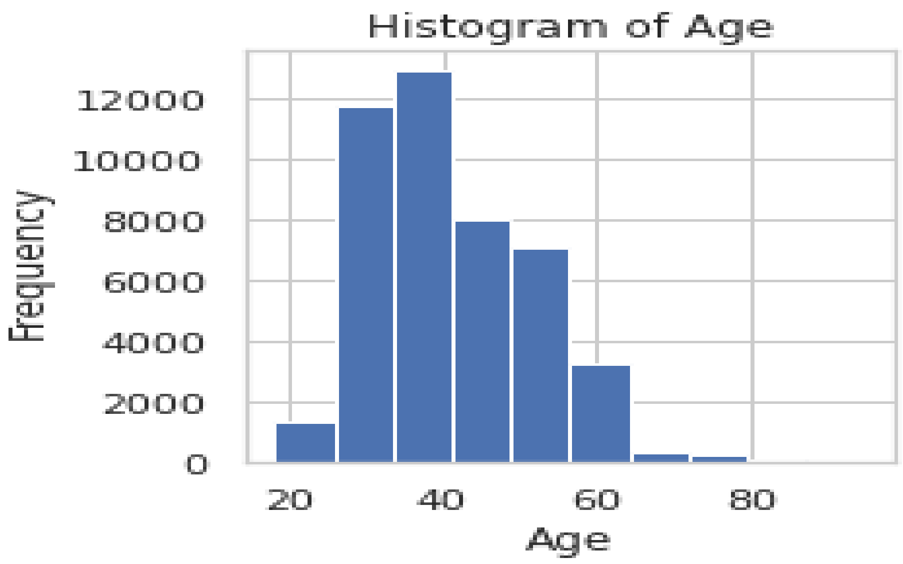
➔ **Exploratory Visualization :**

Firstly, the number of examples belonging to each class are plotted:

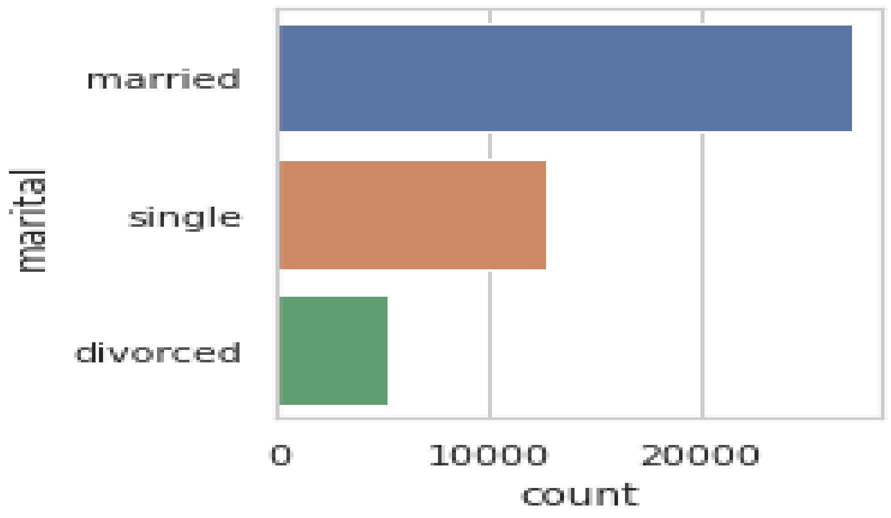


It has been observed that nearly 89% client did not subscribed for the term deposit. Therefore, there is class imbalance in this dataset.

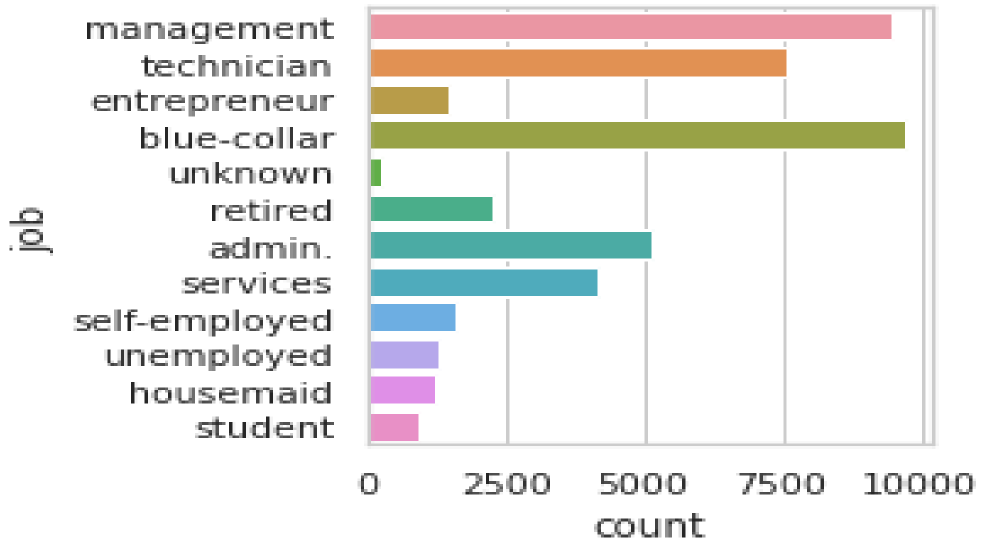
➔ histograms were also plotted for some features:



