**Exp 5**

**You**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

ChatGPT

**ChatGPT**

Sure, here's a breakdown of each line:

1. **import numpy as np**: This line imports the NumPy library, which is commonly used for numerical computations in Python. It is aliased as **np**, which allows you to use NumPy functions and objects with the shorter name **np**.
2. **import matplotlib.pyplot as plt**: This line imports the **pyplot** module from the Matplotlib library, which is used for creating visualizations such as plots, charts, and graphs. It is aliased as **plt** for convenience.
3. **import pandas as pd**: This line imports the Pandas library, which is used for data manipulation and analysis in Python. It is aliased as **pd** for convenience, allowing you to access Pandas functions and objects using the shorter name **pd**.

User

**You**

**dataset = pd.read\_csv(r"D:\College\TE\SEM-2\Practical\DSBDA\5\Social\_Network\_Ads.csv")**

ChatGPT

**ChatGPT**

This line reads a CSV file named "Social\_Network\_Ads.csv" into a Pandas DataFrame object named **dataset**. The CSV file is located at the specified path: **"D:\College\TE\SEM-2\Practical\DSBDA\5\Social\_Network\_Ads.csv"**. The **r** before the string indicates that it's a raw string literal, which is used to prevent Python from treating backslashes **\** as escape characters.

User

**You**

**dataset.head()**

ChatGPT

**ChatGPT**

This line displays the first few rows of the DataFrame **dataset** using the **head()** method. By default, it shows the first 5 rows, giving you a quick overview of the structure and contents of your dataset.

User

**You**

**dataset.info()**

ChatGPT

**ChatGPT**

This line prints a concise summary of the DataFrame **dataset** using the **info()** method. It provides information about the DataFrame's columns, including their names, data types, and the number of non-null values in each column. Additionally, it gives an overall summary of the DataFrame, including the total number of entries and memory usage. This summary is helpful for understanding the structure and characteristics of the dataset.

User

**You**

**dataset.isnull().sum()**

ChatGPT

**ChatGPT**

This line calculates the number of missing values (null values) in each column of the DataFrame **dataset** using the **isnull()** method to create a boolean mask indicating where null values are present, and then **sum()** method to count the number of **True** values (which represent null values) for each column. It returns a Series object containing the sum of null values for each column in the dataset. This information is useful for identifying any missing data that may need to be handled during data preprocessing.

User

**You**

**dataset.shape**

ChatGPT

**ChatGPT**

This line returns the shape of the DataFrame **dataset**, which is a tuple representing the number of rows and columns in the dataset. The first element of the tuple represents the number of rows, and the second element represents the number of columns. So, **dataset.shape** would return something like (rows, columns).

User

**You**

**x = dataset.iloc[:, [2, 3]].values**

**y = dataset.iloc[:, 4].values**

ChatGPT

**ChatGPT**

These lines extract the feature variables and the target variable from the DataFrame **dataset**:

1. **x = dataset.iloc[:, [2, 3]].values**: This line selects the columns at index positions 2 and 3 (0-indexed) from the DataFrame **dataset** using **iloc**, which is primarily integer-location based indexing for selection by position. It extracts the feature variables into the variable **x** as a NumPy array.
2. **y = dataset.iloc[:, 4].values**: This line selects the column at index position 4 from the DataFrame **dataset** using **iloc** and extracts it as the target variable **y**. It also converts it into a NumPy array.

User

**You**

print(x)

ChatGPT

**ChatGPT**

This line prints the contents of the variable **x**, which contains the feature variables extracted from the DataFrame **dataset**. It displays these variables as a NumPy array.

User

**You**

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

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.30, random\_state = 0)**

ChatGPT

**ChatGPT**

These lines split the dataset into training and testing sets using the **train\_test\_split** function from the **sklearn.model\_selection** module:

1. **from sklearn.model\_selection import train\_test\_split**: This line imports the **train\_test\_split** function, which is used to split datasets into random train and test subsets.
2. **x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.30, random\_state = 0)**: This line splits the feature variables (**x**) and the target variable (**y**) into training and testing sets. It assigns 70% of the data to the training set (**x\_train** and **y\_train**) and 30% to the testing set (**x\_test** and **y\_test**). The **random\_state** parameter sets the random seed for reproducibility, ensuring that the split is the same each time the code is run.

