POORNIMA

Experiment:- 5

Objective:- Naives Bayes Classification implementation on Google Colab with deployment using Flask.

⇒

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')

from google.colab import files
Uploaded = files.upload()
Uploaded
dataset = pd.read_excel('/content/DATASET-education.xlsx')
print(dataset.shape)
dataset.info()
print(dataset)

X = dataset.iloc[:, 0:8].values
y = dataset.iloc[:, 9].values
print(X)
print(y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
                                                    random_state=0)
```

POORNIMA

```
print (X_train)
print (X_test)
print (y_test)
print (y_train)
```

→
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

print(X_train)
print(X_test)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
print(y_pred)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion matrix:")
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy: %.2f" % (accuracy_score(y_test, y_pred)*100))
```

POORNIMA

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
precision = precision_score(y_test, y_pred, average='binary')
print('Precision : %.3f' %(Precision*100))

recall = recall_score(y_test, y_pred, average='binary')
print('Recall : %.3f' %(recall*100))

score = f1_score(y_test, y_pred, average='binary')
print('F-measure : %.3f' %(score*100))

import pickle
print("[INFO] saving model...")
saved_model = pickle.dump(classifier, open('/content/drive/MyDrive/
        DATASET-Education.pkl', 'wb'))
model = pickle.load(open('/content/drive/MyDrive/DATASET-Education.pkl','rb'))
model.predict(X_test)

import joblib
filename = '/content/drive/MyDrive/DATASET.Education.xlsx'
joblib.dump(classifier, filename)

randomforest_model = joblib.load(filename)
result = randomforest_model.score(X_test, y_test)
print(result)
```

```
marks in seconds = 85.30 #@param {type: "numbr"}
marks in high seconds = 72.00 #@param {type: "numbr"}
aggregate marks = 74.90 #@param {type: "numbr"}
VII sem mark = 67 #@ param {type: "numbr"}
VI sem mark = 67 #@ param {type: "numbr"}
V sem mark = 67 #@ param{type: "numbr"}
final performance = 56.67 #@param{type: "numbr"}
medium = 1 #@param {type: "numbr"}
naive_Bayes_model = joblib.load(filename)
output = naive_Bayes_model.predict([[marks in seconds, marks in highseconds, aggregate,
                                     VII sem mk, VI sem mk, V sem mk, final perfo, medium]])
print("Placed =", output)
  if output == [1]:
      print('Student will be placed')
  else:
      print('Student will not be placed')

%mkdir templates -p

⇒ %%writefile templates/index.html
<!DOCTYPE html>
<html>
<!-- From https://codepen.io/bryfyfes/pen/EXdtg -->
<head>
<title>Machine Learning Lab Experiment Deployment</title>
<meta charset="utf-8">
```

```
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel=
    'Stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='Stylesheet'
    type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel=
    stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
    rel='Stylesheet' type='text/css'>
~~<link href=~~
<style><!DOCTYPE html>
  h1 {text-align: center}
  h2 {text-align: center}
  h3 {text-align: center}
  p {text-align: center}
  div {text-align: center}
</style>
</head>
<body>
<div class="" style="background-color: blue;">
<div class="clearfix">
<div class="col-md-12">
<center><p style="font-size: 40px; color: white; margin-top: 10px;"
    Poornima Institute of Engineering & Technology</p></center>
<center><p style="font-size: 30px; color: white; margin-top: 10px;">
    Department of Computer Engineering</p></center>
<center><p style="font-size: 20px; color: white; margin-top: 10px;">
    Machine Learning Lab Manual</p></center>
```

```
</div>
</div>
</div>
<div class="login">
<h2> Naive Bayes classifier for Prediction of Placement</h2>
<h4> Developed by Aryan Vijay</h4>
<!-- Main input for Receiving query to our ML -->
<form action="{{ url_for('predict')}}" method="post">
<div class="form-floating mb-3">
<input type="number" class="form-control" id="floatingInput"
name="percentage10" placeholder="Percentage" min="1" max="10"
required="required" />
<label for="floatingInput">Enter 10th class Percentage</label>
<label for="floatingInput">Enter 12th class Percentage</label>
<label for="floatingInput">Enter BTECH Aggregate</label>
<label for="floatingInput">Enter VII sem Placement Performance Marks</label>
<label for="floatingInput">Enter VI sem Placement Performance Marks</label>
<label for="floatingInput">Enter V sem Placement Performance Marks</label>
<label for="floatingInput">Enter Final Performance Marks</label>
</div>
<br>
<label for="medium">Choose a Medium</label>
<select name="medium" id="medium">
<option value="1">Hindi</option>
<option value="2">English</option>
</select>
```

```
<button type="submit" class="btn btn-primary btn-block btn-large">
        Predict Student Placement </button>
<div class="" style="background color blue;">
<div class="clearfix">
</div>
</div>
</div>
</body>
</html>
```

→ !pip install flask-ngrok
!pip install pyngrok.

!pyngrok authtoken 27hDyMSLHX1ikBLnd Tmy CDD gB2G-6hzVGx...

→
```
import numpy as np
from flask import Flask, request, jsonify, render_template
from flask_ngrok import run_with_ngrok.
import pickle
import pandas as pd
dataset = pd.read_excel('/content/DATASET.education.xlsx')
X = dataset.iloc[:, 0:8].values
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
app = Flask(__name__)
model = pickle.load(open('/content/drive/MyDrive/DATASET education.pkl',
                          'rb'))
```

```
run_with_ngrok(app)
@app.route('/')
def home():
    return render_template("index.html")
@app.route('/predict', method=['GET'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
percentage10 = int(request.args.get('Percentage10'))
percentage12 = int(request.args.get('Percentage12'))
BTECHpercentage = int(request.args.get('BTECHpercentage'))
marks7sem = int(request.args.get('marks7sem'))
marks6sem = int(request.args.get('marks6sem'))
marks5sem = int(request.args.get("marks5sem"))
finalperformance = int(request.args.get('finalperformance'))
medium = int(request.args.get('medium'))
prediction = model.predict([[percentage10, percentage12, BTECHpercentage, marks7sem,
                marks6sem, marks5sem, finalperformance, medium]])
print(prediction)
if prediction == [1]:
    output = 'placed'
else:
    output = 'notplaced'
return render_template('index.html', prediction_text = 'Made by prediction
                Student will be : {}'.format(output))
app.run()
```