Modelling

Dataset

```
# import the libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, LabelEncoder,
MinMaxScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
%run viz.py
%run classification.py
```

```
C:\Users\Saurabh\Anaconda3\lib\site-packages\dask\config.py:168:
YAMLLoadWarning: calling yaml.load() without Loader=... is deprecated,
as the default Loader is unsafe. Please read
https://msg.pyyaml.org/load for full details.
  data = yaml.load(f.read()) or {}
C:\Users\Saurabh\Anaconda3\lib\site-packages\distributed\config.py:20:
YAMLLoadWarning: calling yaml.load() without Loader=... is deprecated,
as the default Loader is unsafe. Please read
https://msg.pyyaml.org/load for full details.
  defaults = yaml.load(f)
```

```
# read the data
df = pd.read_csv('ml_dataset.csv')
df.head()
```

```
   age         worklass  fnlwgt  education  education-num  \
0   39         State-gov   77516  Bachelors             13
1   50  Self-emp-not-inc   83311  Bachelors             13
2   38           Private  215646    HS-grad              9
3   53           Private  234721       11th              7
4   28           Private  338409  Bachelors             13

       martial-status         occupation   relationship   race     sex
\
0       Never-married       Adm-clerical  Not-in-family  White    Male

1  Married-civ-spouse    Exec-managerial        Husband  White    Male

2            Divorced  Handlers-cleaners  Not-in-family  White    Male

3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male

4  Married-civ-spouse     Prof-specialty           Wife  Black  Female
```

```
   capital-gain  capital-loss  hours-per-week native-country income
id
0          2174             0              40  United-States  <=50K
0
1             0             0              13  United-States  <=50K
1
2             0             0              40  United-States  <=50K
2
3             0             0              40  United-States  <=50K
3
4             0             0              40           Cuba  <=50K
4
```

Understanding the variables

Let's understand the variables involved in modelling:

1. `id` : The unique id of the employee
2. `age` : The age of employee
3. `worklass` : The occupation type of the employee
4. `fnlwgt` : It refers to final weight. This is the number of people the census believes the entry represents
5. `education` : Education qualification of employee
6. `education-num` : Qualification with number
7. `martial-status` : Maritial status
8. `occupation` : Profession followed by the employee
9. `relationship` : Relation of the employee
10. `race` : Ethnicity
11. `sex` : Gender
12. `capital-gain` : Increase in the capital asset value of the employee
13. `capital-loss` : Decrease in the capital asset value of the employee
14. `hours-per-week` : Hours spent working per week
15. `native-country` : Origin country of the employee
16. `income` : This is the variable to be predicted either it's >50k or <50K

```python
# lets drop id column as it's irrelevant for modelling
df = df.drop('id', axis = 1)
```

```python
# Lets see the data
description(df).data_description(summary = True)
```

```
The number of points in this data is 32561

The shape of the data is (32561, 15)

Let's see the data :

The summary of data set is :
```

```
                age        fnlwgt  education-num  capital-gain
capital-loss  \
count  32561.000000  3.256100e+04   32561.000000  32561.000000
32561.000000
mean      38.581647  1.897784e+05      10.080679   1077.648844
87.303830
std       13.640433  1.055500e+05       2.572720   7385.292085
402.960219
min       17.000000  1.228500e+04       1.000000      0.000000
0.000000
25%       28.000000  1.178270e+05       9.000000      0.000000
0.000000
50%       37.000000  1.783560e+05      10.000000      0.000000
0.000000
75%       48.000000  2.370510e+05      12.000000      0.000000
0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000
4356.000000

       hours-per-week
count    32561.000000
mean        40.437456
std         12.347429
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000

The count of n.a values in each column is:
age               0
worklass          0
fnlwgt            0
education         0
education-num     0
martial-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

```
# let's see the target variable
df['income'].value_counts()
```

```
<=50K    24720
>50K      7841
Name: income, dtype: int64
```

We see the target variable is very **unbalanced**

```
## value counts
col  = ['worklass','native-country','occupation']
for i in col:
    print(df[i].value_counts())
    print()
```

