Project 3

**Perdition of Student Marks with Linear Regression**

**OVERVIEW**

A project to understand and implement concepts of Linear Regression that will outline how the regression concept works. The prediction will be determined on the number of hours a student will study and the scores he will receive accordingly.

**Software Requirements**

1. Programming Language: Python

2. Environment: Jupyter Notebooks / Google Collab

3. Database: CSV(export type)

4. Operation System: Windows XP or above

5. Libraries Used: Pandas,Folium, Seaborn, Scikit, SKLEARN

6. Datasets used: Student Dataset

1. **Open a New Notebook and import the required libraries and read the csv file**

|  |  |
| --- | --- |
|  | import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  import scipy.stats as stats  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LinearRegression |

Description:

Importing required libraries for the project, scipy library contains scientific algorithms that are used in python for statistical formulas and sklearn features various classification, regression and clustering algorithms.

1. **Importing the Student Dataset**

df = pd.read\_csv('/content/data.csv')

Description:

Reading the input data file named ‘data’ which is in csv format present in the content file and the data frame is stored in variable named ‘df’.

1. **Viewing and Exploring the Data**

df

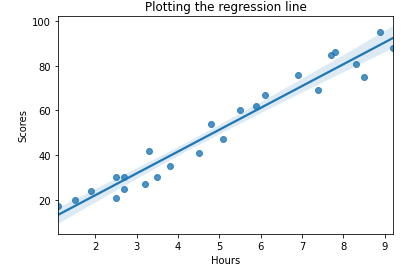
|  |  |
| --- | --- |
| Output:    Description:  The data frame is displayed which consists of 2 columns which are number of hours spent studying and number of marks scored.    df.shape    Output:  (25, 2)    Description:  It gives number of rows and columns present in data frame.  df.info()  Output:  <class 'pandas.core.frame.DataFrame'>  RangeIndex: 25 entries, 0 to 24  Data columns (total 2 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Hours 25 non-null float64  1 Scores 25 non-null int64  dtypes: float64(1), int64(1)  memory usage: 528.0 bytes  Description :  It gives the information regarding data frame which consists of class of data frame, range index which gives number of entries, data columns gives number of columns and number of values which are not null in each column and memory used by the data frame.    df.describe()    Output:      Description:  It gives the statistical information of the data frame which contains mean of the data,  Standard deviation, minimum value, maximum value and the count is 25 since we have 25 data records for hours and 25 data records for scores and thus the value is 25.    df.columns    Output:  Index(['Hours', 'Scores'], dtype='object')    Description:  It gives the names of columns and the data type of elements present in those columns.    df.corr()    Output:      Description:  It gives the measure of how correlated the data is for hours and hours it is 1 as both are identical. Similarly for hours and scores they are 97% correlated and it is evident that data is correlated.    df.isnull()    Description:  It gives the number of null values in the data record. If the value is null it returns a Boolean value ‘True’ and if it is not null it returns ‘False’. By default the value is taken as float if no data item is present its value is 0.00 by default.  Output:     1. **Visualizing the Linear Relation between Hours & Scores ( Drawing a joint Plot)**   sns.jointplot(df['Hours'], df['Scores'], kind = "reg").annotate(stats.pearsonr)  plt.show()    Description:  This plot is used to obtain plot joint bar graph for both hours and scores using pearson relation and annotated using stats folder. We can see from the graph there are no major ups and downs and we can say that the data set is correlated    Output: |  |

**5. Visualizing the Correlation**

sns.regplot(x = "Hours", y = "Scores", data = df)

plt.title("Plotting the regression line")

Output:



Description:

Here two columns are taken hours and scores and the data imported is data frame named df, and the plot is of regression type. By looking at the graph we can say that the data set is linearly related and strongly related.

