**Aim :-**

The Aim of this project is to generate models provided in the dataset using various classification and regression techniques in python. Analyze the results presents in the model and optimize model which is most suitable for this dataset.

**Dataset Description : -** Dataset contains information about information used for bank marketing. Bank Marketing dataset contains 21 variables and 4119 Rows. This Dataset is provided by a Portuguese banking institution based on bank marketing campaigns and based on real world problem.

**Training Dataset : 3089**

**Testing Dataset : 1030**

**Link to Dataset :**

**http://archive.ics.uci.edu/ml/datasets/Bank+Marketing/**

**Data Cleaning** :- Dataset contains 21 attributes in the training dataset and 20 in the test dataset & one attribute in the target test dataset**.** No N/A values found when summary of all dataset performed . Attribute List as follows :-

Input variables:  
# bank client data:  
1 - age (numeric)  
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')  
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)  
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')  
5 - default: has credit in default? (categorical: 'no','yes','unknown')  
6 - housing: has housing loan? (categorical: 'no','yes','unknown')  
7 - loan: has personal loan? (categorical: 'no','yes','unknown')  
# related with the last contact of the current campaign:  
8 - contact: contact communication type (categorical: 'cellular', 'telephone')   
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ., 'nov', 'dec')  
10 - day\_of\_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')  
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.  
# other attributes:  
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)  
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)  
14 - previous: number of contacts performed before this campaign and for this client (numeric)  
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')  
# social and economic context attributes  
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)  
17 - cons.price.idx: consumer price index - monthly indicator (numeric)   
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)   
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)  
20 - nr.employed: number of employees - quarterly indicator (numeric)  
Output variable (desired target):  
21 - y - has the client subscribed a term deposit? (binary: 'yes',' no')

Following code used for initial variable analysis :-

**# Read CSV file from the working directory**

**> datatry = pd.read\_csv('C:\\Users\\Kshitij\\Desktop\\bank-additional.csv',sep=';')**

**# file: File name   
 # sep: Field separator**

**# Determine Rows and columns of CSV file**

**datatry.shape()**

(4119, 21)

**# Determine statistical parameters (mean/median etc)**

**datatry.describe()**

**# Determine Dataset contain any null value**

**> datatry.isnull().sum()**

Returns each column contain any null value

**# importing necessary libraries of python**

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

from sklearn.neighbors import NearestNeighbors

from sklearn.neighbors import KNeighborsClassifier

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

from math import sqrt

**# splitting dataset into two part**

**# a 70% of training dataset**

**# b 30% testing dataset**

x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,y1,random\_state=1)

# **Finding unique value for the selected column**

**datatry['job'].unique()**

**# Converting chars dataset into numerical data**

Job\_column = {"job":{"blue-collar": 0, "services": 1,"admin.":2,"entrepreneur":3,"self-employed":4,"technician":5,"management":6,"student":7,"retired":8,"housemaid":9,"unemployed":10,"unknown":11}}

**datatry.replace(Job\_column,inplace=True)**

# **Repeat above step for all char Datacolumn**

**KNN Classification Algorithm :**

KNN Algorithm is one of the classification algorithm in which selected value K will be searched in the entire dataset which are nearest to the unknown

label. The most popular response value of K will be selected as predicted

response value of K. KNN can be described into below steps :-

1 Calculate the distance between current points and unknown label

2 sort all distances into increasing order

3 Take K items nearest to unknown lebel.

4.Find majority vote (class) among these items.

5.Return the majority class will be the class of unknown label.

**# prepare model with training dataset**

X will be the data frame of predictive variables

X =datatry[['age','marital','housing','loan','default','day\_of\_week','durat ion','campaign','pdays','cons.price.idx','euribor3m','euribor3m','nr.employed']]

**Y1 will be the response variable**

**y1 = datatry.y**

**Training and Testing dataset is taken from above mentioned classification algorithm - KNN**

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

**x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,y1,random\_state=1)**

**Determining the shape of testing and training datasets**

**x\_train.shape**

**(3089, 13)**

**x\_test.shape**

**(1030, 13)**

**y\_train.shape**

**(3089,)**

**y\_test.shape**

**(1030,)**

First dimension x\_train dataframe (of predictive variables) should be equal to the first dimension of the y\_train dataframe(consists of response variable)

**# Creating object of KNN Algorithm.**

**knn = KNeighborsClassifier()**

**model fit method will learn relationship between X\_train and y\_train dataframes**

**knn.fit(x\_train, y\_train)**

**KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',**

**metric\_params=None, n\_jobs=1, n\_neighbors=5, p=2,**

**weights='uniform')**

**predicting from KNN model using x\_test dataframe**

**y\_predict\_class= knn.predict(x\_test)**

**interpretation from KNN algorithm**

**Null Accuracy from KNN algorithm** : This will provide baseline to the model and determine predominate class in the dataset.