```
Private             22696
Self-emp-not-inc     2541
Local-gov            2093
?                    1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: worklass, dtype: int64

United-States                 29170
Mexico                          643
?                               583
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Guatemala                        64
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
France                           29
```

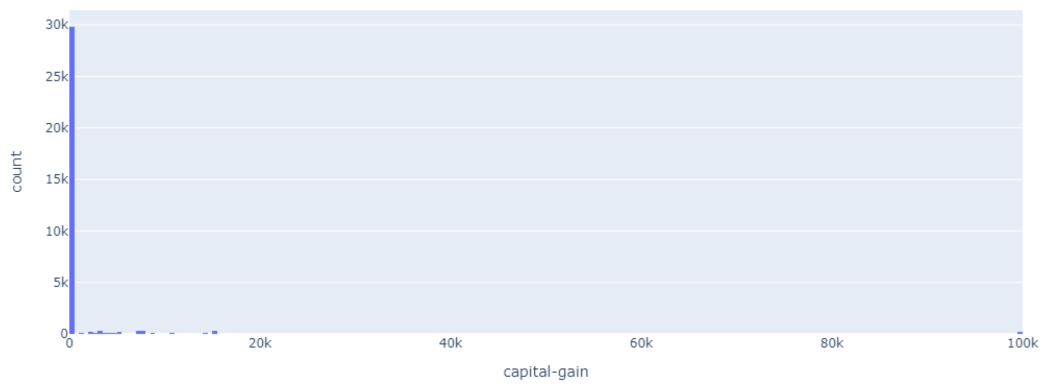
```
Greece                            29
Ecuador                           28
Ireland                           24
Hong                              20
Cambodia                          19
Trinadad&Tobago                   19
Thailand                          18
Laos                              18
Yugoslavia                        16
Outlying-US(Guam-USVI-etc)        14
Honduras                          13
Hungary                           13
Scotland                          12
Holand-Netherlands                 1
Name: native-country, dtype: int64

Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspct    2002
?                    1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: occupation, dtype: int64
```

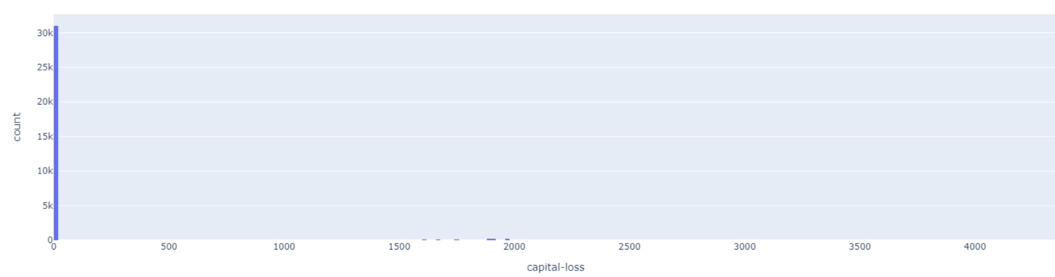
Since these values can't be dropped as they contribute significant count to the dataset. We will replace these values with the most occuring counts in the respective column i.e. **mode of the column**

```
for i in col:
     df.loc[df[i] == "?",i] = df[i].value_counts().idxmax()
```

Visualisation

```
# import viz
v = visualisation(df)

# gain and loss plot
v.hist_plot("capital-gain")
v.hist_plot("capital-loss")
```





We see most of the values here are mostly zero and have less distribution of values greater than zero. Still we will keep these two columns.

```
# value counts shows no repetition and shows the same
df['education'] = df['education'].astype(str) + ' - ' +
df['education-num'].astype(str)
df['education'].value_counts()
```

```
HS-grad - 9          10501
Some-college - 10     7291
Bachelors - 13        5355
Masters - 14          1723
Assoc-voc - 11        1382
11th - 7              1175
Assoc-acdm - 12       1067
10th - 6               933
7th-8th - 4            646
Prof-school - 15       576
9th - 5                514
12th - 8               433
Doctorate - 16         413
5th-6th - 3            333
1st-4th - 2            168
Preschool - 1           51
Name: education, dtype: int64
```

Since education and education-num gives same information we can drop either of them.