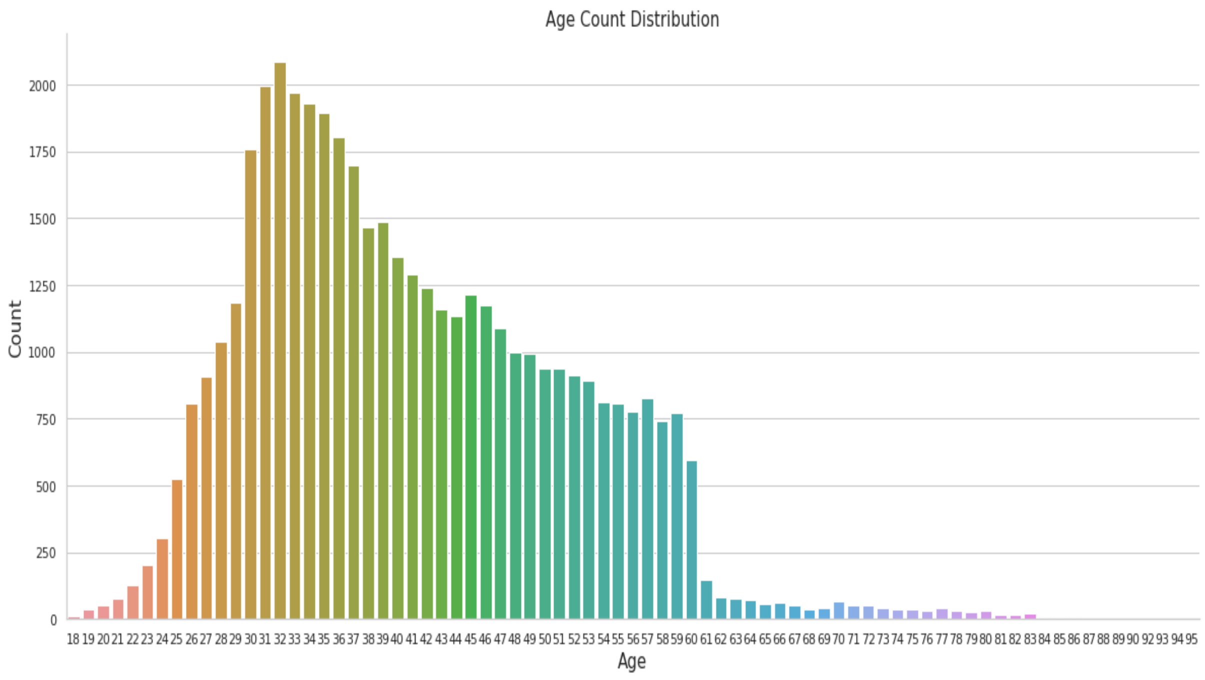
it is clearly shown from the above graph that most of the client under observations belongs to age group of 30-40 years old.



HISTOGRAM OF MARITAL FEATURE: It is observed that most of the clients are married followed by single and only some clients are divorced.

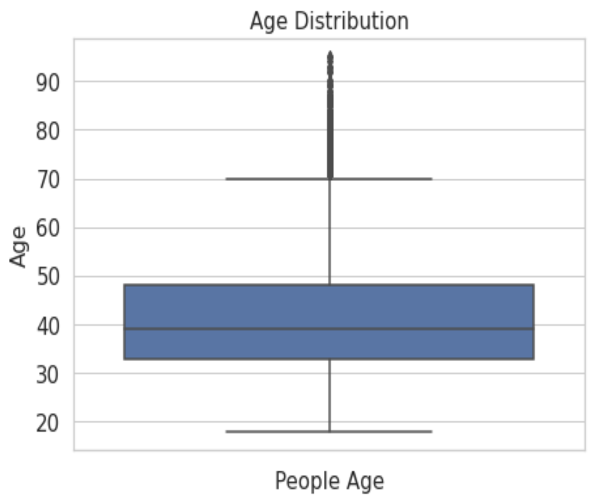


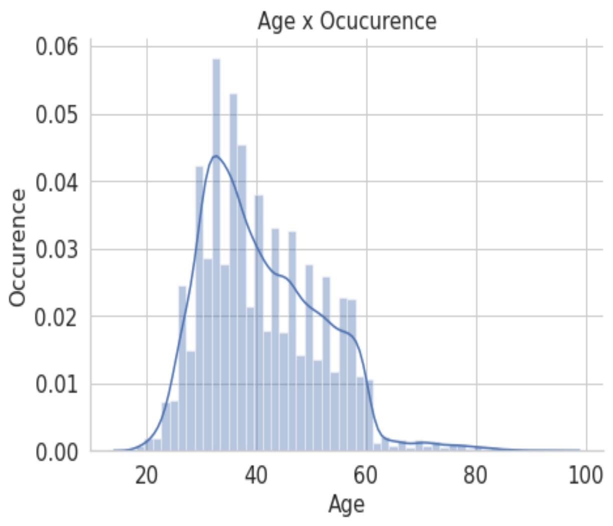
HISTOGRAM OF JOB FEATURE : This graph shows the no. of clients in various job and most of the are either in management field or technician or in blue collar job.

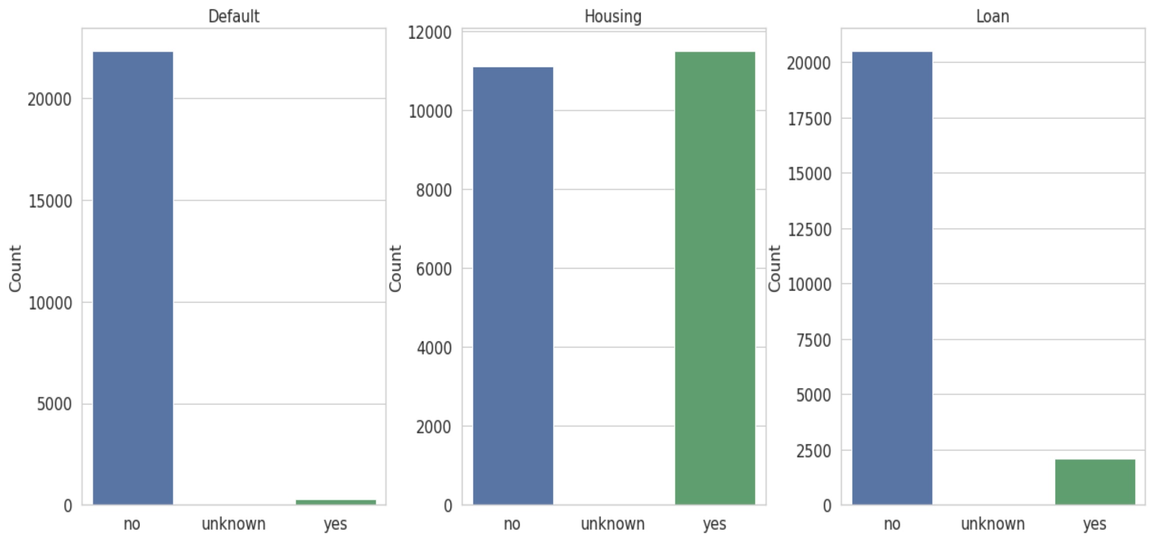


This graph shows that most of the customer belongs to 30-40 year age group and there are only some clients who is above 60 years of age.

Below graph shows the box-plot of age distribution of the clients and plot of histogram along with a curve.





