**GRIP:The Sparks Foundation**

**Data Science and Business Analytics Intern**

**TASK 1:Prediction using Supervised ML**

In [46]:

***#importing the required libraries***

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

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

In [47]:

***#reading the dataframe from remote link***

df = pd.read\_csv("ML.Dataset.txt") In [48]:

***#now let us observe the dataset***

Df

|  |  |  |  |
| --- | --- | --- | --- |
|  | Hours | Scores | |
| 0 | 2.5 | 21 |
| 1 | 5.1 | 47 |
| 2 | 3.2 | 27 |
| 3 | 8.5 | 75 |
| 4 | 3.5 | 30 |
| 5 | 1.5 | 20 |
| 6 | 9.2 | 88 |
| 7 | 5.5 | 60 |
| 8 | 8.3 | 81 |
| 9 | 2.7 | 25 |
| 10 | 7.7 | 85 |
| 11 | 5.9 | 62 |
| 12 | 4.5 | 41 |
| 13 | 3.3 | 42 |
| 14 | 1.1 | 17 |
| 15 | 8.9 | 95 |
| 16 | 2.5 | 30 |
| 17 | 1.9 | 24 |
| 18 | 6.1 | 67 |
| 19 | 7.4 | 69 |
| 20 | 2.7 | 30 |
| 21 | 4.8 | 54 |
| 22 | 3.8 | 35 |
| 23 | 6.9 | 76 |
| 24 | 7.8 | 86 |

Out[48]:

**Exploratory Data Analysis**

In [49]:

df.columns

Out[49]:

Index(['Hours', 'Scores'], dtype='object')

In [50]:

***#to find more information***

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

In [51]:

df.describe()

Out[51]:

|  | **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 |

In [52]:

***#to find the number of columns and rows***

df.shape

Out[52]:

(25, 2)

**Outlier Removal**

In [53]:

**def** null\_detection(df):

num\_cols = []

count = 0

t = []

**for** i **in** num\_cols:

z = np.abs(stats.zscore(df[i]))

**for** j **in** range(len(z)):

**if** z[j]>3 **or** z[j]<-3:

t.append(j)

count+=1

df = df.drop(list(set(t)))

df = df.reset\_index()

df = df.drop('index', axis=1)

print(count)

**return** df

In [54]:

df = null\_detection(df)

0

Hence there is no outliner in the dataset

**Visualizing the dataset**

In [69]:

*#plotting the dataset*

plt.rcParams["figure.figsize"] = [10,5]

df.plot(kind='line', x='Scores', y='Hours',style='.',color='blue',)

plt.xlabel('Scores')

plt.ylabel('Hours')

plt.grid()

plt.show()

Chart, scatter chart

Description automatically generated

*#determining the correlation between the variables*

Graphical user interface, text, application

Description automatically generated

**Data preperation**

*#using iloc function we will divide the data*

X = df.iloc[:, :1].values

Y = df.iloc[:, 1:].values

*#Splitting data into training and testing data*

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)

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(X\_train , Y\_train)

LinearRegression()

**Visualizing the model**

In [68]:

line = model.coef\_\*X + model.intercept\_

plt.rcParams["figure.figsize"] = [10,5]

plt.scatter(X\_train , Y\_train,color= 'blue')

plt.plot(X, line , color= 'black');

plt.xlabel('Hours')

plt.ylabel('Scores')

plt.grid()

plt.show()Chart, scatter chart

Description automatically generated

**Making predictions**

In [62]:

print(X\_test)

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

[[1.5]

[3.2]

[7.4]

[2.5]

[5.9]]

In [63]:

Y\_test

Out[63]:

array([[20],

[27],

[69],

[30],

[62]], dtype=int64)

In [64]:

y\_pred

Out[64]:

array([[16.88414476],

[33.73226078],

[75.357018 ],

[26.79480124],

[60.49103328]])

In [65]:

***#comparing Actual vs Predicted***

comp= pd.DataFrame({'Actual':[Y\_test],'Predictd':[y\_pred]})

comp

Out[65]:

|  | **Actual** | **Predicted** |
| --- | --- | --- |
| **0** | [[20], [27], [69], [30], [62]] | [[16.884144762398037], [33.73226077948984], [7... |

In [70]:

*#Testing with my data*

hours = 8.20

my\_pred = model.predict([[hours]])

print("The predicted score if a person sudies for", hours, "hours is", my\_pred[0])

In [67]:

The predicted score if a person sudies for 8.2 hours is [83.28554318]

*#Evaluating the model*

**from sklearn import metrics**

**print('Mean Absolute Error:',metrics.mean\_absolute\_error(Y\_test, y\_pred))**

Mean Absolute Error: 4.183859899002982

**Thank you**

Nitesh sai.c