```
# lets drop this column as it's irrelevant for modelling
df = df.drop('education-num', axis = 1)
```

Data Preparation

For preparing the data for **modelling** we will follow the following steps:

1. Define the 'Target' variable and label encode it as 0 and 1 i.e. **'<=50K' : 0, '>50K' : 1.**
2. Seperate out **continous** and **categorical** variables for preparation.
3. We will one hot encode categorical variables and normalize our continous variables.
4. After this we will stack these two features into one large data set.
5. Split the data into train/test variables(80/20).
6. Since we observed that our class is very imbalanced we will use oversampling technique called SMOTE to balance our dataset.

```
# define target variable
y = df['income']
df = df.drop('income', axis = 1)

# label encode output variable
le = LabelEncoder()
y = le.fit_transform(y)

# segregate continous and categorical variables
num_feat = ['age', 'fnlwgt' ,'hours-per-week', 'capital-gain', 
'capital-loss']

X = df[num_feat]
df = df.drop(num_feat, axis = 1)

# one hot encode categorical variables
encoder =  OneHotEncoder()
X_cat = encoder.fit_transform(df).toarray()
cat_feat = encoder.get_feature_names().tolist()
```

```
C:\Users\Saurabh\Anaconda3\lib\site-packages\sklearn\utils\
deprecation.py:87: FutureWarning: 

Function get_feature_names is deprecated; get_feature_names is 
deprecated in 1.0 and will be removed in 1.2. Please use 
get_feature_names_out instead.
```

```python
# all features
cols = num_feat + cat_feat

# normalise the continous variables
scaler = MinMaxScaler(feature_range=(0, 1))
X      = scaler.fit_transform(X)

# stack these values together
X    = np.hstack([X,X_cat])

# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20)

# transform the dataset with class distribution
sm = SMOTE(k_neighbors=2)
X_train,y_train = sm.fit_resample(X_train,y_train)
X_test,y_test = sm.fit_resample(X_test,y_test)

# let's confirm the balanced counts
np.unique(y_train, return_counts = True)
```

```
(array([0, 1]), array([19823, 19823], dtype=int64))
```

Modelling

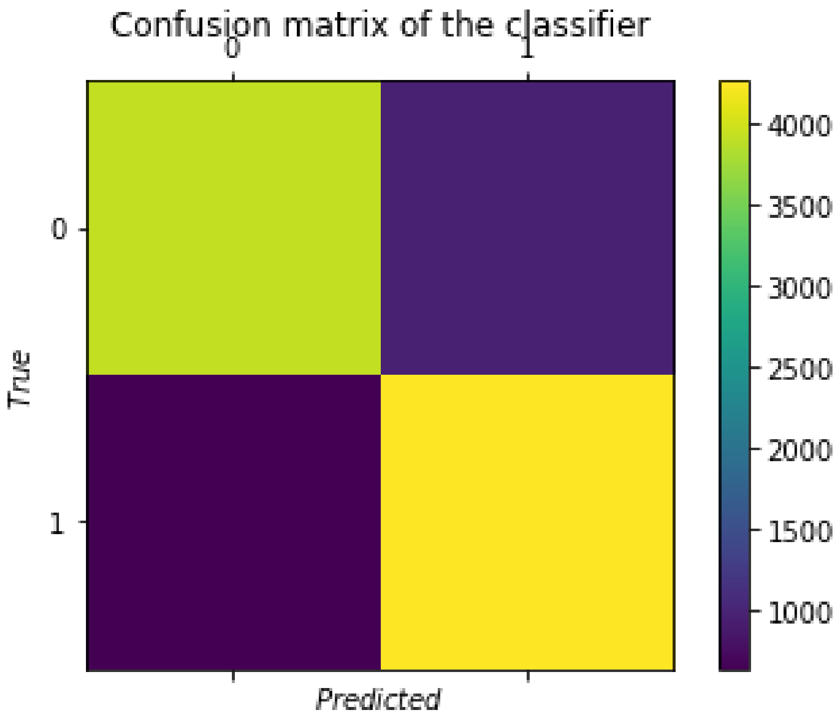
> We have created a class module `classification.py` to run all the machine learning algorithms. The class file contains all the methods to run the algorithms along with grid search CV, XAI and classification metrics to see the performance of model.

The algorithms which we will be using are:

1. **Logistic Regression** : This model is used to get baseline accuracy.
2. **Random Forest** : The purpose of this model is to use the "bagging" approach for improving the accuracy of the model. Using this approach we try to build and see the feature importance of each independent variable.
3. **XG Boost** : The purpose of this model is to use the "boosting" approach to improving the weak learners. We actually try to reduce the prediction error in each step of our iteration.
4. **Fully connected Neural Network** : We run a two layered neural network for classification.