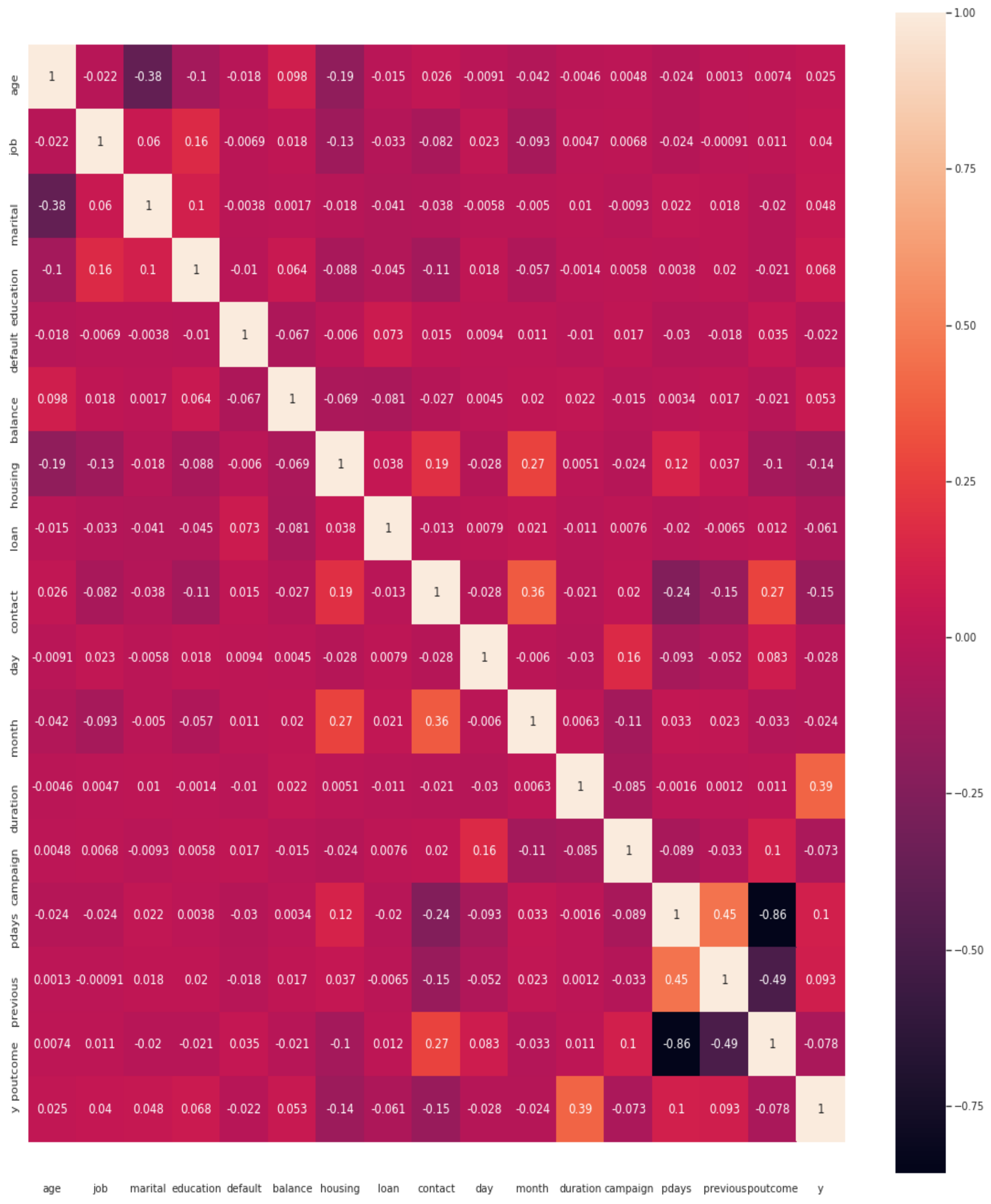


This graph shows various other binary classified attributes, whether the client has any previous loan default or not (most of them are not), whether they have housing loan or not, and whether they have any previous personal loan or not.

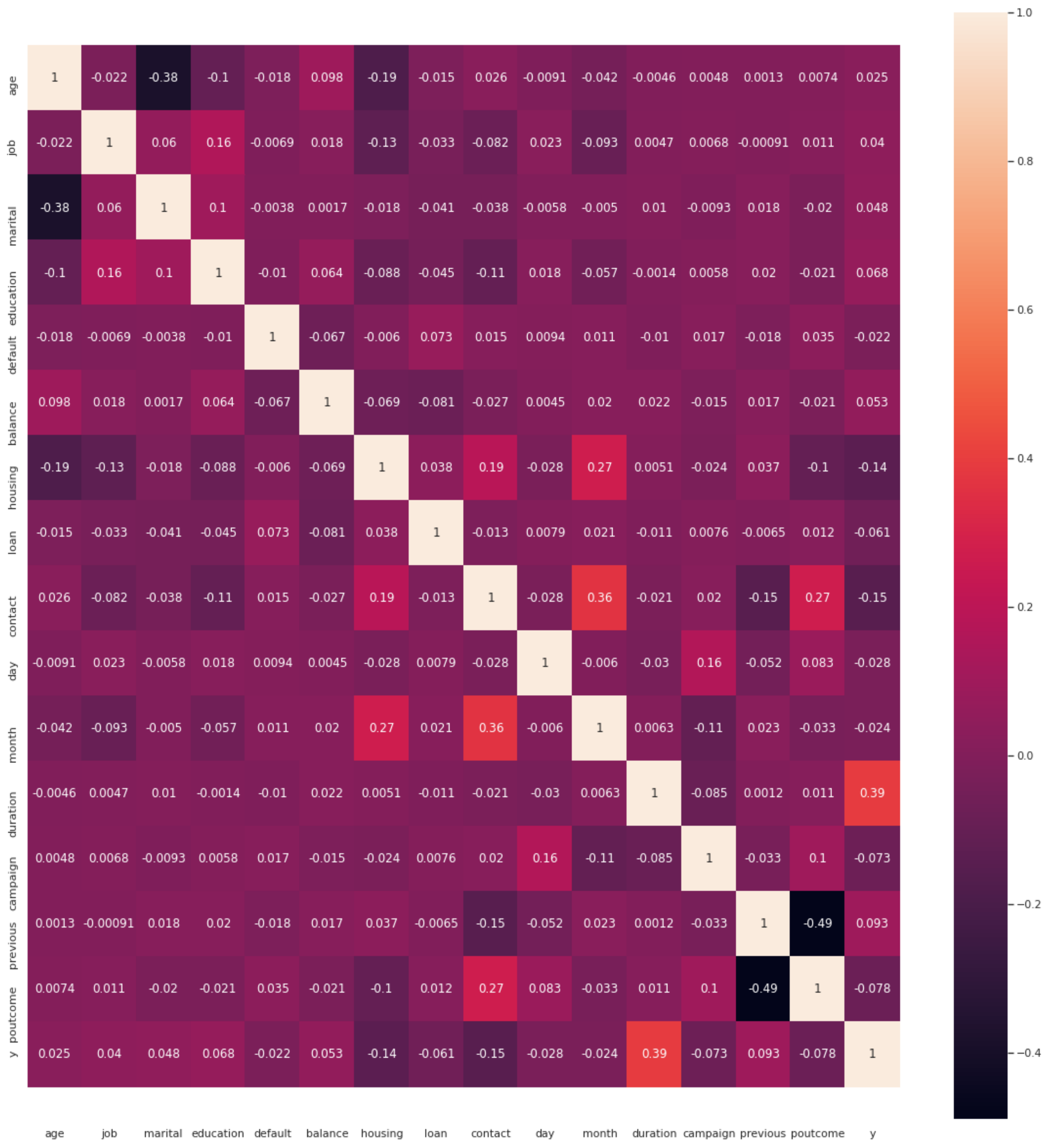
After doing exploratory visualization, we perform **LABEL ENCODING** to all the binary class attributes so that all those will be converted to numeric type and for easier calculations.

