1. **INTRODUCTION**

Our project on ‘*Marketing: To predict if the client will subscribe a bank term deposit or not*’ is based on the analysis of a given database to predict or compute the final result and suggest if the client will subscribe the term deposit or not. Our project incorporates use of 5 different classification models, which include, ‘Logistic Regression’ ,’K-NN’,’SVM’(Support Vector Machine) ,’Decision Tree’ and ‘Random Forest Classification’. All the different classification methods are used on the Training set, to predict test result and to compute a confusion matrix. The result of all the different methods of classifications are later merged to generate a data frame of zeros. The mode of each row is computed, which is used to calculate the final test set and the accuracy of our result.

1. **OBJECTIVE**

Our project entitled ‘*Bank Marketing: To predict if the client will subscribe a bank term deposit or not’*, uses the data set in order to predict if a client will or will mot subscribe a bank deposit. The project uses five different classification methods on the training set in order to predict a test result and compute confusion matrix. The result of different classification methods is used to calculate a data frame of zeros. Finally, the mode of each row is used to calculate the final test set and to predict the accuracy of our result.

1. **METHODOLOGY/PROCEDURE**

**Importing dataset**

The first step to any data science project is to import our data.

Everything on the computer is stored in the file system. "Directories" is just another word for "folders", and the "working directory" is simply the folder we're currently in. Now that you know what our current working directory is and where the dataset is in our file system, we can specify the file path to it. After that we have imported the .csv file using pandas library. After that we have separated the dependent values(X) and independent values(y). While taking the dependent values, we have selected only those inputs which actually count neglecting some data like mobile number, month, contact, etc.

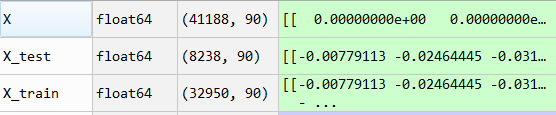
**Data Pre-processing**

Data Pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner.

Categorical variables are known to hide and mask lots of interesting information in a data set. Many machine learning algorithms can support categorical values without further manipulation but there are many more algorithms that do not. Categorical features can only take on a limited, and usually fixed, number of possible values. Our dataset contains some categorical data like job, education, housing loan, etc which are encoded and each categorical value has been assigned a certain dummy variable by label encoding and one hot encoding.

One important aspect of all machine learning models is to determine their accuracy. Now, in order to determine their accuracy, we have split the dataset into 2 parts: first one for training our machine learning model, and second one for testing our model. The train\_test\_split function takes several arguments which are explained below:

* X, y: These are the feature matrix and response vector which need to be split.
* test\_size: It is the ratio of test data to the given data. For example, setting test\_size = 0.2 means 20% of the dataset will be separated as test set and the rest 80% as training set
* random\_state: If you use random\_state = some\_number, then you can guarantee that your split will be always the same. This is useful if you want reproducible results, for example in testing for consistency in the documentation (so that everybody can see the same numbers).



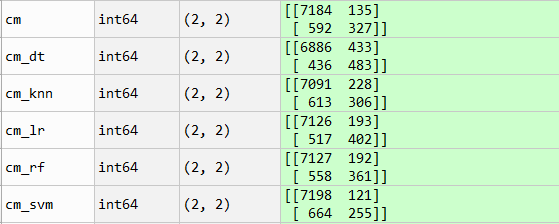
Our dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations, this is a problem. If left alone, these algorithms only take in the magnitude of features neglecting the units. To supress this effect, we need to bring all features to the same level of magnitudes. This we have achieved by feature scaling.

