Abstract:

My goal in this project is to detect if the client will subscribe or not using parameters, This data set contains records relevant to a direct marketing campaign of a Portuguese banking institution. The marketing campaign was executed through phone calls. more than one call needs to be made to a single client before they either decline or agree to a term deposit subscription. The classification goal is to predict if the client will subscribe (yes/no) to the term deposit (variable y).

Data:

There are 41188 client data and 20 columns in the data (csv file), each column will be parameters for classification and last column will be result data (subscribe or not).

some of features is Job, Marital, Education, etc.

In this project I used pandas to read csv file.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  campaign        41188 non-null  int64
 11  pdays           41188 non-null  int64
 12  previous        41188 non-null  int64
 13  poutcome        41188 non-null  object
 14  emp.var.rate    41188 non-null  float64
 15  cons.price.idx  41188 non-null  float64
 16  cons.conf.idx   41188 non-null  float64
 17  euribor3m       41188 non-null  float64
 18  nr.employed     41188 non-null  float64
 19  y               41188 non-null  object
dtypes: float64(5), int64(4), object(11)
memory usage: 6.3+ MB
```

These are columns in my csv file.

```
data.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

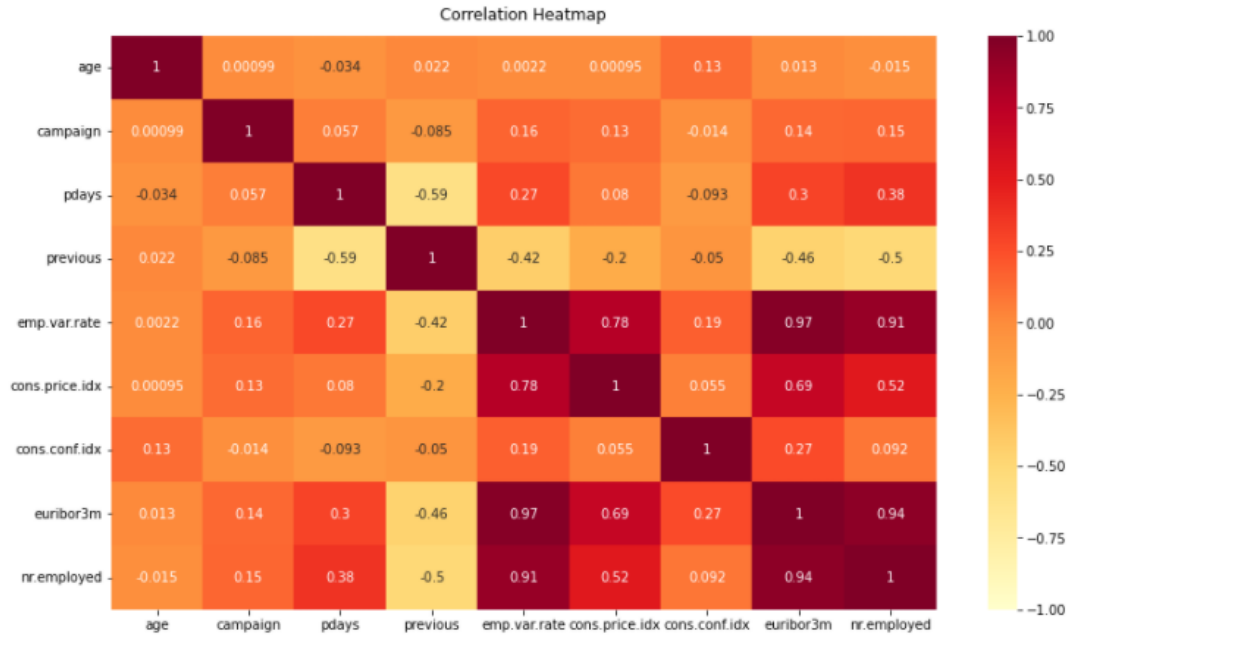
This is output for header of dataframe data.

```
data.isnull().sum()
```

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

In this case, we have to delete row that contains null value.

```
plt.figure(figsize=(14, 8))
heatmap = sns.heatmap(data.corr(),vmin=-1, vmax=1, annot=True,cmap='YlOrRd')
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);
plt.show()
```



This is heatmap for data correlation. We will set threshold as 0.3 in this project.