After doing the exploratory visualization and feature engineering, correlation matrix was calculated to find out those features that are highly correlated to each other, a heatmap (shown below) was plotted to identify those candidate feature pairs.

Feature pair pdays-previous is highly positively correlated, therefore, we can remove feature ‘pdays’.


age 1 -0.022 -0.38 -0.1 -0.018 0.098 -0.19 -0.015 0.026 -0.0091 -0.042 -0.0046 0.0048 -0.024 0.0013 0.0074 0.025
job -0.022 1 0.06 0.16 -0.0069 0.018 -0.13 -0.033 -0.082 0.023 -0.093 0.0047 0.0068 -0.024 -0.00091 0.011 0.04
marital -0.38 0.06 1 0.1 -0.0038 0.0017 -0.018 -0.041 -0.038 -0.0058 -0.005 0.01 -0.0093 0.022 0.018 -0.02 0.048
education -0.1 0.16 0.1 1 -0.01 0.064 -0.088 -0.045 -0.11 0.018 -0.057 -0.0014 0.0058 0.0038 0.02 -0.021 0.068
default -0.018 -0.0069 -0.0038 -0.01 1 -0.067 -0.006 0.073 0.015 0.0094 0.011 -0.01 0.017 -0.03 -0.018 0.035 -0.022
balance 0.098 0.018 0.0017 0.064 -0.067 1 -0.069 -0.081 -0.027 0.0045 0.02 0.022 -0.015 0.0034 0.017 -0.021 0.053
housing -0.19 -0.13 -0.018 -0.088 -0.006 -0.069 1 0.038 0.19 -0.028 0.27 0.0051 -0.024 0.12 0.037 -0.1 -0.14
loan -0.015 -0.033 -0.041 -0.045 0.073 -0.081 0.038 1 -0.013 0.0079 0.021 -0.011 0.0076 -0.02 -0.0065 0.012 -0.061
contact 0.026 -0.082 -0.038 -0.11 0.015 -0.027 0.19 -0.013 1 -0.028 0.36 -0.021 0.02 -0.24 -0.15 0.27 -0.15
day -0.0091 0.023 -0.0058 0.018 0.0094 0.0045 -0.028 0.0079 -0.028 1 -0.006 -0.03 0.16 -0.093 -0.052 0.083 -0.028
month -0.042 -0.093 -0.005 -0.057 0.011 0.02 0.27 0.021 0.36 -0.006 1 0.0063 -0.11 0.033 0.023 -0.033 -0.024
duration -0.0046 0.0047 0.01 -0.0014 -0.01 0.022 0.0051 -0.011 -0.021 -0.03 0.0063 1 -0.085 -0.0016 0.0012 0.011 0.39
campaign 0.0048 0.0068 -0.0093 0.0058 0.017 -0.015 -0.024 0.0076 0.02 0.16 -0.11 -0.085 1 -0.089 -0.033 0.1 -0.073
pdays -0.024 -0.024 0.022 0.0038 -0.03 0.0034 0.12 -0.02 -0.24 -0.093 0.033 -0.0016 -0.089 1 0.45 -0.86 0.1
previous 0.0013 -0.00091 0.018 0.02 -0.018 0.017 0.037 -0.0065 -0.15 -0.052 0.023 0.0012 -0.033 0.45 1 -0.49 0.093
poutcome 0.0074 0.011 -0.02 -0.021 0.035 -0.021 -0.1 0.012 0.27 0.083 -0.033 0.011 0.1 -0.86 -0.49 1 -0.078
y 0.025 0.04 0.048 0.068 -0.022 0.053 -0.14 -0.061 -0.15 -0.028 -0.024 0.39 -0.073 0.1 0.093 -0.078 1
age job marital education default balance housing loan contact day month duration campaign pdays previous poutcome y
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Correlation grid after removing ‘pdays’:



From the above correlation matrix obtained, we can see that the correlation between all features are very less, and that there will be no correlation among the features.

ALGORITHMS AND TECHNIQUES:

For this project following classifiers have been used:

- Logistic Regression
- Decision Tree Classifier:
    - Decision Trees automatically finds the most important attributes (feature selection) from the original set of attributes since the top most nodes on which the tree is split are essentially the most important variables within the dataset. Decision trees require relatively little effort from users for data preparation since each node is independent of the other in terms of the magnitude (value) of the attribute. Decision Trees are also very good at finding non-linear relationships.
    - Decision Trees can be extremely sensitive to small perturbations in the data: a slight change can result in a drastically different tree. They can easily overfit and hence aren't smooth.
    - Decision Trees is a good candidate for the problem since we aren't sure of the data is linearly separable just by looking at the data. Moreover, about half of the features are not numeric (even before one-hot encoding) so it is better to use Decision Trees since it works good with mixed data.
- Random Forest Classifier:
    - Random Forest inherits most of the advantages of a Decision Tree. Moreover, there is a reduction in overfitting since several trees are averaged and hence there is a significantly lower risk of overfitting. And there is less variance due to the use of multiple trees.
    - Random Forests are slow to create since there are number of Decision Trees are required to be created. Moreover, results of learning are incomprehensible. Compared to a single decision tree, or to a set of rules, they don't give you a lot of insight
    - Random Forest is a good candidate for the problem since we aren't sure of the data is linearly separable just by looking at the data. Moreover, half of the features are not numeric (even before one-hot encoding) so it is better to use Decision Trees since it works fine with mixed data.
    - The data is slightly unbalanced and Random Forests are known to deal really well with imbalanced data.
- AdaBoost Classifier:
    - An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.
    - Logistic Regression and Decision Tree Classifiers have been used as base estimators for AdaBoost classification purposes.

