**MACHINE LEARNING MODEL**

TO EXPLORE SUPERVISED MACHINE LEARNING MODEL

In this regression model we have to predict the percentage of marks that a student is expected to score based upon the number of hours they studied.

It is a simple regression task with two variables.

In this model particularly I have predicted score if student study for **9.25** Hrs day.

**WHAT IS SUPERVISED MACHINE LEARNING?**

Supervised Machine Learning model teaches the model by providing a dataset with example input and output.

Human teacher expertise is used to tell the model which output is correct.

The steps for designing a supervised model are: -

**INPUT > MODEL > OUTPUT/PREDICTION**

Supervised machine learning models are further grouped into: -

1. Classification
2. Regression

**Regression** is further divided into…

1. Linear Regression
2. Logistic Regression

Steps involved in the coding part of this project: -

1. Import libraries
2. Reading data from remote link
3. Plotting distribution of scores
4. Preparing the data
5. Training the Algorithm
6. Plotting Regression line
7. Making predictions
8. Comparing Actual and Predicted
9. Testing of data asked to perform
10. Evaluating the Model (Mean Absolute Error)

Average of sum of difference between predictions and actual values.

Mean Absolute Error=

**STEP1: -**

**Import of library files** is the first step here we will import **Pandas**, **Numpy**, **Matplotlib.**

1) **Pandas**: Pandas is used to create the data frame or to organise data.

2) **Numpy**: Numpy is used for the computation and processing of multidimensional and single

dimensional array.

3)**Matplotlib**: Matplotlib is Python plotting library used for plotting graphs.

4)**Seaborn:** Seaborn is a library using for making statistical graphics in python

**CODE: -**

*# Importing all libraries required in this notebook*

import pandas as pd

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

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

%matplotlib inline

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

# **%matplotlib inline** (This line is used to set up a backend for the matplotlib library.)

**STEP2: -**

**Reading data from remote link “**<http://bit.ly/w-data>” used this link to extract the data.

**CODE: -**

*# Reading data from remote link*

url = “<http://bit.ly/w-data>”

data = pd.read\_csv(url)

print ("Data imported successfully. . .")

data.head(n)

Line 1 - Here we use ‘url’ to link our model to remote data.

Line 2 - Here **‘pd.read\_csv(url)’** is used to read the csv file in the ‘**url**’ here **CSV**

(**C**omma **S**eparated **V**alues) is list of data stored in a url.

Line 3 - print data has been imported successfully.

Line 4 – Here **‘data. head (n)’** is used to get the first ten values of the list.

**STEP3: -**

**Plotting distribution of scores** plot the scores in graph this graph shows the relation between Hours Studied and Percentage Score.

**CODE: -**

*# Plotting the distribution of scores*

data. plot(x='Hours', y='Scores', style='o')

plt. title('Hours vs Percentage')

plt. xlabel('Hours Studied')

plt. ylabel('Percentage Score')

plt. show()

Line 1 – Here we use  **‘plot()’** function to assign values of x-axis and y-axis.

Line 2 ­– Here **‘plt. title('Hours vs Percentage')’** is used to assign title to the graph.

Line 3 – Here **‘plt.xlabel('Hours Studied')’**and **‘plt.ylabel('Percentage Score')’**

is used to give names to X-axis and Y-axis.

Line 4 – Here **‘plt.show()’** is used to display the graph.

**STEP 4: -**

**Preparing the data** the next step is to divide the data into attributes (input) and labels (output).

**CODE: -**

X = s\_data.iloc[:, :-1].values

y = s\_data.iloc[:, 1].values

Line 1 - **‘X = s\_data.iloc[:, :-1].values’** This gives the row vector of the second last column values.

Line 2 - **‘y = s\_data.iloc[:, 1].values’**  This gives the row vector of the last column 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)

Line : – Train test split module is imported from sklearn model.

**STEP 5: -**

**Training the Algorithm** We have split our data into training and testing sets, and now is finally the time to train our algorithm.

**CODE: -**

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

regressor = LinearRegression()

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

print("Training complete.")

Line 1 – ‘**from** **sklearn.linear\_model** **import** LinearRegression’ to minimize the

residual sum of squares between the observed targets in the datset and the targets predicted

the linear approximation.

Line 2 –

Line 3 – *These lines are used for training the model.*

Line 4 –

**Training Complete.**

**STEP 6: -**

**Plotting Regression line**

**CODE: -**

*# Plotting the regression line*

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

*# Plotting for the test data*

plt.scatter(X, y)

plt.plot(X, line);

plt.show()

**(Plotting the regression Line)**

Line 1 - multiply each element of x with regressor.coef\_ and add regressor.intercept\_ to the product.

**(Plotting for the test data)**

Line 2 - **‘plt.scatter(X, y)’**  Here each value of dataset is represented using a dot.

Line 3 – **‘plt.plot(X, line);’** Here we use this code to plot regression line.

Line 4 – **‘plt.show()’** To display the output regression graph.

**STEP 7: -**

**Making predictions** now that we have trained our algorithm, it's time to make some predictions.

**CODE: -**

print(X\_test) *# Testing data - In Hours*

y\_pred = regressor.predict(X\_test) *# Predicting the scores*

Line 1 – Testing the data

Line 2 – Make predictions on the test data, execute the following script.

**STEP 8: -**

**Comparing Actual and Predicted** Here we will compare actual data and predicted data.

**CODE: -**

*# Comparing Actual vs Predicted*

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

df

Line 1 –

Here we are comparing the data model. Test and Predicted

Line 2 –

**STEP 9: -**

**Testing of data asked to perform** Testing for the score of the student at 9.25 hours.

**CODE: -**

*# You can also test with your own data*

hours = 9.25

own\_pred = regressor.predict([[hours]])

print ("No of Hours = **{}**".format(hours))

print ("Predicted Score = **{}**".format(own\_pred[0]))

Line 1 – **‘hours = 9.25’** Assign no of hours.

Line 2 – **‘own\_pred = regressor.predict(hours)’** code for prediction of own data.

Line 3 – **‘print ("No of Hours = {}".format(hours))’** Print number of hours.

Line 4 – **‘print ("Predicted Score = {}".format(own\_pred[0]))’** Print predicted score.

**STEP 10: -**

**Evaluating the Model**

The final step is to evaluate the performance of algorithm. This step is particularly important to compare how well different algorithms perform on a particular dataset. For simplicity here, we have chosen the mean square error.

**CODE: -**

**from** **sklearn** **import** metrics

print('Mean Absolute Error:',

metrics.mean\_absolute\_error(y\_test,y\_pred))

Line 1 – **‘from sklearn import metrics’**For assessing prediction error we import module

metrices.

Line 2 – **‘print('Mean Absolute Error:',**

**metrics.mean\_absolute\_error(y\_test,y\_pred)’**

To print absolute mean error using mean absolute error of metrices module.