Machine Learning
LAB MANUAL

Subject Code: B23-CSE-313

Class: 3rd Year (5th Semester)

Bachelor of Technology

in

Computer Engineering

Prepared By: Mrs. Shubh Chawala and Mr. Ashish Kharb

Assistant Professor, Department of Computer Engineering

DEPARTMENT OF COMPUTER ENGINEERING

STATE INSTITUTE OF ENGINEERING & TECHNOLOGY, NILOKHERI

(AFFILIATED TO KURUKSHETRA UNIVERSITY, KURUKSHETRA)

1. Course Purpose and Objectives

The purpose of this laboratory course is to develop the ability to implement the supervised and the unsupervised machine learning algorithms. The objective is to provide students with hands-on experience in data preprocessing, predictive modeling, classification, clustering, and dimensionality reduction to enhance practical problem-solving skills.

2. Course Outcomes (CO)

- **CO1:** To develop proficiency in fundamental operations for machine learning.
- **CO2:** To implement supervised learning algorithms for predictive modeling and classification.
- **CO3:** To apply unsupervised learning techniques for clustering and dimensionality reduction.
- **CO4:** To enhance problem-solving skills through practical machine learning implementations.

3. Lab Rules, Safety Instructions & Evaluation Scheme

Lab Rules

1. Students must carry this manual and their observation notebook to every lab session.
2. Ensure all Python scripts, Jupyter notebooks, and datasets are saved in the designated user workspace and backed up regularly.
3. Code should be kept simple and beginner-friendly; avoid complex exception handling to ensure a clear understanding of the core machine learning algorithms.
4. Maintain silence and professional decorum.

Hardware & Software Requirements

- **Hardware:** PC with Intel i3/i5 or higher, 8GB RAM (recommended for data processing), 50GB HDD/SSD free space.
- **Software:**
    - **Operating System:** Windows 10/11, Ubuntu/Linux, or macOS.
    - **Compilers/Tools:** Python 3.x Interpreter, pip package manager.
    - **Required Libraries:** NumPy, Pandas, Scikit-learn, Matplotlib.
    - **Editors/IDEs:** Jupyter Notebook, Google Colab, Visual Studio Code (VS Code), or PyCharm.

INDEX

S NO.	NAME OF THE EXPERIMENT	SIGNATURE
1.	Implement Python Program to demonstrate matrices addition, subtraction and multiplication.	
2.	Implement the Linear Regression algorithm using Python.	
3.	Implement the Logistic Regression algorithm using Python.	
4.	Implement the K-nearest Neighbour algorithm.	
5.	Implement the Decision Tree algorithm.	
6.	Implement the Random Forest algorithm.	
7.	Implement the Naive Bayesian Classification algorithm.	
8.	Implement the Support Vector Machine algorithm.	
9.	Implement the Perceptron Neural Learning Network.	
10.	Implement the Backpropagation Learning using Python.	

PRACTICAL-1

AIM-Implement Python Program to demonstrate matrices addition, subtraction and multiplication**.**

TOOLS REQUIRED- VISUAL STUDIO CODE

SOURCE CODE:

```
def matrix_addition(matrix1, matrix2):
    return [[matrix1[i][j] + matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]


def matrix_subtraction(matrix1, matrix2):
    return [[matrix1[i][j] - matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]


def matrix_multiplication(matrix1, matrix2):
    result = [[sum(a * b for a, b in zip(matrix1_row, matrix2_col)) for matrix2_col in zip(*matrix2)] for matrix1_row in matrix1]
    return result


def print_matrix(matrix):
    for row in matrix:
        print(row)


def main():
    matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
    matrix2 = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]

    print("Matrix 1: ")
    print_matrix(matrix1)
    print("\n")

    print("Matrix 2: ")
```

```
    print_matrix(matrix2)
    print("\n")

    print("Addition of Matrix 1  and Matrix 2:")
    print_matrix(matrix_addition(matrix1, matrix2))
    print("\n")

    print("Subtraction of Matrix 1  and Matrix 2:")
    print_matrix(matrix_subtraction(matrix1, matrix2))
    print("\n")

    print("Multiplication of Matrix 1  and Matrix 2:")
    print_matrix(matrix_multiplication(matrix1, matrix2))

if __name__ == "__main__":
    main()
```

OUTPUT-

```
Matrix 1:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]

Matrix 2:
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]

Addition of Matrix 1  and Matrix 2:
[10, 10, 10]
[10, 10, 10]
[10, 10, 10]
```

```
Matrix 1:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]

Matrix 2:
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]

Subtraction of Matrix 1  and Matrix 2:
[-8, -6, -4]
[-2, 0, 2]
[4, 6, 8]
```

```
Matrix 1:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]

Matrix 2:
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]

Multiplication of Matrix 1  and Matrix 2:
[30, 24, 18]
[84, 69, 54]
[138, 114, 90]
```