If correlation is less than 0.3, we will delete columns from my data and also too high correlation columns will be removed.

```
# delete some rows from threshold
corr = data.corr()

threshold = 0.3
haha = ['age', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']
result = []
for temp in haha:
  for t in haha:
    if corr[temp][t] >= threshold and corr[temp][t] < 0.9:
      result.append(temp)

result = list(set(result))
final = []
for temp in haha:
  if not temp in result:
    final.append(temp)

print(final)

data.drop(final, axis=1, inplace=True)
data.head()
```

```
['age', 'campaign', 'previous', 'cons.conf.idx']
```

	job	marital	education	default	housing	loan	contact	month	day_of_week	pdays	poutcome	emp.var.rate	cons.price.idx	euribor3m	nr.employed	y
0	housemaid	married	basic.4y	no	no	no	telephone	may	mon	999	nonexistent	1.1	93.994	4.857	5191.0	no
1	services	married	high.school	unknown	no	no	telephone	may	mon	999	nonexistent	1.1	93.994	4.857	5191.0	no
2	services	married	high.school	no	yes	no	telephone	may	mon	999	nonexistent	1.1	93.994	4.857	5191.0	no
3	admin.	married	basic.6y	no	no	no	telephone	may	mon	999	nonexistent	1.1	93.994	4.857	5191.0	no
4	services	married	high.school	no	no	yes	telephone	may	mon	999	nonexistent	1.1	93.994	4.857	5191.0	no

In this step, calculate each correlations and remove some rows. In this project, ‘age’, ‘campaign’, ‘previous’ and ‘cons.conf.idx’ columns will be removed.

```
le = preprocessing.LabelEncoder()
data1 = data.apply(le.fit_transform)
```

```
data1.head()
```

	job	marital	education	default	housing	loan	contact	month	day_of_week	pdays	poutcome	emp.var.rate	cons.price.idx	euribor3m	nr.employed	y
0	3	1	0	0	0	0	1	6	1	26	1	8	18	287	8	0
1	7	1	3	1	0	0	1	6	1	26	1	8	18	287	8	0
2	7	1	3	0	2	0	1	6	1	26	1	8	18	287	8	0
3	0	1	1	0	0	0	1	6	1	26	1	8	18	287	8	0
4	7	1	3	0	0	2	1	6	1	26	1	8	18	287	8	0

In this project, data is still string values, so that we need to convert as integer or float values.

I used LabelEncoder method to do.

And also result will be like that.

```
X = data1.iloc[:,0:15]
y = data1.iloc[:,15]

X.shape, y.shape
```

```
((39404, 15), (39404,))
```

Finally after cleaning the data, the shape of data is 39404, 15 and 39404, 1

```
# split train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=1, stratify=y)
```

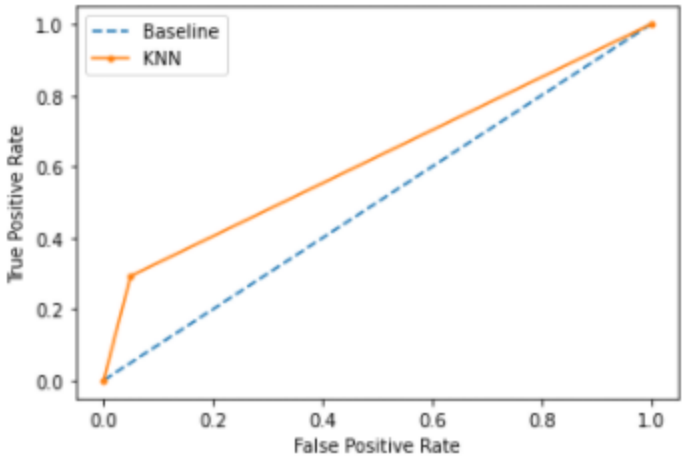
Before creating a model, we need to split data as train and test data.