```python
# initialise the class
c = classification( X_train,y_train,X_test,y_test,cols)

# logistic regression
model = c.logistic_regression()
```

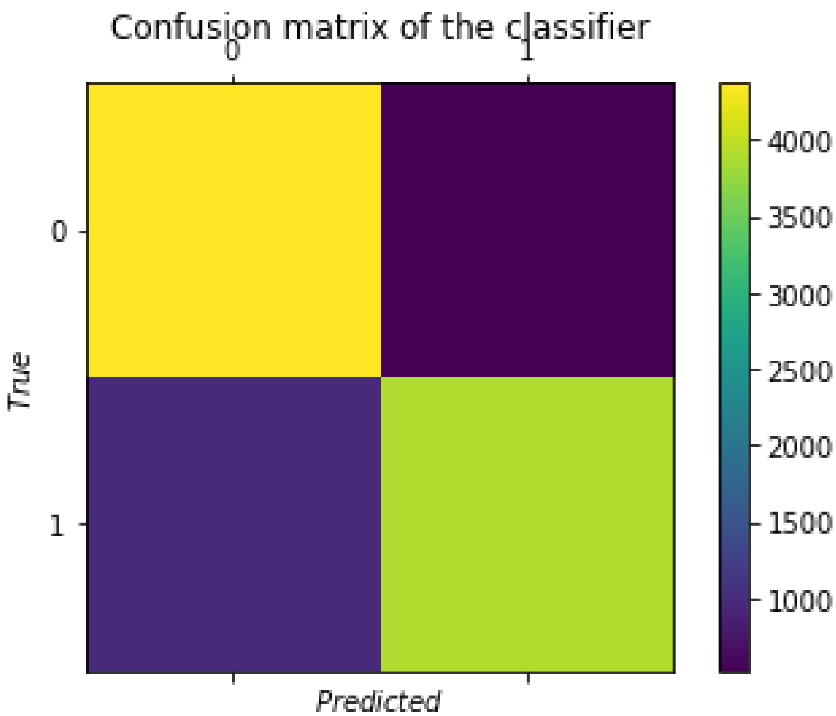
```
Performing modelling for Logistic Regression

C:\Users\Saurabh\Anaconda3\lib\site-packages\sklearn\utils\
validation.py:993: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().


Best parameters = {'C': 1291.5496650148827, 'penalty': 'l2'}

C:\Users\Saurabh\Anaconda3\lib\site-packages\sklearn\utils\
validation.py:993: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().


Test results for test set
Generating the results wait for it....
              precision    recall  f1-score   support

           0       0.86      0.80      0.83      4897
           1       0.81      0.87      0.84      4897

    accuracy                           0.84      9794
   macro avg       0.84      0.84      0.84      9794
weighted avg       0.84      0.84      0.84      9794

The results of your model are:

<IPython.core.display.HTML object>

None
The confusion matrix is :
[[3929  968]
 [ 637 4260]]
```




```
# random forest
model = c.random_forest(feature_importance=True)
```

```
Performing modelling for Random forest

C:\Users\Saurabh\Anaconda3\lib\site-packages\sklearn\model_selection\
_search.py:926: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().


