Project 3

**Perdition 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 :

1. **Importing the Student Dataset**

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

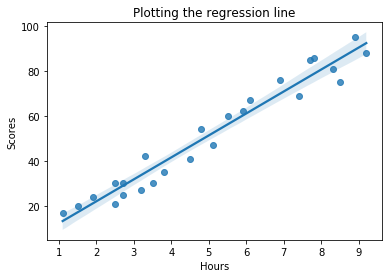
Description :

1. **Viewing and Exploring the Data**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. print("Now our data is loaded") 2. df   Output:      Description :  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 :  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 :   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 : |  |

1. **Visualizing the Correlation**

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



Description :

## 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 :

**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,

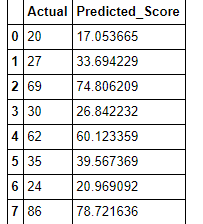
39.56736879, 20.96909209, 78.72163554])

Description :

## Comparing Actual vs Predicted Value

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

df1



Description :

|  |
| --- |
|  |

**Conclusion**

**To be filled by the trainee.**

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