PROGRAM-2

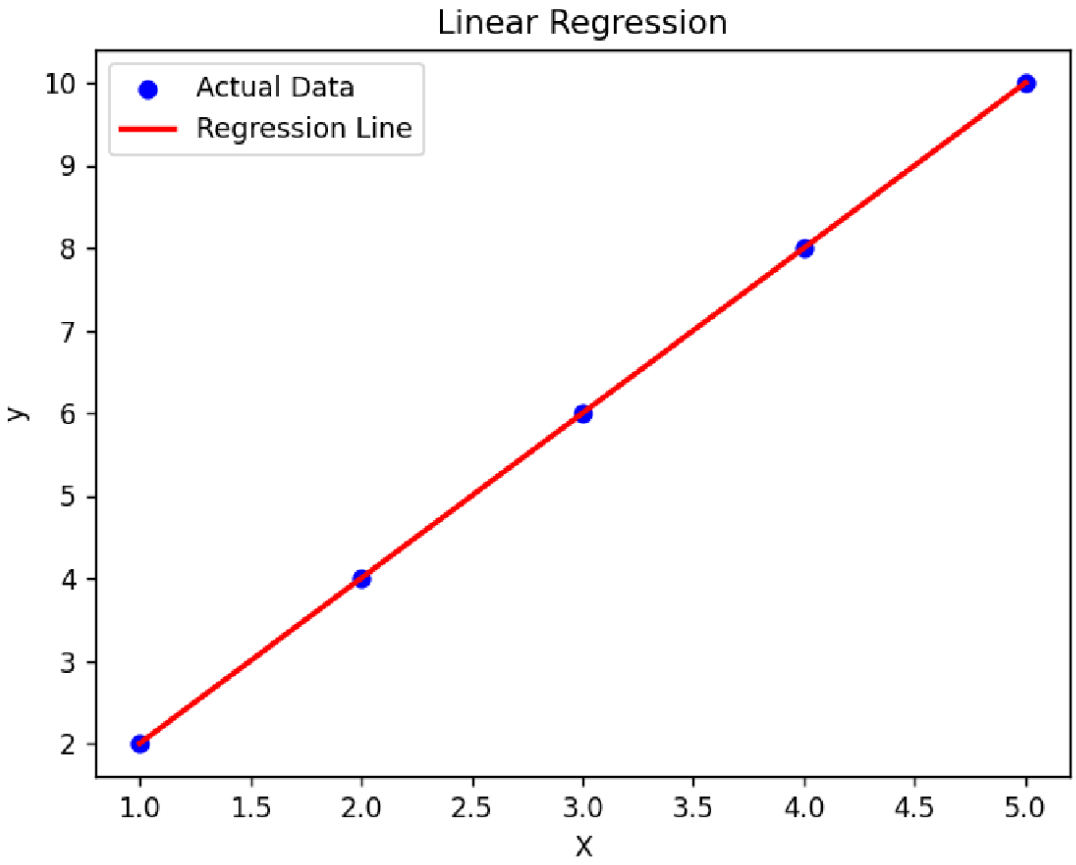
AIM- Implement the Linear Regression algorithm using Python.

TOOL USED- Visual studio code

SOURCE CODE-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

data = {'X': [1, 2, 3, 4, 5], 'y': [2, 4, 6, 8, 10]}
df = pd.DataFrame(data)
X = df[['X']]
y = df['y']

model = LinearRegression()
model.fit(X, y)

predictions = model.predict(X)
df['Predictions'] = predictions

plt.scatter(X, y, color='blue',label='Actual Data')
plt.plot(X, predictions, color='red', linewidth=2, label='Regression Line')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.title('Linear Regression')
plt.show()

