Programe 1 :- *Write a program to implement Simple Linear Regression Using Python*

```
# import the necessary modules
#We import the pandas and numpy modules to read and manipulate the data

import panda as pd
import numpy as np

# The data is stored in a csv file, so the read_csv() method is used to read the data into a dataframe

dataset = pd.read_csv('student_scores.csv')

# We dedcribe() the data to see the average, min, max, std and count for each column

dataset.describe()

# This code is extracting the independent variable(X) and the dependent variable (y) from a given dataset.

The command 'iloc' is used to select columns and rows based on the index position in the dataset.

The 'X' is the independent variable, and the 'y' is the dependent variable.

The 'dataset.iloc[:, :-1].values' command extracts the values of all the columns except the last one, which contains the dependent variable, and assigns those values to the 'X' variable.

The 'dataset.iloc[:, 1].values' command extracts the values from the second column, which contains the dependent variable, and assigns those values to the 'y' variable.

X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values

# We split the data into two parts; right and test using the train_test_split() method from sklearn.model_selection import train_test_split

train_test_split: This function splits the data set into two separate parts – a training set and a test set. The training set is
```

used to build and train the machine learning model while the test set is used to evaluate its performance on unseen data. The function splits the input data set into two parts based on the parameters entered - test_size and random_state. The test_size parameter determines what percentage of the input data will be used for the test set, while the random_state parameter controls how the data is split between the sets.

```
from sklearn.model_selection import train_test_split

# This code is used to divide a dataset into training and testing sets for supervised learning.

X and y represent the independent and dependent variables in the dataset, respectively.

The train_test_split() function from the sklearn.model_selection library is used to split the data into two separate data sets: X_train, which is used to train a model; and X_test which is used to evaluate the trained model.

The test_size argument specifies the proportion of the data to be used for testing, while the random_state argument provides a seed for the random number generator to ensure the results are reproducible.

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# The above code imports the LinearRegression class from the sklearn.linear_model library. It then creates an object called 'regressor' which is an instance of the LinearRegression class. The fit() method provided by the LinearRegression class is then used to fit the regressor object using the X_train and y_train data which are two matricies representing the training data. The fit() method is used to adjust the coefficients of the model so that it can be used to predict the values of a given input.
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Check the shape of the training data
# This is a code which returns the shape of the X_train array/matrix. The shape is represented as a tuple containing the
```

```
number of rows and columns. For example, if X_train has 10 rows and
5 columns, then X_train.shape will return (10, 5).

X_train.shape

# Check the testing data

X_test

# We use the predict funcion to predict the score for the test
dataset
# This code uses the regressor created to predict the outcome of a
set of observations stored on X_test.  It stores its prediction of
the outcome of X_test in a new variable y_pred.

y_pred = regressor.predict(X_test)
y_pred

# Import the metrics module

from sklearn import metrics

# The code evaluates the performance of a linear regression model.
It uses the mean absolute error, mean squared error, and root mean
squared error metrics to do so.
# The metrics.mean_absolute_error(y_test, y_pred) function is used
to calculate the mean absolute error. It takes two parameters,
y_test for the true values of the target variable and y_pred for
the predicted values of the target variable.

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,
y_pred))
# The metrics.mean_squared_error(y_test, y_pred) function is used
to calculate the mean squared error. It also takes two parameters,
y_test for the true values of the target variable and y_pred for
the predicted values of the target variable.


print('Mean Squared Error:', metrics.mean_squared_error(y_test,
y_pred))

# The np.sqrt(metrics.mean_squared_error(y_test, y_pred)) function
is used to calculate the root mean squared error. It takes the mean
squared error as input and calculates the square root of it.
```

```
# Finally, the print statements are used to display the calculated values of the mean absolute error, mean squared error, and root mean squared error.

print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# We check the score() of the Linear Regression Model
# This code is calculating the accuracy of a Machine Learning algorithm. It is using the regressor variable, which is being used to assess the predictive accuracy of the model. It is using two variables: X_test and y_test. X_test is the input vector used for the testing data, and y_test is the corresponding output vector for the testing data. By calling the score() method on the regressor, the accuracy is calculated and returned.

regressor.score(X_test,y_test)
```

Programe 2 :- *Write a program to implement Multiple Linear Regression Using Python*

```
import pandas as pd
import numpy as np

dataset = pd.read_csv('house_data.csv')

dataset.shape

X = dataset.iloc[:, [5]].values

y = dataset.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

y

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
```

```
y_pred

from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,
y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,
y_pred))
print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

regressor.score(X_test,y_test)
```

Programe 3 :- *Write a program to implement Logistic Regression Using Python*

```
# import the necessary modules
#We import the pandas and numpy modules to read and manipulate the
data

import pandas as pd
import numpy as np

# Read the dataset
#We use the read_csv() method of the Pandas module to read the data
from the csv file 'User_Data' into a DataFrame.

dataset = pd.read_csv("User_Data.csv")

# Define predictor and target variables
#We define the predictor and target variables. We are using X to
represent the predictor variables, which is the first three columns
of the dataset. We assign the fourth column to the target variable
y.

x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# import the necessary modules
# We are also imported train_test_split to split the data into
training and test sets

from sklearn.model_selection import train_test_split

# Split the data into training and test data
```

```
# We use the train_test_split() method to divide the data into
training and test sets. We specify the test_size of 0.20, which is
20%.

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size
= 0.20,
random_state = 0)

# import the necessary modules
# We import the StandardScaler to standardize the data

from sklearn.preprocessing import StandardScaler

# Scale the data
# We use the StandardScaler() method of the sklearn.preprocessing
module to standardize the data.#We use the LinearRegression()
method to prediction the score.
# This method is imported from the sklearn.linear_model module

sc_x = StandardScaler()
xtrain = sc_x.fit_transform(X_train)
xtest = sc_x.transform(X_test)

# import the necessary modules
# We import LogisticRegression to generate the logistic regression
model.

from sklearn.linear_model import LogisticRegression

# Generate the logistic regression model
# We use the LogisticRegression() method of the
sklearn.linear_model module to create our logistic regression model.

classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, y_train)

# Make the predictions

# We use the predict() method of the LogisticRegression class to
generate the prediction results.

y_pred = classifier.predict(xtest)

# Print the predictions

# Print the prediction results generated by the algorithm.
```

```
y_pred


# Calculate the accuracy score 

# We use the accuracy_score() method of the sklearn.model_selection module to calculate the accuracy score.

from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```