In this project, I used the train_test_split() method, training data will be 75% of whole data, remaining data will be testing data.

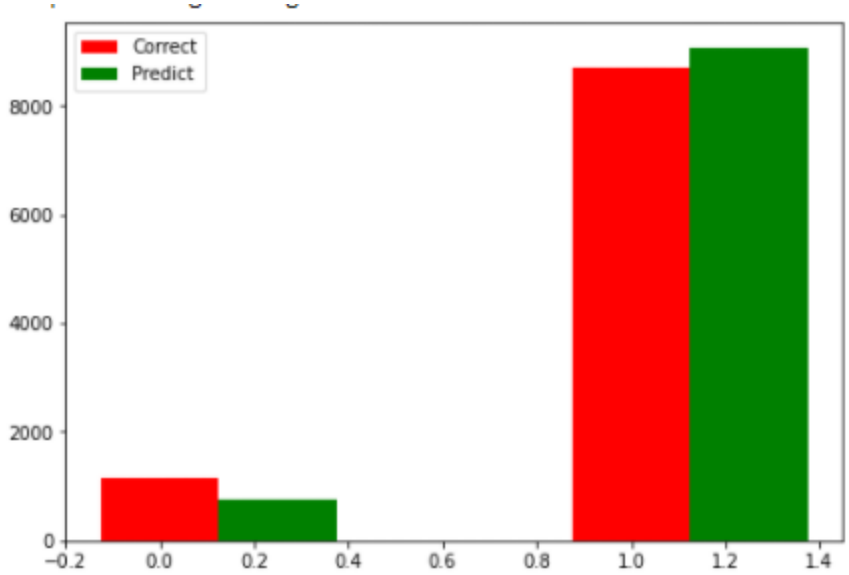
```
=====Confusion Matrix=====
[[8271  430]
 [ 813  337]]


=========Report==========
              precision    recall  f1-score   support

           0       0.91      0.95      0.93      8701
           1       0.44      0.29      0.35      1150

    accuracy                           0.87      9851
   macro avg       0.67      0.62      0.64      9851
weighted avg       0.86      0.87      0.86      9851
```

This is output for KNN model, Accuracy is 0.87

ROC/AUC
0.6218119356595694



Roc/Auc values is 0.621 and also chart for ROC

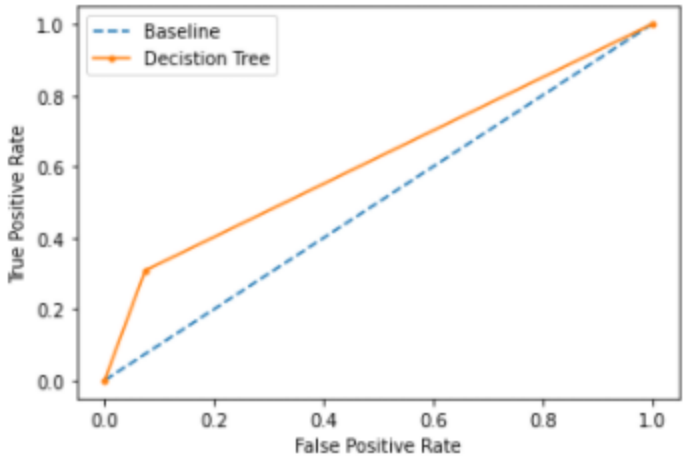


This is chart for correct and predict data, red column is number of correct data, blue column is number of predict data