Best parameters = {'max_features': 0.1, 'min_samples_split': 2,
'n_estimators': 75}

C:\Users\Saurabh\Downloads\Dain_Studios_Task\classification.py:131:
DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().


Test results for test set
Generating the results wait for it....
              precision    recall  f1-score   support

           0       0.82      0.89      0.85      4897
           1       0.88      0.80      0.84      4897
```

```
    accuracy                           0.85      9794
   macro avg       0.85      0.85      0.85      9794
weighted avg       0.85      0.85      0.85      9794

The results of your model are:

<IPython.core.display.HTML object>

None
The confusion matrix is :
[[4370  527]
 [ 987 3910]]
```




```
The feature importance is :
                   variable  importance
0                       age    0.163339
1                    fnlwgt    0.118611
2            hours-per-week    0.101381
31  x2_Married-civ-spouse    0.075825
3              capital-gain    0.063331
50               x4_Husband    0.048943
33         x2_Never-married    0.042314
53             x4_Own-child    0.024383
51         x4_Not-in-family    0.024278
22        x1_Bachelors - 13    0.019534
39      x3_Exec-managerial    0.017316
4              capital-loss    0.017213
```

```
24         x1_HS-grad - 9    0.015972
45      x3_Prof-specialty    0.013194
43       x3_Other-service    0.013036
```

```
The feature importance viz for data index 0 is:

<IPython.core.display.HTML object>
```

```
# lets arbitarily see the feature importance of one of the index
c.feature_importance_lime(model, i= 6)
```

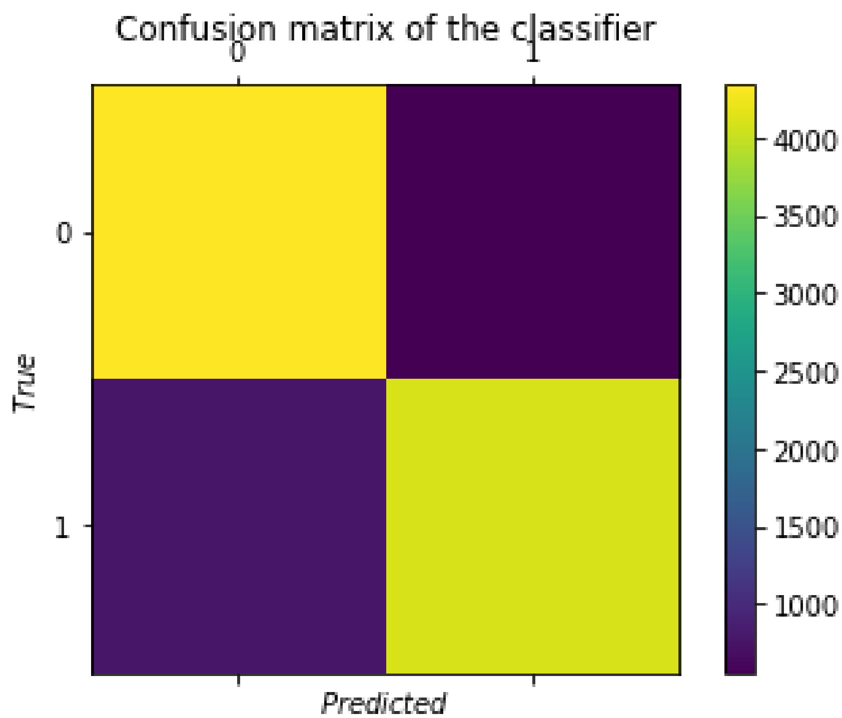
```
The feature importance viz for data index 6 is:

<IPython.core.display.HTML object>
```

The feature importance matrix shows the relevant features for classification. Also we can see categorically which features are important for model output.

```
# XG Boost
model = c.XG_Boost(feature_importance=True)
```

```
Performing modelling for XG Boost Classifier

C:\Users\Saurabh\Anaconda3\lib\site-packages\sklearn\preprocessing\
_label.py:98: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().