- Support Vector Machine:
    - Support Vector Machines arecapabletoproduce decision boundaries (linear or non-linear).
    - Support Vector Machine has many advantages.They haveregularization parameter that helps avoid over-fitting.It uses thekernel trick and tries to maximize the margin to get better results for test data.
    - Support Vector Machines have disadvantages as well.choosing appropriately hyper parameters of the SVM that will allow for sufficient generalization performance. Moreover, kernel models can be quite sensitive to over-fitting the model selection criterion as choosing the appropriate kernel function can be tricky.
    - Support Vector Machines is a suitable candidate because it canfind non- linear decision boundaries for classification.Moreover,they work well when dealing with nominal (binary) attributes.

For each classifier, the hyperparameter class_weight = 'balanced' has been used to make sure that class imbalance is accounted for in every classification task. Moreover, the hyperparameter stratify has been used during train-test aplit to make sure that the equal percentage of each class is divided among training and testing dataset. Similarly, stratified K-fold function approach has been used to ensure the same for cross validation.

DATA PREPROCESSING:

A). THE DATA WE HAVE TAKEN HERE FOR CLASSIFICATION HAS NO NULL VALUE AND IT IS ALREADY PREPROCESSED. SO, TO USE DATA CLEANING AND DATA FILLING PART, WE USE A CODE THAT RANDOMLY REMOVES SOME VALUES FROM THE RANDOMLY SELECTED COLUMN SO THAT WE CAN USE DATA FILLING METHOD (FILLNA METHOD) TO FILL THE 'MODE' VALUE IN THE NULL COLUMN. THIS WILL DEMONSTRATE THE DATA FILLING PART.

B). NUMERIC ATTRIBUTES ARE NORMALISED USING MinMaxScaler FUNCTION.

C). BINARY ATRIBUTES ARE CONVERTED TO EITHER 0 OR 1 USING LABELENCODER FUNCTION.

D). CATEGORICAL ATTRIBUTES (NON BINARY) ARE CONVERTED TO MULTIPLE FEATURES, DEPENDING ON NUMBER OF CATEGORIES FOR THAT ATTRIBUTE, USING GET_DUMMIES FUNCTION WHICH BASICALLY APPLIES ONE-HOR ENCODING TO THE ATTRIBUTES AND REPLACE THEM WITH NEW SET OF ONE HOT ENCODED FEATURES.

FEATURE TRANSFORMATION:

Principal Component Analysis was used for feature transformation (or dimensionality reduction). This also resulted in reducing the F1-Score for all the classifiers

IMPLEMENTATION :

For implementation, all the dependencies were loaded in the beginning that were required for Data preprocessing, classification and metric evaluation.

For data preprocessing (as discussed above), pandas library was used to store the data in the data-frame and apply transformations (including One Hot Encoding). Then, correlation was checked among the features by calculating the correlation matrix and it was found that a feature pair had a high negative correlation (in case of first dataset) and 5 feature pairs has positive correlation (in case of second dataset) so those features were removed. Finally, the preprocessing sub-library of sklearn was used for normalization and Label Encoding.

Then for each classifier mentioned in Algorithms and Techniques section , GridSearchCV function was used to apply cross validation on different combinations of parameters.

As discussed earlier, F1-Score and Area under Curve was as the evaluation metric. Results were stored in a common data-frame to later compare those results. Later, Accuracy was added as a metric to show how m is leading it can be. Finally, all the classifiers were compared with each other on the basis of F1-Score and Area under Curve. It was made sure that the test data is used only when the best parameters for the classifier were obtained from GridSearchCV.

REFINEMENT:

Initially, one hot encoding was applied to all the categorical attributes and later attributes were simply normalised. Then all the classifiers were looped and GridSearchCV was applied with a very small set of parameters.

After normalising the features, we can use train_test_split() method from sklearn's model_selection library to split our sample dataset into training and testing part with 80% training data and 20% testing data and we use StratifiedKFold technique for cross validation.

IN THE GRIDSEARCHCV METHOD, FOLLOWING PARAMETER VALUE IS USED :

1). LOGISTIC REGRESSION:

```
params = [
        {
           "penalty" : ['l2'],
           "C" : [0.01, 0.1, 1.0, 10.0],
           "solver" : ["sag"],
           "max_iter" : [100, 200]
        },
        {
           "penalty" : ['l1'],
           "C" : [0.01, 0.1, 1.0, 10.0],
           "solver" : ["saga"],
           "max_iter" : [100, 200]
        }
]
clf = GridSearchCV(estimator = LogisticRegression(n_jobs = -1, class_weight = "balanced"),
param_grid = params, cv = cv,
             n_jobs = -1, scoring = "f1")
clf.fit(X_train, y_train)
```

After applying gridsearchCV method, following is the best parameter for logistic regression:

```
{'C': 1.0, 'max_iter': 100, 'penalty': 'l2', 'solver': 'sag'}
```

Following is the result obtained after performing Logistic Regression:

```
F1-Score on test data is : 0.548
Area under ROC on test data is : 0.834
Accuracy on test data is : 0.841
```

Summarising the result:

```
Report :
               precision    recall  f1-score   support

           0       0.97      0.84      0.90      3992
           1       0.41      0.83      0.55       529

    accuracy                           0.84      4521
   macro avg       0.69      0.83      0.73      4521
weighted avg       0.91      0.84      0.86      4521
```

<u>2). DECISION TREE:</u>

```
params = {
        "criterion" : ["gini", "entropy"],
        "splitter" : ["best", "random"],
        "max_depth" : [25, 50, 75, None],
        "min_samples_split" : [3, 6],
        "min_samples_leaf" : [2, 4],
    }
clf = GridSearchCV(estimator = DecisionTreeClassifier(class_weight = "balanced", presort =
True), param_grid = params, cv = cv,
            n_jobs = -1, scoring = "f1")
clf.fit(X_train, y_train)
```

After applying gridsearchCV method, following is the best parameter for Decision Tree :

```
{'criterion': 'gini',
 'max_depth': 25,
 'min_samples_leaf': 2,
 'min_samples_split': 6,
 'splitter': 'random'}
```

Following is the result obtained after performing Decision tree algorithm:

```
F1-Score on test data is : 0.580
Area under ROC on test data is : 0.815
Accuracy on test data is : 0.875
```

Summarising the result:

```
Report :
              precision    recall  f1-score   support

           0       0.96      0.89      0.93      3992
           1       0.48      0.74      0.58       529

    accuracy                           0.88      4521
   macro avg       0.72      0.82      0.75      4521
weighted avg       0.91      0.88      0.89      4521
```

3). RANDOM FOREST :

```
params = {
        "n_estimators" : [100, 200],
        "criterion" : ["gini", "entropy"],
        "max_depth" : [15,30,45],
        "min_samples_split" : [3, 6],
        "min_samples_leaf" : [2, 4]
    }
clf = GridSearchCV(estimator = RandomForestClassifier(class_weight = "balanced",
n_jobs = -1), param_grid = params, cv = cv, n_jobs = -1, scoring = "f1")
clf.fit(X_train, y_train)
```