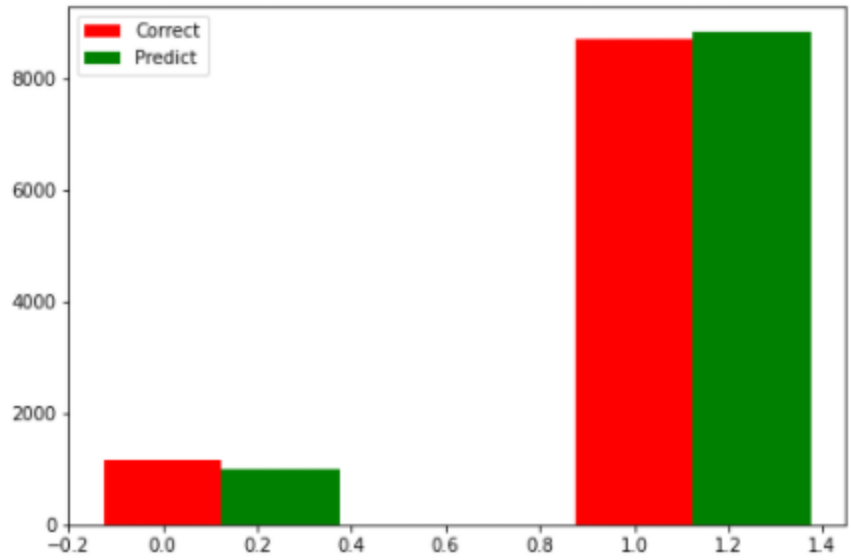
```
=====Confusion Matrix=====
[[8042  659]
 [ 805  345]]


=========Report==========
              precision    recall  f1-score   support

           0       0.91      0.92      0.92      8701
           1       0.34      0.30      0.32      1150

    accuracy                           0.85      9851
   macro avg       0.63      0.61      0.62      9851
weighted avg       0.84      0.85      0.85      9851
```

This is output for Decision Tree Classification

```
ROC/AUC
0.6172256562214239
```



Roc/AUC value is 0.617 and chart for ROC.

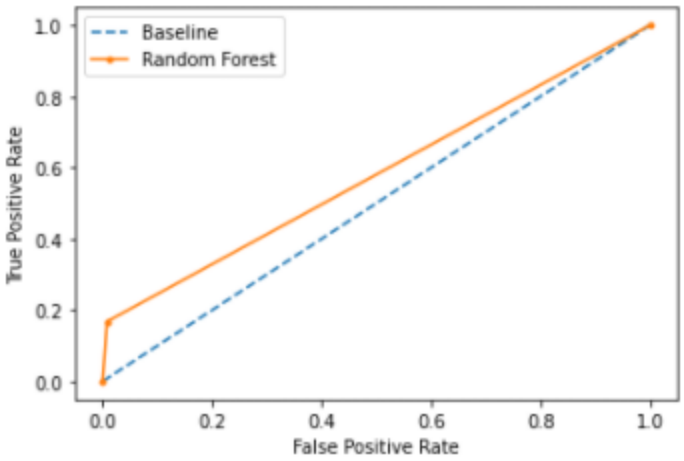


This is a chart for correct and predict data, red column is the number of correct data, blue column is the number of predict data

