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:

i) Accuracy
ii) Precision
iii) F-score
iv) 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

```
Logistic Regression
[[ 223 1266]
 [ 234 4673]]


Accuracy: 76.54784240150094
Recall: 95.23130222131648
Precision: 78.68328001347028
F score: 86.17001659598007
```

2. k-NN

```
K Nearest Neighbour
[[ 770  719]
 [ 615 4292]]


Accuracy: 79.14321450906817
Recall: 87.46688404320359
Precision: 85.65156655358211
F score: 86.54970760233918
```

3. Decision Tree

```
Decision Tree
[[ 846  643]
 [ 759 4148]]


Accuracy: 78.08005003126954
Recall: 84.53230079478297
Precision: 86.57900229597162
F score: 85.54341101257991
```

4. Random Forest

```
Random Forest
[[ 827  662]
 [ 461 4446]]


Accuracy: 82.44215134459037
Recall: 90.60525779498676
Precision: 87.0399373531715
F score: 88.78681977034448
```

5. Naive Bayes

```
Naive Bayes
[[1021  468]
 [1077 3830]]


Accuracy: 75.84427767354597
Recall: 78.05176278785409
Precision: 89.11121451838065
F score: 83.21564367191743
```

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

```
Logistic Regression :      76.54784240150094 ,   95.23130222131648 ,   78.68328001347028 ,   86.17001659598007
K Nearest Neighbour :      79.14321450906817 ,   87.46688404320359 ,   85.65156655358211 ,   86.54970760233918
Decision Tree :      78.08005003126954 ,   84.53230079478297 ,   86.57900229597162 ,   85.54341101257991
Naive Bayes :      75.84427767354597 ,   78.05176278785409 ,   89.11121451838065 ,   83.21564367191743
Random Forest :      82.44215134459037 ,   90.60525779498676 ,   87.0399373531715 ,   88.78681977034448
```