print("Predictions:", predictions)
```

OUTPUT-


Linear Regression
Actual Data
Regression Line
y
X
10
9
8
7
6
5
4
3
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

PROGRAM-3

AIM- Implement the Logistic Regression algorithm using Python.

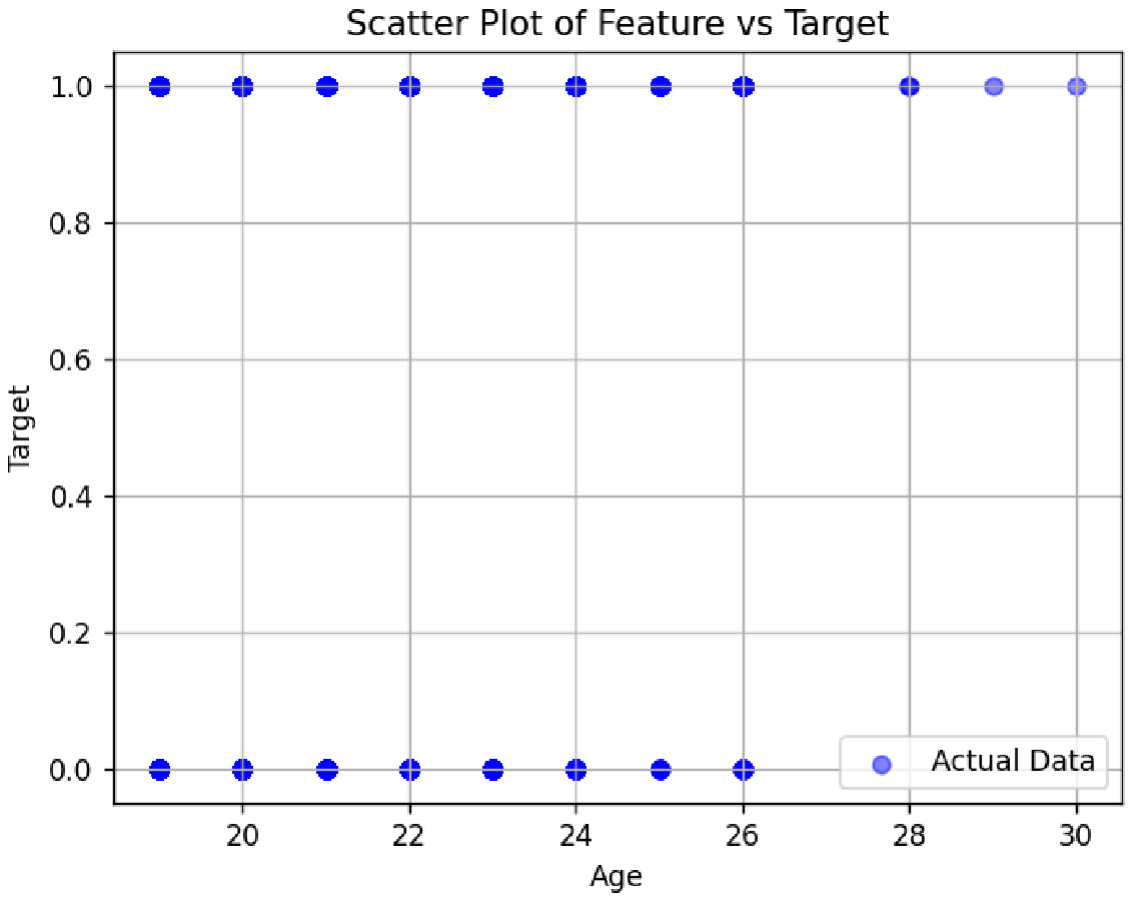
TOOL USED- VISUAL STUDIO CODE

SOURCE CODE –

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:\\Users\\km367\\OneDrive\\Pictures\\Documents\\collegePlace.csv")
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X = pd.get_dummies(X)
if y.dtype == 'object':
    y = pd.factorize(y)[0]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2529)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
cr = classification_report(y_test, y_pred)
print("Classification Report:\n", cr)
feature = X.columns[0]
plt.scatter(X[feature], y, color='blue', label='Actual Data', alpha=0.5)
plt.xlabel(feature)
plt.ylabel('Target')
plt.title('Scatter Plot of Feature vs Target')
plt.legend()
plt.grid(True) plt.show()
```

OUTPUT-

```
Accuracy: 0.7662921348314606
              precision    recall  f1-score   support

           0       0.75      0.73      0.74       401
           1       0.78      0.80      0.79       489

    accuracy                           0.77       890
   macro avg       0.76      0.76      0.76       890
weighted avg       0.77      0.77      0.77       890
```



PROGRAM- 4

AIM- Implement the K-nearest Neighbour algorithm.