After applying gridsearchCV method, following is the best parameter for Random forest algorithm:

```
{'criterion': 'gini',
 'max_depth': 45,
 'min_samples_leaf': 4,
 'min_samples_split': 6,
 'n_estimators': 200}
```

Following is the result obtained after performing Random forest algorithm :

```
F1-Score on test data is : 0.660
Area under ROC on test data is : 0.858
Accuracy on test data is : 0.904
```

Summarising the result:

```
Report :
               precision    recall  f1-score   support

           0       0.97      0.92      0.94      3992
           1       0.56      0.80      0.66       529

    accuracy                           0.90      4521
   macro avg       0.77      0.86      0.80      4521
weighted avg       0.92      0.90      0.91      4521
```

4). SUPPORT VECTOR MACHINE:

```
params = [
        {
          "C" : [0.01, 0.1, 1.0],
          "kernel" : ["poly"],
          "degree" : [2, 3, 4],
          "gamma" : [0.001, "auto", 0.1]
        },
        {
          "C" : [0.01, 0.1, 1.0],
          "kernel" : ["rbf", "sigmoid"],
          "gamma" : [0.001, "auto", 0.1]
        },
        {
          "C" : [0.01, 0.1, 1.0],
          "kernel" : ["linear"]
        }
      ]
clf = GridSearchCV(estimator = SVC(class_weight = "balanced", max_iter = 10000,
verbose = True), param_grid = params,
             cv = cv, n_jobs = -1, scoring = "f1", verbose = 2)
clf.fit(X_train, y_train)
```

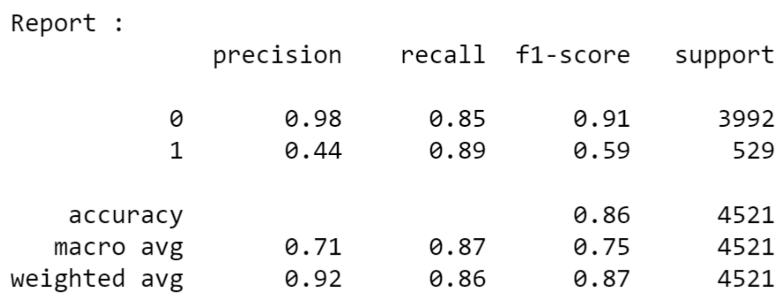
After applying gridsearchCV method, following is the best parameter for Support Vector machine algorithm :

```
{'C': 1.0, 'degree': 3, 'gamma': 0.1, 'kernel': 'poly'}
```

Following is the result obtained after performing Support Vector machine algorithm :

```
F1-Score on test data is : 0.590
Area under ROC on test data is : 0.871
Accuracy on test data is : 0.855
```

Summarising the result:



```
Report :
              precision    recall  f1-score   support

           0       0.98      0.85      0.91      3992
           1       0.44      0.89      0.59       529

    accuracy                           0.86      4521
   macro avg       0.71      0.87      0.75      4521
weighted avg       0.92      0.86      0.87      4521
```

COMPARING ALL THE METRICS AFTER PERFORMING ALL THE MODEL ALGORITHM:

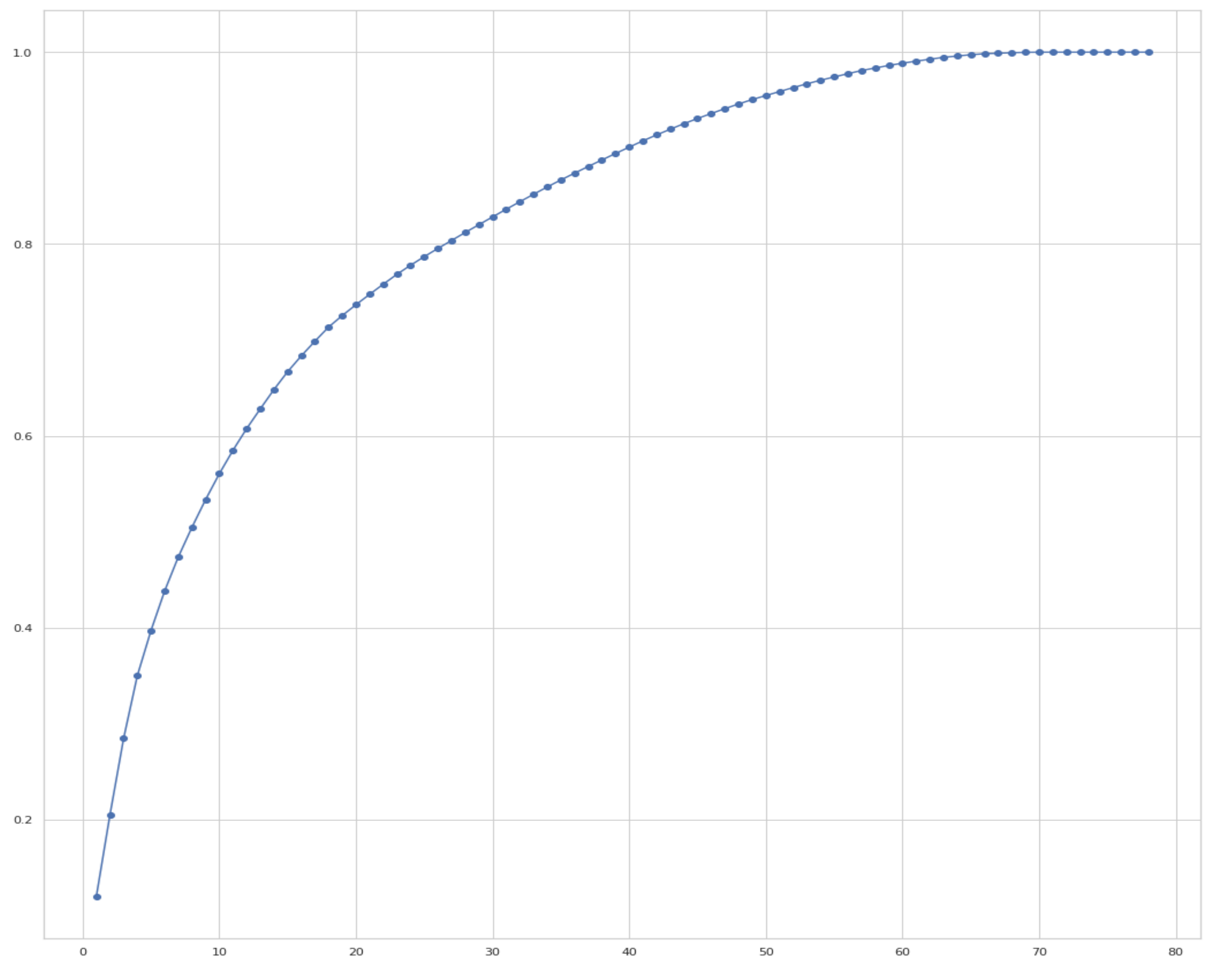
	CLASSIFIER	**F1-SCORE**	**AUC**	**ACCURACY**
0	SUPPORT VECTOR MACHINE	0.590	0.871	0.855
1	RANDOM FOREST	0.660	0.858	0.904
2	DECISION TREE	0.580	0.815	0.875
3	LOGISTIC REGRESSION	0.548	0.834	0.841

MODEL AFTER PRINCIPAL COMPONENT ANALYSIS:

After evaluating our model by different algorithm, we now use PCA algorithm for Dimensionality Reduction to check which features to retain and which to eliminate and check if their will be be some improvement in parameters.

