```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.svm import SVC
from sklearn.metrics import roc_auc_score


!gdown --id 1WpfQNlfCPvvQVWA0IhiVSKmz4u8lZMdA
!gdown --id 1keyc76QM4qaqBPqpxXZucCKeAF79DZ-D
df = pd.read_csv('train.csv')
df.head()
```

```
/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id` was deprecated ir
  category=FutureWarning,
Downloading...
From: https://drive.google.com/uc?id=1WpfONlfCPvvOVWA0IhiVSKmz4u8lZMdA
```

```
df.isna().sum()
```

```
age                  0
workclass         2498
fnlwgt               0
education            0
educational-num      0
marital-status       0
occupation        2506
relationship         0
race                 0
gender               0
capital-gain         0
capital-loss         0
hours-per-week       0
native-country     763
income_>50K          0
dtype: int64
```

2	31	Private	174201	Bachelors	13	civ-		Husband	White	Male	

```
df = df.dropna()
```

						Married-					

```
df.isna().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
educational-num   0
marital-status    0
occupation        0
relationship      0
race              0
gender            0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income_>50K       0
dtype: int64
```

```
df.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'educational-num',
       'marital-status', 'occupation', 'relationship', 'race', 'gender',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'income_>50K'],
      dtype='object')
```

```
categorical_features = list(df.select_dtypes(include=['object']).columns)
```

```
categorical_features
```

```
['workclass',
 'education',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'gender',
 'native-country']
```

```
def encoding(df):
    label_encoder_feat = {}
    for i, feature in enumerate(categorical_features):
        label_encoder_feat[feature] = LabelEncoder()
        df[feature] = label_encoder_feat[feature].fit_transform(df[feature])
    return df
```

```
newdf = encoding(df)
```

```
newdf.head()
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	c

```
from sklearn.model_selection import train_test_split
```

1	17	2	244602	2	8	4	7	3	4	1	

```
features = newdf.drop('income_>50K',axis = 1)
target = newdf['income_>50K']
```

3	58	5	110199	5	4	2	13	0	4	1	

