Lab Assignment 5

Chaudhary Hamdan

1905387

Date: 02-03-2022

Table filled in percentage values

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Recall | Precision | F-Score |
| LR | 76.54784240150094 | 95.23130222131648 | 78.68328001347028 | 86.17001659598007 |
| knn | 79.14321450906817 | 87.46688404320359 | 85.65156655358211 | 86.54970760233918 |
| DT | 78.08005003126954 | 84.53230079478297 | 86.57900229597162 | 85.54341101257991 |
| NB | 75.84427767354597 | 78.05176278785409 | 89.11121451838065 | 83.21564367191743 |
| RF | 82.44215134459037 | 90.60525779498676 | 87.0399373531715 | 88.78681977034448 |

For the dataset ‘income.csv’: find the following performance metrics:

1. Accuracy
2. Precision
3. F-score
4. Recall/Sensitivity

Confusion matrix

**Classification Report**

**# coding: utf-8**

**# In[207]:**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn import preprocessing**

**import seaborn as sns**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.naive\_bayes import GaussianNB**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.metrics import confusion\_matrix**

**from sklearn.metrics import accuracy\_score**

**from sklearn.metrics import recall\_score**

**from sklearn.metrics import precision\_score**

**from sklearn.metrics import f1\_score**

**# In[208]:**

**import warnings**

**warnings.filterwarnings('ignore')**

**# In[209]:**

**df = pd.read\_excel('income.xlsx')**

**# In[210]:**

**df.head()**

**# In[211]:**

**df.drop(columns=['capitalgain', 'capitalloss'], inplace=True)**

**# In[212]:**

**df.head()**

**# In[213]:**

**cols = ['JobType', 'EdType', 'maritalstatus', 'occupation', 'relationship', 'race',**

**'gender', 'nativecountry', 'SalStat']**

**for col in cols:**

**le = preprocessing.LabelEncoder()**

**df[col] = le.fit\_transform(df[col])**

**# In[214]:**

**df.head()**

**# In[215]:**

**df.SalStat.value\_counts()**

**# 1 : Less than or equal to 50k, 0 means less than 50k**

**# In[216]:**

**df.head()**

**# In[217]:**

**corr = df.corr()**

**corr.style.background\_gradient(cmap='coolwarm')**

**# In[218]:**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(df.drop(columns = ['SalStat']), df['SalStat'], test\_size = 0.2)**