For **Principal Component Analysis** sklearn's PCA function was used to get the most contributing components.

As shown in the below figure, we can observe that first 50 component amount to a cumulative of 0.95 of the total variance. So, we can take into account, the first 50 component to get 95% accuracy with less number of features and we can then observe that if the accuracy is improved or not.



Analysing the model training after applying PCA :

1). LOGISTIC REGRESSION:

```
params = [
        {
          "penalty" : ['l2'],
          "C" : [0.01, 0.1, 1.0, 10.0],
          "solver" : ["sag"],
          "max_iter" : [100, 200]
        },
        {
          "penalty" : ['l1'],
          "C" : [0.01, 0.1, 1.0, 10.0],
          "solver" : ["saga"],
          "max_iter" : [100, 200]
        }
]
clf = GridSearchCV(estimator = LogisticRegression(n_jobs = -1, class_weight =
"balanced"), param_grid = params, cv = cv,
            n_jobs = -1, scoring = "f1")
clf.fit(X_train, y_train)
```

After applying gridsearchCV method, following is the best parameter for Logistic Regression :

```
{'C': 0.1, 'max_iter': 100, 'penalty': 'l2', 'solver': 'sag'}
```

Following is the result obtained after performing for Logistic Regression :

```
F1-Score on test data is : 0.515
Area under ROC on test data is : 0.799
Accuracy on test data is : 0.834
```

Summarising the result:

```
Report :
               precision    recall  f1-score   support

           0       0.96      0.84      0.90      3992
           1       0.39      0.75      0.51       529

    accuracy                           0.83      4521
   macro avg       0.68      0.80      0.71      4521
weighted avg       0.90      0.83      0.85      4521
```

<u>2). DECISION TREE:</u>

```
params = {
        "criterion" : ["gini", "entropy"],
        "splitter" : ["best", "random"],
        "max_depth" : [15, 20, 25, None],
        "min_samples_split" : [3, 6],
        "min_samples_leaf" : [2, 4],
    }
clf = GridSearchCV(estimator = DecisionTreeClassifier(class_weight = "balanced",
presort = True), param_grid = params, cv = cv,
            n_jobs = -1, scoring = "f1")
clf.fit(X_train, y_train)
```

After applying gridsearchCV method, following is the best parameter for Decision Tree algorithm :

```
{'criterion': 'gini',
 'max_depth': 15,
 'min_samples_leaf': 2,
 'min_samples_split': 3,
 'splitter': 'best'}
```

Following is the result obtained after performing for Decision Tree algorithm :

```
F1-Score on test data is : 0.492
Area under ROC on test data is : 0.770
Accuracy on test data is : 0.834
```

Summarising the result:

```
Report :
              precision    recall  f1-score   support

           0       0.95      0.85      0.90      3992
           1       0.38      0.69      0.49       529

    accuracy                           0.83      4521
   macro avg       0.67      0.77      0.70      4521
weighted avg       0.89      0.83      0.85      4521
```

<u>3). RANDOM FOREST:</u>

```
params = {
        "n_estimators" : [100, 200],
        "criterion" : ["gini", "entropy"],
        "max_depth" : [15, 25, 35],
        "min_samples_split" : [3, 6],
        "min_samples_leaf" : [2, 4]}
clf = GridSearchCV(estimator = RandomForestClassifier(class_weight = "balanced",
n_jobs = -1), param_grid = params, cv = cv,
             n_jobs = -1, scoring = "f1")
clf.fit(X_train, y_train)
```

After applying gridsearchCV method, following is the best parameter for Random forest :

```
{'criterion': 'entropy',
 'max_depth': 15,
 'min_samples_leaf': 4,
 'min_samples_split': 3,
 'n_estimators': 100}
```

Following is the result obtained after performing for Random forest :

```
F1-Score on test data is : 0.544
Area under ROC on test data is : 0.756
Accuracy on test data is : 0.884
```

Summarising the result:

```
Report :
              precision    recall  f1-score   support

           0       0.94      0.92      0.93      3992
           1       0.50      0.59      0.54       529

    accuracy                           0.88      4521
   macro avg       0.72      0.76      0.74      4521
weighted avg       0.89      0.88      0.89      4521
```

4). SUPPORT VECTOR MACHINE:

```
params = [
        {"C" : [0.3, 1.0, 3.0], "kernel" : ["poly"],
          "degree" : [2, 3, 4],
          "gamma" : ["auto", 0.1, 0.3]
        },
        {"C" : [0.3, 1.0, 3.0], "kernel" : ["rbf", "sigmoid"],
          "gamma" : ["auto", 0.1, 0.3]},
        {"C" : [0.3, 1.0, 3.0], "kernel" : ["linear"]}
      ]
clf = GridSearchCV(estimator = SVC(class_weight = "balanced", max_iter = 5000,
verbose = True), param_grid = params,
            cv = cv, n_jobs = -1, scoring = "f1", verbose = 2)
clf.fit(X_train, y_train)
```

After applying gridsearchCV method, following is the best parameter for Support Vector machine algorithm :

```
{'C': 1.0, 'gamma': 0.3, 'kernel': 'rbf'}
```

Following is the result obtained after performing Support Vector machine algorithm :

```
F1-Score on test data is : 0.538
Area under ROC on test data is : 0.845
Accuracy on test data is : 0.825
```

Summarising the result:

```
Report :
              precision    recall  f1-score   support

           0       0.98      0.82      0.89      3992
           1       0.39      0.87      0.54       529

    accuracy                           0.83      4521
   macro avg       0.68      0.85      0.72      4521
weighted avg       0.91      0.83      0.85      4521
```

COMPARING ALL THE METRICS AFTER PERFORMING ALL THE MODEL ALGORITHM:

	CLASSIFIER	**F1-SCORE**	**AUC**	**ACCURACY**
0	SUPPORT VECTOR MACHINE	0.5382	0.8451	0.8250
1	RANDOM FOREST	0.5435	0.7564	0.8840
2	DECISION TREE	0.4922	0.7700	0.8343
3	LOGISTIC REGRESSION	0.5148	0.7992	0.8336

RESULT :

After evaluating all the model building directly and after dimensionality reduction, we obtained the following final result:

- For the amount of data we have, the amount of computation is very huge, when it comes to GridSearchCV method for classifiers like SVM, ADABoost etc., so we are not able to test all the hyperparameter combination.
- Out of all the algorithm we have trained the model, Random Forest Classifier seems to be the best model for the given parameters and datasets.
- The accuracy obtained in Random Forest Classifier is highest among all the techniques.
- The model trained without applying Dimensionality Reduction is better as compared to model after PCA, so we can say that instead of improvement in accuracy in dimensionality reduction, it has decreased significantly. So we can say that more features can be engineered for the given data.
- Instead of using hyperparameter class_weight = “balanced”, we can also try SMOTE (Synthetic Minority Over-Sampling Technique) sampling, as will create synthetic example using the dataset.

Following is the comparison of all the technique with and without PCA :

Without PCA:

	CLASSIFIER	F1-SCORE	AUC	ACCURACY
0	SUPPORT VECTOR MACHINE	0.590	0.871	0.855
1	RANDOM FOREST	0.660	0.858	0.904
2	DECISION TREE	0.580	0.815	0.875
3	LOGISTIC REGRESSION	0.548	0.834	0.841

With PCA:

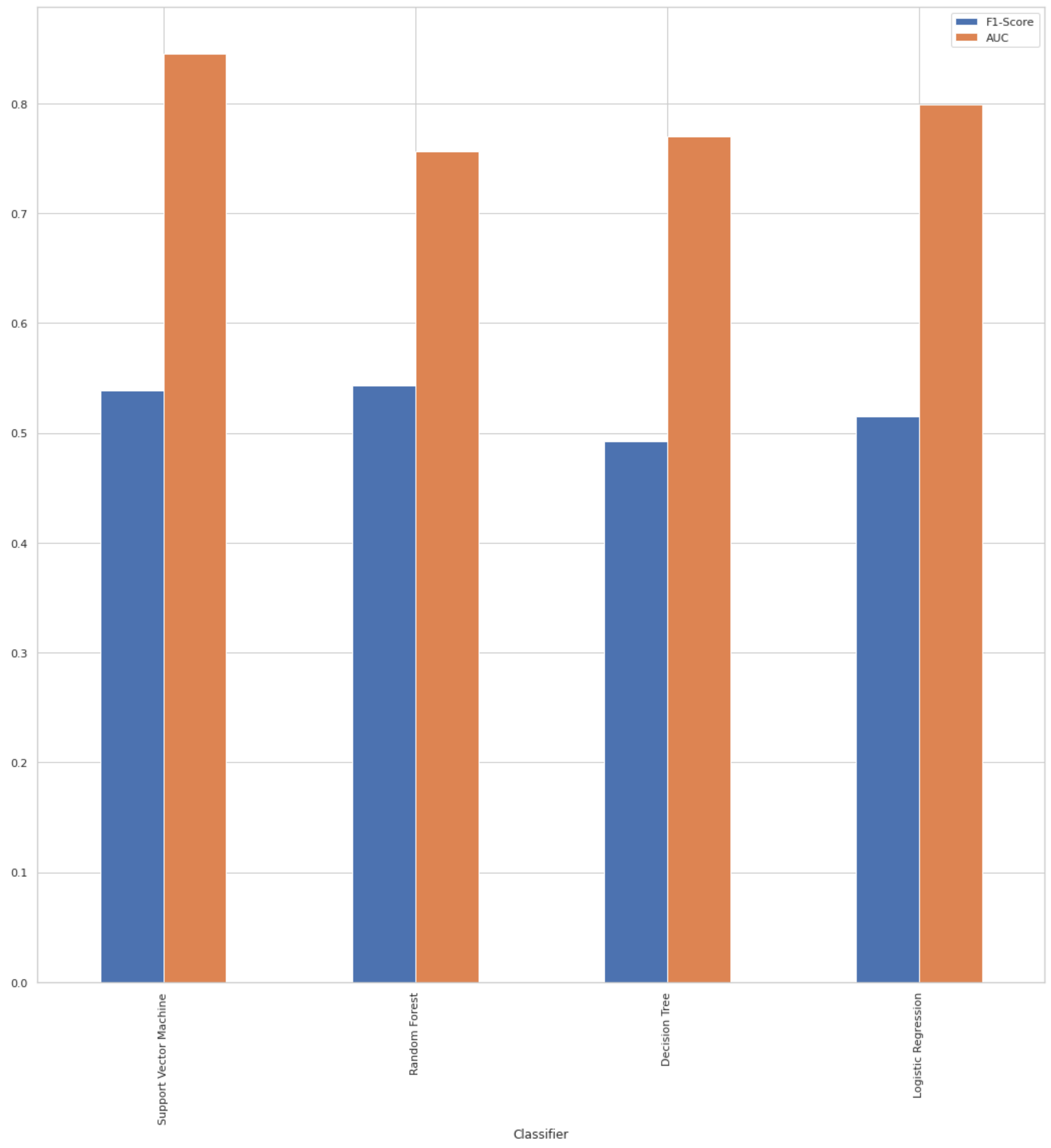
	CLASSIFIER	F1-SCORE	AUC	ACCURACY
0	SUPPORT VECTOR MACHINE	0.5382	0.8451	0.8250
1	RANDOM FOREST	0.5435	0.7564	0.8840
2	DECISION TREE	0.4922	0.7700	0.8343
3	LOGISTIC REGRESSION	0.5148	0.7992	0.8336

These both table shows that Dimensionality Reduction didn’t improve the accuracy on our model.

Free-Form Visualization:

For each dataset, a bar plot has been created to compare the results of all the classifiers. Although, transformation were applied but it didn't improve the F1-scores and the Area under ROC. So , weplotted the results for classification on original set of features.

Comparison of F1-score and AUC of all the techniques that we have used :



CONCLUSION:

We can see from the above result that feature reduction fails to improve the classification scores.

And as far the original features, RANDOM FOREST classifier turns out to be the best classifier in terms of F1-Score and Area Under ROC curve.

Lessons Learnt from Project:

- Understanding of the data according to problem statement.
- Data cleaning and Exploratory Data Analysis.
- Data Pre-processing
- How to apply GridSearchCV for specific model.
- Use of different Machine Learning Models.
- Interpretation of different models using confusion matrix like AUC, accuracy, f1-score.