C:\Users\Saurabh\Anaconda3\lib\site-packages\sklearn\preprocessing\
_label.py:133: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().


Best parameters = {'learning_rate': 0.01, 'max_depth': 75,
'n_estimators': 200}

C:\Users\Saurabh\Anaconda3\lib\site-packages\sklearn\preprocessing\
_label.py:98: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().

C:\Users\Saurabh\Anaconda3\lib\site-packages\sklearn\preprocessing\
_label.py:133: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
```

```
Test results for test set
Generating the results wait for it....
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      4897
           1       0.88      0.84      0.86      4897

    accuracy                           0.86      9794
   macro avg       0.86      0.86      0.86      9794
weighted avg       0.86      0.86      0.86      9794

The results of your model are:

<IPython.core.display.HTML object>

None
The confusion matrix is :
[[4338  559]
 [ 794 4103]]
```




```
The feature importance is :
                 variable  importance
31  x2_Married-civ-spouse    0.908614
3            capital-gain    0.006242
19              x1_9th - 5    0.005416
40     x3_Farming-fishing    0.003922
17         x1_5th-6th - 3    0.003052
18         x1_7th-8th - 4    0.003030
```

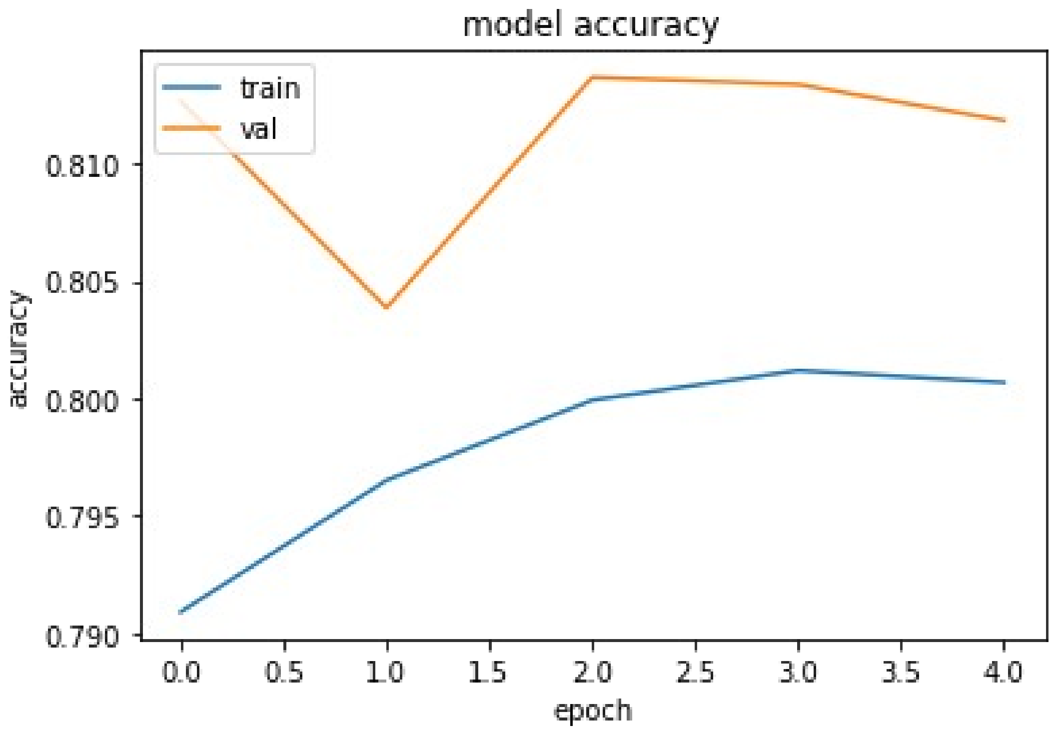
```
14            x1_11th - 7    0.002838
23      x1_Doctorate - 16    0.002408
94           x7_Portugal    0.002172
63           x7_Cambodia    0.002135
22      x1_Bachelors - 13    0.002052
64             x7_Canada    0.002014
4           capital-loss    0.001838
13           x1_10th - 6    0.001820
43      x3_Other-service    0.001766