User

**You**

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**x\_train = sc.fit\_transform(x\_train)**

**x\_test = sc.transform(x\_test)**

ChatGPT

**ChatGPT**

These lines standardize the feature variables using the **StandardScaler** from the **sklearn.preprocessing** module:

1. **from sklearn.preprocessing import StandardScaler**: This line imports the **StandardScaler** class, which is used to standardize features by removing the mean and scaling to unit variance.
2. **sc = StandardScaler()**: This line creates an instance of the **StandardScaler** class named **sc**.
3. **x\_train = sc.fit\_transform(x\_train)**: This line standardizes the feature variables in the training set (**x\_train**) by fitting the scaler to the data and transforming it. It calculates the mean and standard deviation of each feature in the training set and scales the features accordingly.
4. **x\_test = sc.transform(x\_test)**: This line standardizes the feature variables in the testing set (**x\_test**) using the mean and standard deviation calculated from the training set. It transforms the testing set based on the scaling parameters learned from the training set.

User

**You**

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

**classifier = LogisticRegression(random\_state = 0)**

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

1. **Import LogisticRegression**: This line imports the logistic regression classifier from scikit-learn.
2. **Instantiate Classifier**: Here, a logistic regression classifier object is created, setting the random state for reproducibility.
3. **Fit the Model**: This line trains the classifier using the provided training data (**x\_train** and **y\_train**). The model learns the relationship between features and labels.

User

**You**

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

ChatGPT

**ChatGPT**

This line predicts the labels for the test data (**x\_test**) using the trained logistic regression classifier (**classifier**). The predicted labels are stored in the variable **y\_pred**.

User

**You**

**print(y\_pred)**

ChatGPT

**ChatGPT**

Printing **y\_pred** will display the predicted labels for the test data. These predictions are generated by the logistic regression classifier based on the features in **x\_test**.

User

**You**

**print(y\_test)**

ChatGPT

**ChatGPT**

Printing **y\_test** will display the actual labels for the test data. These are the true labels against which the predictions made by the classifier (**y\_pred**) can be compared to evaluate the model's performance.

User

**You**

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

**cm = confusion\_matrix(y\_test, y\_pred)**

**print(cm)**

ChatGPT

**ChatGPT**

This code computes the confusion matrix using the predicted labels (**y\_pred**) and the actual labels (**y\_test**). The confusion matrix provides a summary of the model's performance by comparing the predicted classes against the true classes. It is a table with four cells showing the counts of true positive, false positive, true negative, and false negative predictions. Printing **cm** will display this confusion matrix.

User

**You**

**Accuracy=(74+31)/120**

**Accuracy**

ChatGPT

**ChatGPT**

To calculate accuracy using the confusion matrix, you sum the counts of correct predictions (true positives and true negatives) and divide by the total number of predictions.

The sum of correct predictions (true positives + true negatives) is 74+22=9674+22=96.

So, the accuracy is:

Accuracy=74+22120=96120=0.8Accuracy=12074+22​=12096​=0.8

So, the accuracy is 0.8, or 80%.

User

**You**

**Error\_rate=(5+10)/120**

**Error\_rate**

ChatGPT

**ChatGPT**

The error rate, also known as misclassification rate, is calculated by summing the counts of incorrect predictions (false positives and false negatives) and dividing by the total number of predictions.

The sum of incorrect predictions (false positives + false negatives) is 15+9=2415+9=24.

So, the error rate is:

Error rate=15+9120=24120=0.2Error rate=12015+9​=12024​=0.2

So, the error rate is 0.2, or 20%.

User

**You**

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

**precision\_score(y\_test, y\_pred)**

ChatGPT

**ChatGPT**

The **precision\_score** function from scikit-learn calculates the precision of a classification model. Precision measures the proportion of true positive predictions out of all positive predictions made by the model.

To use it, you provide the true labels (**y\_test**) and the predicted labels (**y\_pred**).

This will print the precision score, which is a value between 0 and 1. A higher precision score indicates that the classifier made fewer false positive predictions relative to the total number of positive predictions it made.

User

**You**

**recall\_score(y\_test, y\_pred)**

ChatGPT

**ChatGPT**

The **recall\_score** function from scikit-learn calculates the recall of a classification model. Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions out of all actual positive instances in the data.

You provide the true labels (**y\_test**) and the predicted labels (**y\_pred**) to this function.

This will print the recall score, which is also a value between 0 and 1. A higher recall score indicates that the classifier captured more of the actual positive instances in the data.

User

**You**

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

**f1\_score(y\_test, y\_pred)**

ChatGPT

**ChatGPT**

The **f1\_score** function from scikit-learn calculates the F1 score, which is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall.

To use it, you provide the true labels (**y\_test**) and the predicted labels (**y\_pred**).

This will print the F1 score, which is a value between 0 and 1. A higher F1 score indicates better performance of the classifier in terms of both precision and recall.

Top of Form