**from sklearn import metrics**

**print(metrics.accuracy\_score(y\_test,y\_predict\_class))**

0.91359223301

**# calculate null accuracy**

**y\_test.value\_counts()**

0 930

1 100

Name: y, dtype: int64

**y\_test.mean()**

0.0970873786407767

**1 - y\_test.mean()**

0.9029126213592233In

**max(y\_test.mean(),1-y\_test.mean())**

0.9029126213592233

**# null accuracy is 90.29 % less than 91.35 % is the model accuracy so this model does not looks good**

**cons of null accuracy :** it does not tell you underline distribution of data

and any underline noise in data. This disadvantages is eliminated by confusion metrics.

**#confusion Matrix**

**print(metrics.confusion\_matrix(y\_test,y\_predict\_class))**

[[891 39]

[ 50 50]]

created by actual observed values and predicted values of response variables.

metrics accuracy score = 891 + 50/891+39+50+50 = .9135 or 91.35%

**Classification error = 1- print(metrics.confusion\_matrix(y\_test,y\_predict\_class)) = .0865 or 8.6 %**

**sensitivity**

**metrics.recall\_score(y\_test,y\_predict\_class)**

recall means when actual value is positive then how much model is able to predict positive values

= 50/50+50 = .5 or 50%

#There is no library to calculate specificity using pandas so we need to calculate manually specificity means when actual value is negative then howoften model is able to predict negative values :

891/891+39 = .958 or 95.8 %

# false positive rate

# when actual value is negative how often model is incorrect

**#1 - specificity**

# 1- 95.8

# = 2.x2%

**Precision** : when positive value is predicted how often prediction is correct.

50/50+39 = 5617 = 56.17%

Predict first 10 values from test dataset

**knn.predict(x\_test)[0:10]**

array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0], dtype=int64)

**knn.predict\_proba(x\_test)[0:10,:]**

array([[ 0.8, 0.2],

[ 0.8, 0.2],

[ 0.2, 0.8],

[ 0.8, 0.2],

[ 1. , 0. ],

[ 1. , 0. ],

[ 1. , 0. ],

[ 0.8, 0.2],

[ 1. , 0. ],

[ 1. , 0. ]])

**metrics.mean\_absolute\_error(y\_test,y\_predict\_class)**

0.086407766990291263

its a difference between observed value and predicted value

**metrics.mean\_squared\_error(y\_test,y\_predict\_class)**

0.086407766990291263

it's a square of difference between observed value and predicted value

**from math import sqrt**

**sqrt(metrics.mean\_squared\_error(y\_test, y\_predict\_class))**

0.2939519807558562

its a square root of square ofdifference between observed value and predicted value.

**predicted\_proba = knn.predict\_proba(x\_test)[0:10,:1]**

**import matplotlib.pyplot as plt**

**plt.hist(predicted\_proba,bins=8)**

(array([ 1., 0., 0., 0., 0., 0., 4., 5.]),

array([ 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),

<a list of 8 Patch objects>)

**plt.title("Histogram of predicted probabilities")**

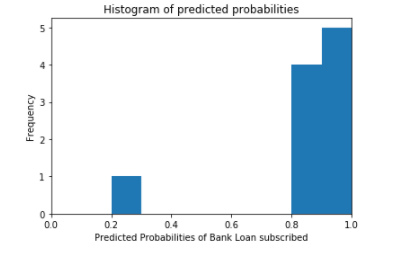
**plt.xlabel("Predicted Probabilities of Bank Loan subscribed")**

**plt.ylabel("Frequency")**

**plt.xlim(0,1)**

**plt.show()**

**Histogram of predicted probabilities from model**

****