TOOL USED- VISUAL STUDIO CODE

SOURCE CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        predictions = [self._predict(x) for x in X]
        return np.array(predictions)

    def _predict(self, x):
        distances = [np.linalg.norm(x - x_train) for x_train in self.X_train]
        k_indices = np.argsort(distances)[:self.k]
        k_nearest_labels = [self.y_train[i] for i in k_indices]
```

```
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

file_path = "C:\\Users\\km367\\OneDrive\\Pictures\\Documents\\collegePlace.csv"
df = pd.read_csv(file_path)
lb=LabelBinarizer()
df['Gender']=lb.fit_transform(df['Gender'])
df=pd.get_dummies(df,columns=['Stream'], dtype=int)

print(df.info())
X = df.iloc[:, :6].values
y = df.iloc[:, 6].values
print(df.head())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

knn = KNN(k=3)
knn.fit(X_train, y_train)

predictions = knn.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")
cr=classification_report(y_test, predictions)
print(cr)
```

BEFORE PREPOCESSING:

	Age	Gender	Stream	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOrNot
0	22	Male	Electronics And Communication	1	8	1	1	1
1	21	Female	Computer Science	0	7	1	1	1
2	22	Female	Information Technology	1	6	0	0	1
3	21	Male	Information Technology	0	8	0	1	1
4	22	Male	Mechanical	0	8	1	0	1

AFTER PRECOSSING:

	Age	Gender	Internships	...	Stream_Electronics And Communication	Stream_Information Technology	Stream_Mechanical
0	22	1	1	...	1	0	0
1	21	0	0	...	0	0	0
2	22	0	1	...	0	1	0
3	21	1	0	...	0	1	0
4	22	1	0	...	0	0	1

OUTPUT-

Accuracy: 85.02%

	precision	recall	f1-score	support
0	0.81	0.89	0.85	280
1	0.89	0.82	0.85	314
accuracy			0.85	594
macro avg	0.85	0.85	0.85	594
weighted avg	0.85	0.85	0.85	594

PROGRAM- 5

AIM- Implement the Decision Tree algorithm.

TOOL USED- VISUAL STUDIO CODE

SOURCE CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
class DECISION:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        predictions = [self._predict(x) for x in X]
        return np.array(predictions)

    def _predict(self, x):
        distances = [np.linalg.norm(x - x_train) for x_train in self.X_train]
        k_indices = np.argsort(distances)[:self.k]
```

```
        k_nearest_labels = [self.y_train[i] for i in k_indices]
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

file_path = "C:\\Users\\km367\\OneDrive\\Pictures\\Documents\\collegePlace.csv"
df = pd.read_csv(file_path)
lb=LabelBinarizer()
df['Gender']=lb.fit_transform(df['Gender'])
df=pd.get_dummies(df,columns=['Stream'], dtype=int)

print(df.info())
X = df.iloc[:, :6].values
y = df.iloc[:, 6].values
print(df.head())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

dtree=DecisionTreeClassifier()
dtree.fit(X_train, y_train)

predictions = dtree.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")
cr=classification_report(y_test, predictions)
print(cr)
```

OUTPUT-

```
Accuracy: 87.54%
              precision    recall  f1-score   support

           0       0.83      0.93      0.88       280
           1       0.93      0.83      0.88       314

    accuracy                           0.88       594
   macro avg       0.88      0.88      0.88       594
weighted avg       0.88      0.88      0.88       594
```

PROGRAM- 6

AIM- Implement the Random Forest algorithm.

TOOL USED- VISUAL STUDIO CODE

SOURCE CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        predictions = [self._predict(x) for x in X]
        return np.array(predictions)

    def _predict(self, x):
        distances = [np.linalg.norm(x - x_train) for x_train in self.X_train]
        k_indices = np.argsort(distances)[:self.k]
```

```
        k_nearest_labels = [self.y_train[i] for i in k_indices]
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

file_path = "C:\\Users\\km367\\OneDrive\\Pictures\\Documents\\collegePlace.csv"
df = pd.read_csv(file_path)
lb=LabelBinarizer()
df['Gender']=lb.fit_transform(df['Gender'])
df=pd.get_dummies(df,columns=['Stream'], dtype=int)

print(df.info())
X = df.iloc[:, :6].values
y = df.iloc[:, 6].values
print(df.head())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

rforest=RandomForestClassifier()
rforest.fit(X_train, y_train)

predictions = rforest.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")
cr=classification_report(y_test, predictions)
print(cr)
```

OUTPUT-

Accuracy: 87.37%

	precision	recall	f1-score	support
0	0.83	0.93	0.87	280
1	0.93	0.83	0.87	314
accuracy			0.87	594
macro avg	0.88	0.88	0.87	594
weighted avg	0.88	0.87	0.87	594

PROGRAM- 7

AIM- Implement the Naive Bayesian Classification algorithm.

TOOL USED- VISUAL STUDIO CODE

SOURCE CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report
class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        predictions = [self._predict(x) for x in X]
        return np.array(predictions)

    def _predict(self, x):
        distances = [np.linalg.norm(x - x_train) for x_train in self.X_train]
        k_indices = np.argsort(distances)[:self.k]
```

```
        k_nearest_labels = [self.y_train[i] for i in k_indices]
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

file_path = "C:\\Users\\km367\\OneDrive\\Pictures\\Documents\\collegePlace.csv"
df = pd.read_csv(file_path)
lb=LabelBinarizer()
df['Gender']=lb.fit_transform(df['Gender'])
df=pd.get_dummies(df,columns=['Stream'], dtype=int)

print(df.info())
X = df.iloc[:, :6].values
y = df.iloc[:, 6].values
print(df.head())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

dtree=GaussianNB()
dtree.fit(X_train, y_train)

predictions = dtree.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")
cr=classification_report(y_test, predictions)
print(cr)
```