**Training the dataset**

Classification is a machine learning discipline of identifying the elements to their set or categories, on the basis of a training set data, where the membership/categories of the elements are known. Here we have used different classification models to get the predicted values. The different classification methods used are:

* Logistic Regression
* K-Nearest Neighbour
* Support Vector Machine
* Decision tree
* Random Forest Classification

Each classification technique gives a prediction and we have created a confusion matrix accordingly. A confusion matrix is nothing but a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix of different classification techniques are presented below:



1. **CODE**

# Importing the libraries

import pandas as pd

import numpy as np

# Importing the dataset

dataset = pd.read\_csv('bank-main.csv')

XX = dataset.iloc[:, [0,1,5,6,10,11,13,14,15,16,17,18,19]].values

X = pd.DataFrame(XX)

yy = dataset.iloc[:, 20].values

y=pd.DataFrame(yy)

# Encoding categorical data

# Encoding the Independent Variable

from sklearn.preprocessing import LabelEncoder,OneHotEncoder

labelencoder\_X = LabelEncoder()

for i in range (1,4):

X.values[:, i] = labelencoder\_X.fit\_transform(X.values[:, i])

X.values[:,7]= labelencoder\_X.fit\_transform(X.values[:, 7])

# Encoding the Dependent Variable

labelencoder\_y = LabelEncoder()

y = labelencoder\_y.fit\_transform(y)

#One-hot encoding

onehotencoder = OneHotEncoder(categorical\_features = [0])

X = onehotencoder.fit\_transform(X).toarray()

# Splitting the dataset into the Training set and Test set

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2,random\_state=0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting Logistic Regression to the Training set

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression()

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred\_lr = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm\_lr = confusion\_matrix(y\_test, y\_pred\_lr)

acc\_lr=(cm\_lr[0][0]+cm\_lr[1][1])/np.sum(cm\_lr)\*100

print("Accuracy of logistic regression is : ",acc\_lr)

# Fitting K-NN to the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred\_knn = classifier.predict(X\_test)

# Making the Confusion Matrix

cm\_knn = confusion\_matrix(y\_test, y\_pred\_knn)

acc\_knn=(cm\_knn[0][0]+cm\_knn[1][1])/np.sum(cm\_knn)\*100

print("Accuracy of K-Nearest Neighbour is : ",acc\_knn)

# Fitting svm classifier to the Training set

from sklearn.svm import SVC

classifier = SVC(kernel='rbf')

classifier.fit(X\_train,y\_train)

# Predicting the Test set results

y\_pred\_svm = classifier.predict(X\_test)

# Making the Confusion Matrix

cm\_svm = confusion\_matrix(y\_test, y\_pred\_svm)

acc\_svm=(cm\_svm[0][0]+cm\_svm[1][1])/np.sum(cm\_svm)\*100

print("Accuracy of Support Vector Machine is : ",acc\_svm)

# Fitting Decision Tree Classification to the Training set

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred\_dt = classifier.predict(X\_test)

# Making the Confusion Matrix

cm\_dt = confusion\_matrix(y\_test, y\_pred\_dt)

acc\_dt=(cm\_dt[0][0]+cm\_dt[1][1])/np.sum(cm\_dt)\*100

print("Accuracy of Decision tree classificationis : ",acc\_dt)

# Fitting Random Forest Classification to the Training set

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred\_rf = classifier.predict(X\_test)

# Making the Confusion Matrix

cm\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

acc\_rf=(cm\_rf[0][0]+cm\_rf[1][1])/np.sum(cm\_rf)\*100

print("Accuracy of Random Forest Classification is : ",acc\_rf)

#Merging the predictions

y\_predm=pd.DataFrame({'Logistic Regression':y\_pred\_lr,'K-Nearest Neighbour':y\_pred\_knn,'SVM':y\_pred\_svm,'Decision Tree Classification':y\_pred\_dt,'Random Forest':y\_pred\_rf})

#Generating a dataframe of zeroes

zero\_data=np.zeros(shape=(1,8238))

y\_pred=pd.DataFrame(zero\_data)