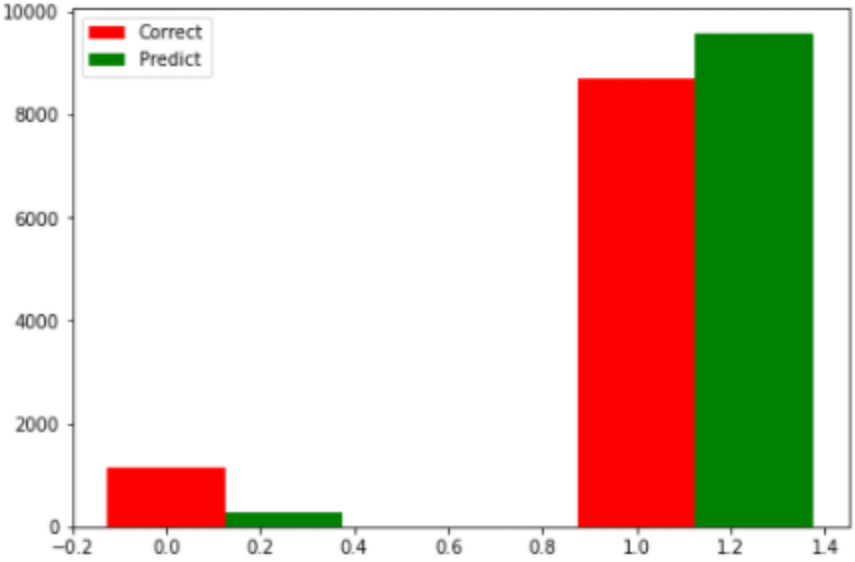
```
=====Confusion Matrix=====
[[8632   69]
 [ 957  193]]


=========Report==========
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      8701
           1       0.74      0.17      0.27      1150

    accuracy                           0.90      9851
   macro avg       0.82      0.58      0.61      9851
weighted avg       0.88      0.90      0.87      9851
```

This is output for Random Forest and accuracy is 0.9

ROC/AUC
0.5799479819910756



ROC/AUC value is 0.5799 and chart for ROC values.

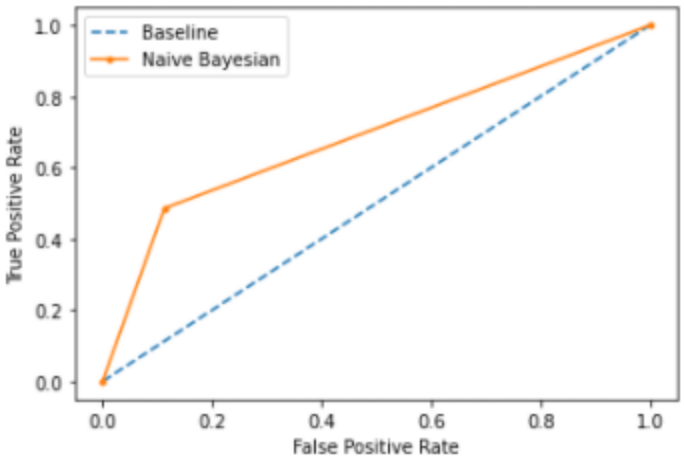


This is a chart for correct and predicted data, red column is number of correct data, blue column is number of predict data

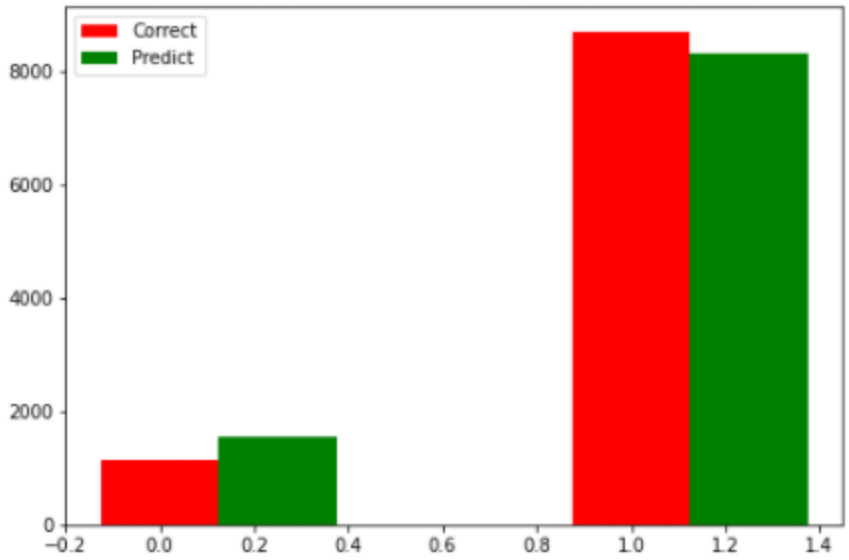
```
=====Confusion Matrix=====
[[7718  983]
 [ 591  559]]


=========Report==========
              precision    recall  f1-score   support

           0       0.93      0.89      0.91      8701
           1       0.36      0.49      0.42      1150

    accuracy                           0.84      9851
   macro avg       0.65      0.69      0.66      9851
weighted avg       0.86      0.84      0.85      9851
```