OUTPUT-

```
Accuracy: 77.78%
              precision    recall  f1-score   support

           0       0.74      0.82      0.78       280
           1       0.82      0.74      0.78       314

    accuracy                           0.78       594
   macro avg       0.78      0.78      0.78       594
weighted avg       0.78      0.78      0.78       594
```

PROGRAM- 8

AIM- Implement the Support Vector Machine algorithm.

TOOL USED- VISUAL STUDIO CODE

SOURCE CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import classification_report
class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        predictions = [self._predict(x) for x in X]
        return np.array(predictions)

    def _predict(self, x):
        distances = [np.linalg.norm(x - x_train) for x_train in self.X_train]
        k_indices = np.argsort(distances)[:self.k]
```

```
        k_nearest_labels = [self.y_train[i] for i in k_indices]
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

file_path = "C:\\Users\\km367\\OneDrive\\Pictures\\Documents\\collegePlace.csv"
df = pd.read_csv(file_path)
lb=LabelBinarizer()
df['Gender']=lb.fit_transform(df['Gender'])
df=pd.get_dummies(df,columns=['Stream'], dtype=int)

print(df.info())
X = df.iloc[:, :6].values
y = df.iloc[:, 6].values
print(df.head())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

sve=SVC()
sve.fit(X_train, y_train)

predictions = sve.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")
cr=classification_report(y_test, predictions)
print(cr)
```

OUTPUT-

```
Accuracy: 72.90%
              precision    recall  f1-score   support

           0       0.71      0.72      0.72       280
           1       0.75      0.73      0.74       314

    accuracy                           0.73       594
   macro avg       0.73      0.73      0.73       594
weighted avg       0.73      0.73      0.73       594
```

PROGRAM- 9

AIM- Implement the Perceptron Neural Learning Network..

TOOL USED- VISUAL STUDIO CODE

SOURCE CODE –

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
dataset= pd.read_csv("C:\\Users\\km367\\OneDrive\\Pictures\\Documents\\collegePlace.csv")
le_stream = LabelEncoder()
le_gender = LabelEncoder()
dataset['Stream'] = le_stream.fit_transform(dataset['Stream'])
dataset['Gender'] = le_gender.fit_transform(dataset['Gender'])
X = dataset.drop('PlacedOrNot', axis=1)
y = dataset['PlacedOrNot']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
mlp = MLPClassifier(hidden_layer_sizes=(10,), activation='relu', solver='adam', max_iter=1000, random_state=1)
mlp.fit(X_train, y_train)
y_pred = mlp.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy score: {accuracy * 100:.2f}%")
cr=classification_report(y_test, y_pred)
print(cr)
```

OUTPUT-

```
Accuracy score: 86.87%
              precision    recall  f1-score   support

           0       0.81      0.94      0.87       280
           1       0.94      0.80      0.87       314

    accuracy                           0.87       594
   macro avg       0.88      0.87      0.87       594
weighted avg       0.88      0.87      0.87       594
```

PROGRAM- 10

AIM- Implement the Backpropagation Learning using Python.

TOOL USED- VISUAL STUDIO CODE

SOURCE CODE –

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv("C:\\Users\\km367\\OneDrive\\Pictures\\Documents\\collegePlace.csv")
df['Stream'] = LabelEncoder().fit_transform(df['Stream'])
df['Gender'] = LabelEncoder().fit_transform(df['Gender'])

X = df.drop('PlacedOrNot', axis=1)
y = df['PlacedOrNot']

X = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred) * 100:.2f}%")
print("\nClassification Report:\n", classification_report(y_test,y_pred))
cr=classification_report(y_test, y_pred)
print(cr)
```

OUTPUT-

```
Accuracy: 86.36%

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.93      0.87       280
           1       0.93      0.80      0.86       314

    accuracy                           0.86       594
   macro avg       0.87      0.87      0.86       594
weighted avg       0.87      0.86      0.86       594
```