## **6. Using Simple linear regression to predict the data as we only have two columns**.

Dividing Our Dataset into training and testing

X = df.iloc[:, :-1].values

y = df.iloc[:, -1].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_si ze = 0.30, random\_state = 0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

Output:

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, n ormalize=False)

Description:

Values are stored in variables X, y. Number of hours of the data set are stored in X and scores in y. train\_test\_split\_( ) is used to split the data set into training and testing data sets. random\_state gives the rate at which values are taken at random, X\_train contains hours which are part of training data set ,X\_test contains hours which are part of testing data. Similarly for y which contains scores. The attributes are stored in X variable. We specified "-1" as the range for columns since we wanted our attribute set to contain allthe columns except the last one, which is "Scores". Similarly the y variable contains the labels. We specified 1 for the label column since the index for "Scores" column is 1. Remember, the column indexes start with 0, with 1 being the second column. Linear regression model is stored in a variable named regressor and fit( ) is used to train the model.

**7. After Training the model , now performing Prediction**

y\_prediction = regressor.predict(X\_test)

y\_prediction

Output:

array([17.05366541, 33.69422878, 74.80620886, 26.8422321 , 60.12335883,39.56736879, 20.96909209, 78.72163554])

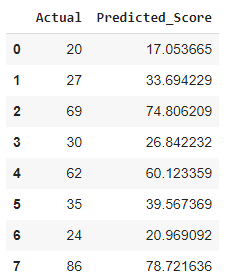
Description:

Using regressor which has been trained the testing data is passed as the input parameter the output predicted by the regressor is stored in variable named y\_prediction.

## **8.** **Comparing Actual vs Predicted Value** df1 = pd.DataFrame({'Actual': y\_test, 'Predicted\_Score': y\_predic tion})

df1

Output:



Description:

The actual values and predicted values are stored in a new data frame called df2 which contains two columns the actual and the values predicted by the model of the testing data set. With the help of above table we can find the variation and accuracy.

**9. Finding the number of students who have scored higher than cut off score**

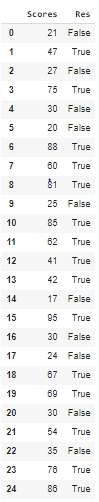
cut\_off = 40

df['Res']=df['Scores']>= 40

#del df['RES']

df

Output:



Description:

The cut off score is set to 40 and the scores obtained by each student are compared and the corresponding Boolean values are stored.

**10. Marking result of each person whether pass or fail**

conditions =[(df['Scores'] >= 40),

               (df['Scores']<40)]

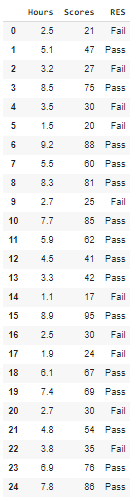
values = ['Pass', 'Fail']

df['RES'] = np.select(conditions, values)

del df['Res']

df

Output:



Description:

Creating a list of values we want to assign for each condition and Creating a new column using np.select ( ) to assign values to it using our lists as arguments.

**11. Calculating the accuracy of model designed in predicting the values**

r2\_score = regressor.score(X\_test,y\_test)

print(r2\_score\*100,'%')

Output:

95.68211104435257 %

Description:

The accuracy of the model is calculated using .score( ) by giving testing data set as input parameter and the model is 95.7% accurate in predicting the values.

**12. Calculating coefficients of linear regression (b0 & b1)**

print('intercept:', regressor.intercept\_)

print('slope:', regressor.coef\_)

Output:

intercept: 2.3708153823418883

slope: [9.78856669]

Description:

The coefficients of linear regression are calucated using .intercept\_ to obtain b0 value and coef\_ to obtain b1 value which are the intercept and slope of linear regression equation.

|  |
| --- |
|  |

**Conclusion**

The values have been predicted based on the number of hours a student will study and the score he/she receives accordingly, using linear regression algorithm and a linear regression model has been designed which can predict the marks that a student will score when provided with number of hours spent studying by that particular student with an accuracy of approximately 96%.

|  |  |
| --- | --- |
|  |  |