This is output for Naïve Bayesian classification and accuracy is 0.84

```
ROC/AUC
0.6865557182332864
```



ROC/AUC value is 0.686 and chart for ROC

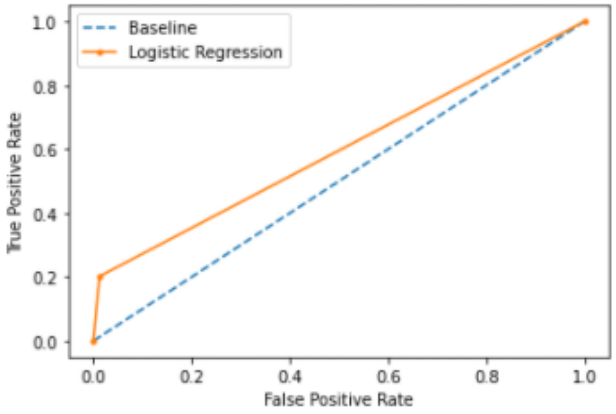


This is a chart for correct and predict data. The red column is the number of correct data, blue column is the number of predict data

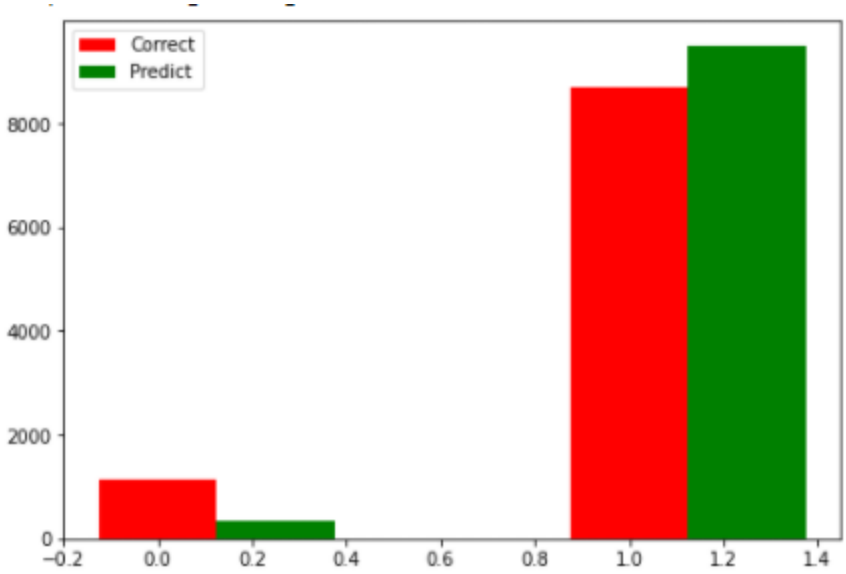
```
=====Confusion Matrix=====
[[8595  106]
 [ 918  232]]


=========Report==========
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      8701
           1       0.69      0.20      0.31      1150

    accuracy                           0.90      9851
   macro avg       0.79      0.59      0.63      9851
weighted avg       0.88      0.90      0.87      9851
```

This is output for Logistic Regression and accuracy is 0.9

```
ROC/AUC
0.5947783113385269
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```



ROC/AUC value is 0.5947 and chart for ROC values.



This is a chart for correct and predicted data. The red column is the number of correct data, blue column is the number of predict data

So in this project, Logistic regression and Random Forest model has same accuracy.

But logistic regression's roc is more than random forest model so that logistic regression model is best.

Python Libararies

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import metrics
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
```