#Mode of each row

y\_pred=y\_predm.mode(axis=1)

y\_predict=y\_pred.iloc[:,0].values

#Overall Calculation

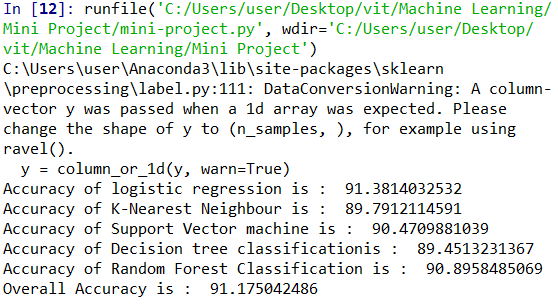
cm = confusion\_matrix(y\_test, y\_predict)

acc=(cm[0][0]+cm[1][1])/np.sum(cm)\*100

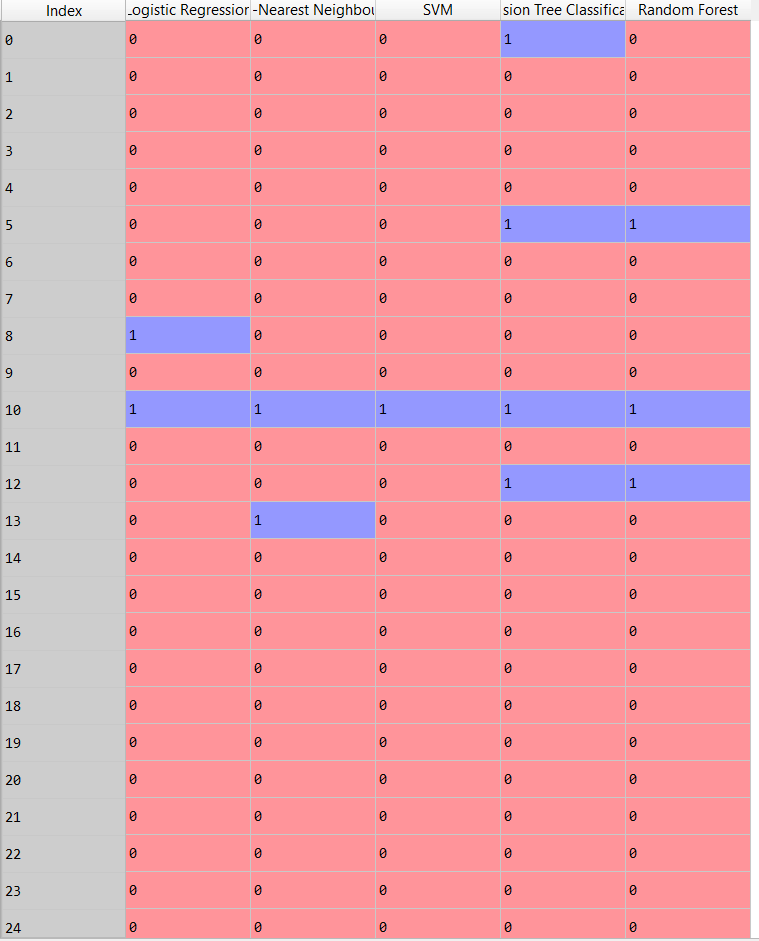
print("Overall Accuracy is : ",acc)

1. **RESULTS AND DISCUSSIONS**

From the confusion matrix we have calculated the accuracy of different classification techniques:



Here that we have set the random state while splitting the train set and test set, we can know which classification model is more appropriate. But for different random states we will get different accuracies which makes choosing a single classification technique more difficult. So in order to overcome this problem, we have made a table consisting all the predicted values and we have made a final prediction by taking mode of all the values and thus calculating the accuracy.



1. **CONCLUSION**

So, We are using different Machine Learning Algorithms to find whether a person will take the loan from the bank or not. So far we considered all the algorithms in Machine Learning. We are getting more than 90% Accuracy by using the following Algorithms(by taking Confusion Matrix into Consideration). Then we looked at the Scatter Plot of a randomly organized data and then we discussed and to make our Prediction more accurate, We took Mode of all the predicted values. So, by taking the Mode,We are getting better Accuracy and our new Predicted table is better than the table predicted by other individual Algorithms.

**7. References** (By Kiril Eremenko)

1. Udemy Machine Learning Course(

2. UCI repository