
SDAIA ACADMEY T5 BOOTCAMPS: DATA SCIENCE

CLASSFICATION MODULE PROJECT

# Predict client subscription

By: Samer

# Table of contents

# Objective

This data set contains records relevant to a direct marketing campaign of a Portuguese banking institution. This marketing campaign was executed through phone calls.

This is classification project to predict client will subscribe or not using other parameters such as Job, Marital, Education, etc.

# Description of the Data

- Portuguese banking institution
- Kaggle
- 20 columns
- 41188 clients data

# Features

- 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

```
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 |

```
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
```

# Feature Engineering

```
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 |

```
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 |

# Data Preparation

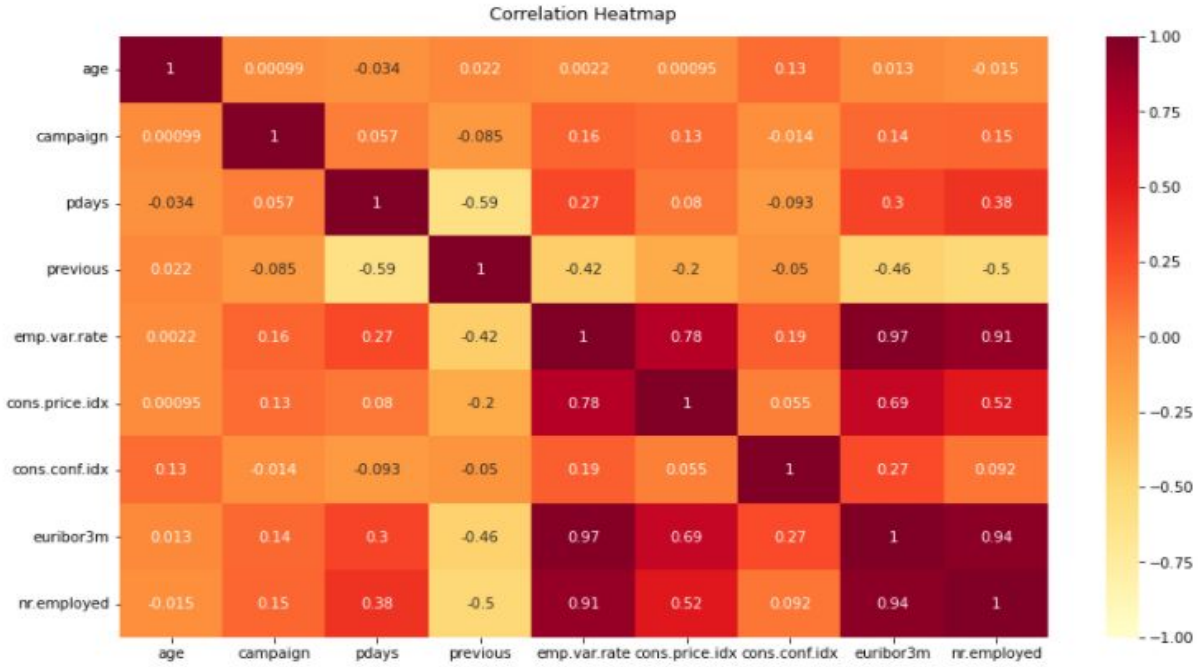
- Removing null and duplicate
- before (41188, 19)
- after (38771, 19)

```
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
```

# Correlation Heatmap

```
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()
```



# Removing

- Drop High correlation and Low correlation
- Age, campaign, previous and cons.conf.idx 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']
```

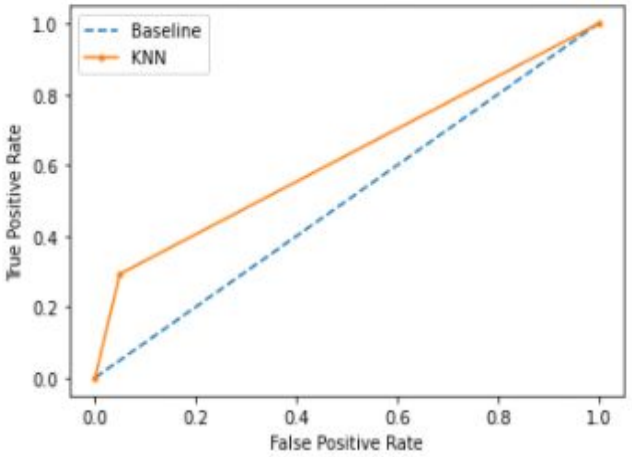
| | job | marital | education | default | housing | loan | contact | month | day_of_week | pdays | poutcome | emp.var.rate | cons.price.idx | euribor3m | nr.employed | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | housemaid | married | basic.4y | no | no | no | telephone | 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 |