The feature importance viz for data index 0 is:

<IPython.core.display.HTML object>
```

Boosting shows different features responsible for modelling the data

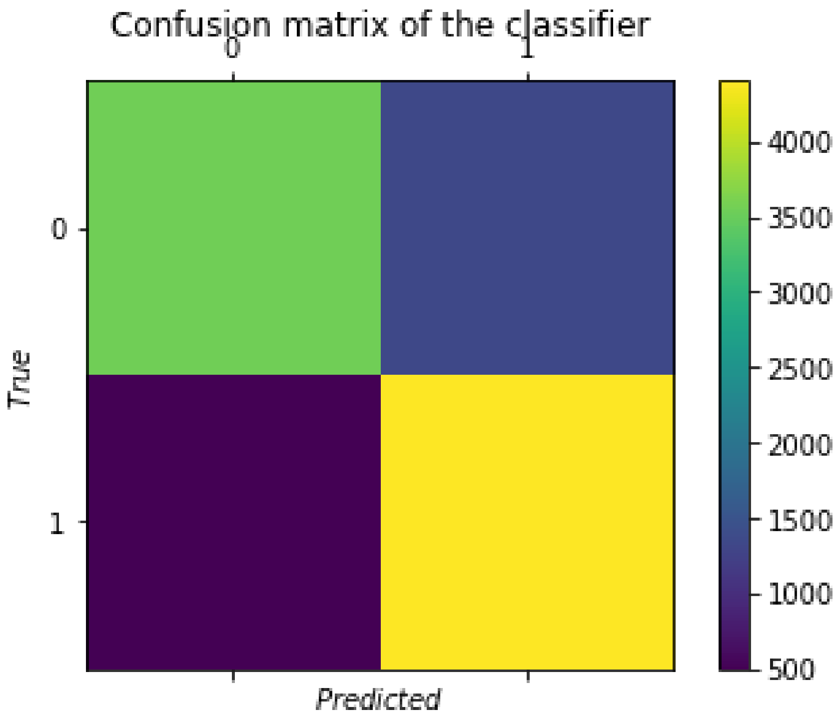
```
# let's run the Neural Network now
model = c.Neural_Network()
```

```
Performing modelling for neural network
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               26880
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 60,034
Trainable params: 60,034
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/5
1239/1239 [==============================] - 9s 8ms/step - loss:
0.4774 - accuracy: 0.7909 - val_loss: 0.4680 - val_accuracy: 0.8126
Epoch 2/5
1239/1239 [==============================] - 10s 8ms/step - loss:
0.4695 - accuracy: 0.7965 - val_loss: 0.4413 - val_accuracy: 0.8039
Epoch 3/5
1239/1239 [==============================] - 10s 8ms/step - loss:
0.4644 - accuracy: 0.7999 - val_loss: 0.4332 - val_accuracy: 0.8137
Epoch 4/5
1239/1239 [==============================] - 4s 3ms/step - loss:
0.4606 - accuracy: 0.8012 - val_loss: 0.4413 - val_accuracy: 0.8134
```

```
Epoch 5/5
1239/1239 [==============================] - 10s 8ms/step - loss:
0.4586 - accuracy: 0.8007 - val_loss: 0.4302 - val_accuracy: 0.8118
```




```
Test results for test set
Generating the results wait for it....
              precision    recall  f1-score   support

           0       0.88      0.72      0.79      4897
           1       0.77      0.90      0.83      4897

    accuracy                           0.81      9794
   macro avg       0.82      0.81      0.81      9794
weighted avg       0.82      0.81      0.81      9794

The results of your model are:

<IPython.core.display.HTML object>

None
The confusion matrix is :
[[3550 1347]
 [ 496 4401]]
```


Confusion matrix of the classifier
0
1
0
1
True
Predicted
4000
3500
3000
2500
2000
1500
1000
500