Project 3

**Prediction of Student Marks with Linear Regression**

**OVERVIEW**

A project to understand and implement the 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 libraires 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 |

**Description:**

Importing all the needed libraries.

1. **Importing the Student Dataset**

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

**Description:**

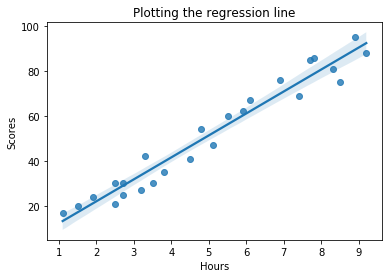
Loading the data to a pandas dataframe.

1. **Viewing and Exploring the Data**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| print("Now our data is loaded")  df  **Output:**      **Description:**  Viewing the data to know how they are arranged.  df.shape  (25, 2)  df.info()  <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:**  Understanding data frame shape like how many col and row we have in given data.  df.describe()   |  | **Hours** | **Scores** | | --- | --- | --- | | **count** | 25.000000 | 25.000000 | | **mean** | 5.012000 | 51.480000 | | **std** | 2.525094 | 25.286887 | | **min** | 1.100000 | 17.000000 | | **25%** | 2.700000 | 30.000000 | | **50%** | 4.800000 | 47.000000 | | **75%** | 7.400000 | 75.000000 | | **max** | 9.200000 | 95.000000 |   df.corr()    **Description:**  We use Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. When this method is applied to a series of string, it returns a different output which is shown in the examples below. Return type: Statistical summary of data frame.  Then we check for null values in the data frames. If null values are found we will replace null value with Na/Nan. Then finding the correlations using corr(). **corr()** is used to find the pairwise correlation of all columns in the dataframe. Any Na values are automatically excluded. For any non-numeric data type columns in the dataframe it is ignored.   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:**  Here we visuals the linear relation using joint plot which helps us to a plot of two variables with bivariate and univariate graphs. This function provides a convenient interface to the JointGrid class, with several canned plot kinds. This is intended to be a fairly lightweight wrapper; if you need more flexibility, you should use JointGrid directly. We can see the linear line where through this we can state that the given data is strong. |  |

1. **Visualizing the Correlation**

sns.regplot(x="Hours", y="Scores", data=df) plt.title("Plotting the regression line")



**Description:**

From this regplot we can find whether that is strong or weak. This is dataframe where each column is a variable and each row is an observation.

## 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

**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)

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

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

Out[27]:

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

**Description:**

Dividing the data into two datasets training dataset and testing dataset. Then by performing linear regression test on training data we get prediction values.

**After Training now performing Prediction**

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

y\_pred

Out[28]:

array([17.05366541, 33.69422878, 74.80620886, 26.8422321 , 60.12335883,

39.56736879, 20.96909209, 78.72163554])

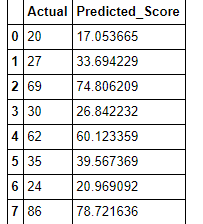
**Description:**

Predicting the data on which linear regression is done. Then we need to compare predicted values to accurate values.

## Comparing Actual vs Predicted Value

df1 = pd.DataFrame({'Actual': y\_test, 'Predicted\_Score': y\_pred})

df1



**Description:**

By visualizing the Data in the dataframe we can get to some conclusion and to make sure our conclusion is true we need to perform further more operations on it.By finding the accuracy of those data we can come to a conclusion that our prediction is true to certain extinct .

|  |
| --- |
|  |

**Conclusion**

Through this we can analyse student’s ability and then can have chance to make them concentrate more and make them sure that they can score good marks. By dividing the data into training set and testing set we can get the results that students are getting improved or not.

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