# KNN model

```
=====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
```

ROC/AUC

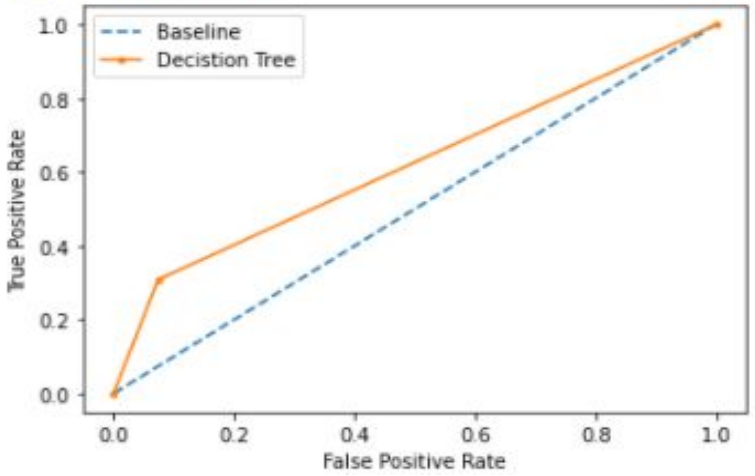
0.6218119356595694



# Decision Tree Classification

```
=====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
```

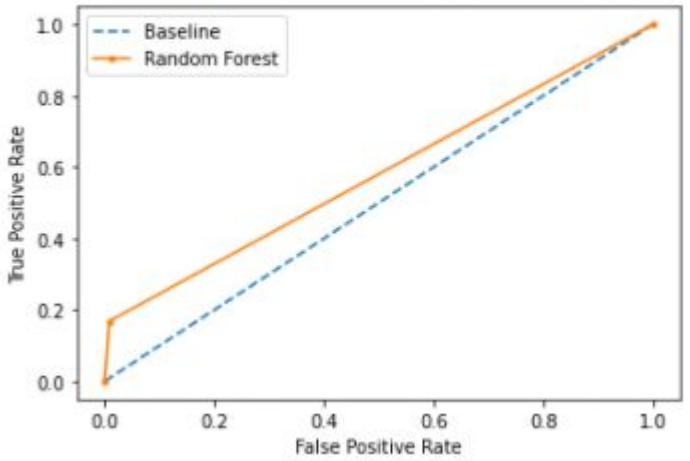
ROC/AUC
0.6172256562214239



# Random Forest

```
=====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
```

ROC/AUC
0.5799479819910756

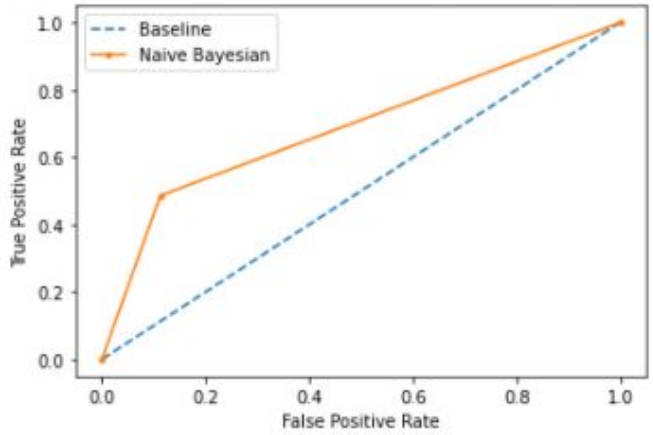


# Naive Bayes

```
=====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
```

ROC/AUC

0.6865557182332864

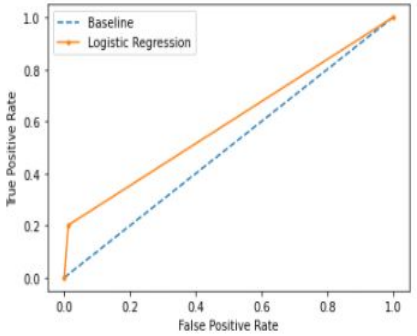


# Logistic Regression

```
=====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
```

```
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
```



# Conclusion

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.