**x\_train.shape, y\_train.shape, x\_test.shape, y\_test.shape**

**# In[219]:**

**algos = []**

**accuracy = []**

**recall = []**

**precision = []**

**f1Score = []**

**# In[220]:**

**algo = "Logistic Regression"**

**model = LogisticRegression()**

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

**y\_pred = model.predict(x\_test)**

**print(algo)**

**print(confusion\_matrix(y\_test, y\_pred), '\n\n')**

**acc = accuracy\_score(y\_test, y\_pred) \* 100**

**print('Accuracy:', acc)**

**rec = recall\_score(y\_test, y\_pred) \* 100**

**print('Recall:', rec)**

**pre = precision\_score(y\_test, y\_pred) \* 100**

**print('Precision:', pre)**

**f1s = f1\_score(y\_test, y\_pred) \* 100**

**print('F score:', f1s)**

**algos.append(algo)**

**accuracy.append(acc)**

**recall.append(rec)**

**precision.append(pre)**

**f1Score.append(f1s)**

**# In[221]:**

**algo = "K Nearest Neighbour"**

**model = KNeighborsClassifier()**

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

**y\_pred = model.predict(x\_test)**

**print(algo)**

**print(confusion\_matrix(y\_test, y\_pred), '\n\n')**

**acc = accuracy\_score(y\_test, y\_pred) \* 100**

**print('Accuracy:', acc)**

**rec = recall\_score(y\_test, y\_pred) \* 100**

**print('Recall:', rec)**

**pre = precision\_score(y\_test, y\_pred) \* 100**

**print('Precision:', pre)**

**f1s = f1\_score(y\_test, y\_pred) \* 100**

**print('F score:', f1s)**

**algos.append(algo)**

**accuracy.append(acc)**

**recall.append(rec)**

**precision.append(pre)**

**f1Score.append(f1s)**

**# In[222]:**

**algo = "Decision Tree"**

**model = DecisionTreeClassifier()**

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

**y\_pred = model.predict(x\_test)**

**print(algo)**

**print(confusion\_matrix(y\_test, y\_pred), '\n\n')**

**acc = accuracy\_score(y\_test, y\_pred) \* 100**

**print('Accuracy:', acc)**

**rec = recall\_score(y\_test, y\_pred) \* 100**

**print('Recall:', rec)**

**pre = precision\_score(y\_test, y\_pred) \* 100**

**print('Precision:', pre)**

**f1s = f1\_score(y\_test, y\_pred) \* 100**

**print('F score:', f1s)**

**algos.append(algo)**

**accuracy.append(acc)**

**recall.append(rec)**

**precision.append(pre)**

**f1Score.append(f1s)**

**# In[223]:**

**algo = "Naive Bayes"**

**model = GaussianNB()**

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

**y\_pred = model.predict(x\_test)**

**print(algo)**

**print(confusion\_matrix(y\_test, y\_pred), '\n\n')**

**acc = accuracy\_score(y\_test, y\_pred) \* 100**

**print('Accuracy:', acc)**

**rec = recall\_score(y\_test, y\_pred) \* 100**

**print('Recall:', rec)**

**pre = precision\_score(y\_test, y\_pred) \* 100**

**print('Precision:', pre)**

**f1s = f1\_score(y\_test, y\_pred) \* 100**

**print('F score:', f1s)**

**algos.append(algo)**

**accuracy.append(acc)**

**recall.append(rec)**

**precision.append(pre)**

**f1Score.append(f1s)**

**# In[224]:**

**algo = "Random Forest"**

**model = RandomForestClassifier()**

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

**y\_pred = model.predict(x\_test)**

**print(algo)**

**print(confusion\_matrix(y\_test, y\_pred), '\n\n')**

**acc = accuracy\_score(y\_test, y\_pred) \* 100**

**print('Accuracy:', acc)**

**rec = recall\_score(y\_test, y\_pred) \* 100**

**print('Recall:', rec)**

**pre = precision\_score(y\_test, y\_pred) \* 100**

**print('Precision:', pre)**

**f1s = f1\_score(y\_test, y\_pred) \* 100**

**print('F score:', f1s)**

**algos.append(algo)**

**accuracy.append(acc)**

**recall.append(rec)**

**precision.append(pre)**

**f1Score.append(f1s)**

**# In[225]:**

**for i in range(5):**

**print(algos[i], ': ', accuracy[i],', ', recall[i],', ', precision[i],', ', f1Score[i])**

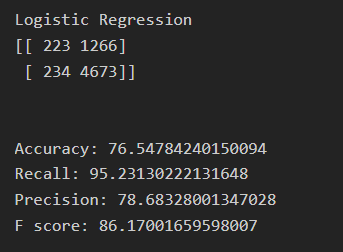
**# In[228]:**

**plt.bar(algos, accuracy)**

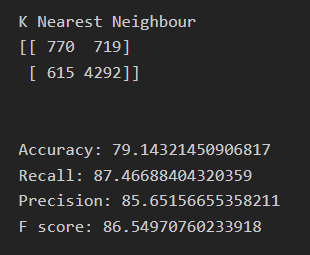
**plt.show()**

Using following machine learning algorithm

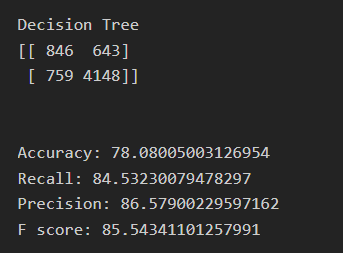
1. Logistic Regression



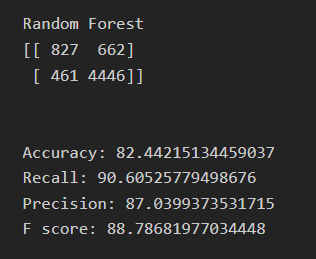
1. k-NN



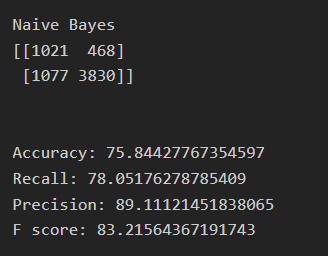
1. Decision Tree



1. Random Forest



1. Naive Bayes



2.Construct a table and compare all the algorithms’ result.Plot a bar for accuracy of each algorithm.