```
features.head()
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	c
0	67	2	366425	10	16	0	3	1	4	1	
1	17	2	244602	2	8	4	7	3	4	1	
2	31	2	174201	9	13	2	3	0	4	1	
3	58	5	110199	5	4	2	13	0	4	1	

```
target.head()
```

```
0    1
1    0
2    1
3    0
4    0
Name: income_>50K, dtype: int64
```

```
X_train,X_test,Y_train,Y_test  = train_test_split(features,target,shuffle=True)
```

```
from xgboost import XGBClassifier
from xgboost import plot_importance

# Training the model
model = XGBClassifier()
model_importance = model.fit(X_train, Y_train)
```

```
# Plotting the Feature importance bar graph
plt.rcParams['figure.figsize'] = [14,12]
sns.set(style = 'darkgrid')
plot_importance(model_importance);
```

```
# Training the model_1
logistic = LogisticRegression(C = 0.5, max_iter = 500)
model_1 = logistic.fit(X_train, Y_train)

# Predictions
pred_1 = model_1.predict(X_test)

print ("The accuracy of model 1 : ",accuracy_score(Y_test, pred_1))
print ("The f1 score of model 1 : ", f1_score(Y_test, pred_1, average = 'binary'))
```

```
     The accuracy of model 1 :  0.7917894323315655
     The f1 score of model 1 :  0.3928980526918671
```

```
# Training the model_2
R_forest = RandomForestClassifier(n_estimators = 200)
model_2 = R_forest.fit(X_train, Y_train)

# Predictions
pred_2 = model_2.predict(X_test)

print ("The accuracy of model 2 : ",accuracy_score(Y_test, pred_2))
print ("The f1 score of model 2 : ", f1_score(Y_test, pred_2, average = 'binary'))
```

```
     The accuracy of model 2 :  0.8516008642702809
     The f1 score of model 2 :  0.6727312107429066
```

```
# Training the model 3
boosted_gd = XGBClassifier(learning_rate = 0.35, n_estimator = 500)
model_3 = boosted_gd.fit(X_train, Y_train)

# Predictions
pred_3 = model_3.predict(X_test)

print ("The accuracy of model 3 : ",accuracy_score(Y_test, pred_3))
print ("The f1 score of model 3 : ", f1_score(Y_test, pred_3, average = 'binary'))
```

```
     The accuracy of model 3 :  0.8668238067177372
     The f1 score of model 3 :  0.7035417577612594
```

```
# Training the model 4
NB = BernoulliNB(alpha = 0.3)
model_4 = NB.fit(X_train, Y_train)

# Predictions
pred_4 = model_4.predict(X_test)

print ("The accuracy of model 4 : ",accuracy_score(Y_test, pred_4))
print ("The f1 score of model 4 : ", f1_score(Y_test, pred_4, average = 'binary'))
```

```
The accuracy of model 4 :  0.7298173246906305
The f1 score of model 4 :  0.5702234025933448
```

```
# Training the model 5
svc = SVC(kernel = 'rbf', max_iter = 1000, probability = True)
model_5 = svc.fit(X_train, Y_train)

# Predictions
pred_5 = model_5.predict(X_test)

print ("The accuracy of model 5 : ",accuracy_score(Y_test, pred_5))
print ("The f1 score of model 5 : ", f1_score(Y_test, pred_5, average = 'binary'))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:289: ConvergenceWarning: Solver terminated early (max_iter=1000).  Co
  ConvergenceWarning,
The accuracy of model 5 :  0.5247495580436063
The f1 score of model 5 :  0.3193135462090308
```

```
list_pred = [pred_1, pred_2, pred_3, pred_4, pred_5]
model_names = ["Logistic Regression", "Random Forest Classifier", "Boosted Gradient Descent", "Bernoulli NB", "SVC"]

for i, predictions in enumerate(list_pred) :
    print ("Classification Report of ", model_names[i])
    print ()
    print (classification_report(Y_test, predictions, target_names = ["<=50K", ">50K"]))
```

```
Classification Report of  Logistic Regression

              precision    recall  f1-score   support

       <=50K       0.80      0.96      0.87      7657
        >50K       0.71      0.27      0.39      2525
```

```
    accuracy                           0.79     10182
   macro avg       0.75      0.62      0.63     10182
weighted avg       0.78      0.79      0.75     10182

Classification Report of  Random Forest Classifier

              precision    recall  f1-score   support

       <=50K       0.88      0.93      0.90      7657
        >50K       0.74      0.62      0.67      2525

    accuracy                           0.85     10182
   macro avg       0.81      0.77      0.79     10182
weighted avg       0.85      0.85      0.85     10182

Classification Report of  Boosted Gradient Descent

              precision    recall  f1-score   support

       <=50K       0.89      0.94      0.91      7657
        >50K       0.79      0.64      0.70      2525

    accuracy                           0.87     10182
   macro avg       0.84      0.79      0.81     10182
weighted avg       0.86      0.87      0.86     10182

Classification Report of  Bernoulli NB

              precision    recall  f1-score   support

       <=50K       0.89      0.73      0.80      7657
        >50K       0.47      0.72      0.57      2525

    accuracy                           0.73     10182
   macro avg       0.68      0.73      0.69     10182
weighted avg       0.79      0.73      0.75     10182

Classification Report of  SVC

              precision    recall  f1-score   support

       <=50K       0.75      0.55      0.63      7657
        >50K       0.25      0.45      0.32      2525

    accuracy                           0.52     10182
   macro avg       0.50      0.50      0.48     10182
```

```
      weighted avg       0.63      0.52      0.56     10182
```

```
test_data =  pd.read_csv('test.csv')

test_data = encoding(test_data)

res = model_3.predict(test_data)

test_data['target'] = res

test_data['outcome'] = res

test_data.index.name = 'id'

res = test_data['outcome']
```

Colab paid products - Cancel contracts here