**Assignment Topic: Linear Regression Algorithm (Prediction of Salary based on YearsExperience)**

* **Dataset:**

The dataset is provided in a CSV file named "Salary\_Data.csv". It contains two columns: "YearsExperience" (input feature) and "Salary" (output variable). The dataset consists of 29 samples.

* **Procedure for the Execution of the work:**

1. Open Jupyter-lab and creates a file named “Predict”. Inside this file I have upload the data sheet “ Salary\_Data.csv” and create a python file named “salarypredict.ipynb”.

2. Import the necessary libraries.

3. Import the Dataset and extract dependent and independent variables.

4. Visualizing the Dataset.

5. Splitting the dataset into the Training set and Test set.

6. Fitting Simple Linear Regression to the Training set.

7. Predicting the Test set results.

8. Visualizing the Training set result.

9. Visualizing the Test set result.

10. Finding the Residuals.

* **Detailed explanation of ML code:**
* **import numpy as np –** To imports the NumPy library and assigns it the alias “np”.
* **import matplotlib.pyplot as plt – To** imports the pyplot module from the Matplotlib library and assigns it the alias "plt".
* **import pandas as pd – To** imports the pandas library and assigns it the alias "pd".
* **import seaborn as sns - To** imports the seaborn library and assigns it the alias "sns".
* **%matplotlib inline – To** enables the inline display of Matplotlib plots within the jupyter notebook interface.
* **from sklearn.model\_selection import train\_test\_split -** To imports the train\_test\_split function from the model\_selection module of the scikit-learn library (sklearn).
* **salary\_data = pd.read\_csv('Salary\_Data.csv') -** This line reads the contents of the CSV file named "Salary\_Data.csv" and assigns it to the variable ‘salary\_data’. The read\_csv() function is provided by the pandas library and is used to read data from the CSV file into a pandas Data Frame.
* **X = salary\_data.iloc[:, :-1].values -** This line extracts independent variables from the ‘salary\_data’ Data Frame and assigns them to the variable X. .The iloc function is used to locate specific rows and columns in the Data Frame. [:, :-1] selects all rows and all columns except the last column.
* **y = salary\_data.iloc[:, 1].values -** This line extracts the or dependent variable from the salary\_data Data Frame and assigns it to the variable y. [:, 1] selects all rows and the second column (index 1) of the Data Frame
* **sns.histplot()** is a function provided by the seaborn library for creating a distribution plot.
* **sns.countplot(y='YearsExperience', data=salary\_data) -** It displays the count of occurrences of each unique value in a categorical variable.
* **sns.barplot(x='YearsExperience', y='Salary', data=salary\_data) -** It displays the relationship between a categorical variable and a numerical variable using rectangular bars.
* **sns.heatmap(salary\_data.corr()) -** It visualizes the correlation between variables in a matrix format using colors.
* **from sklearn.model\_selection import train\_test\_split -** To imports the train\_test\_split function from the model\_selection module of scikit-learn.
* **X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1/3, random\_state=0) -** train\_test\_split() is a function that splits the dataset into two or more subsets for training and testing purposes.
* test\_size=1/3 specifies the proportion of the dataset that should be allocated for testing.
* X\_train contains the training data for the features.
* X\_test contains the testing data for the features.
* y\_train contains the training data for the target variable.
* y\_test contains the testing data for the target variable.
* **from sklearn.linear\_model import LinearRegression -** imports the LinearRegression class from the linear\_model module of scikit-learn.
* **lr = LinearRegression() -** creates an instance of the LinearRegression class and assigns it to the variable lr.
* **lr.fit(X\_train, y\_train) -** The fit() method is called on the lr object to train the linear regression model.

* lr.predict() - It is used to make predictions based on the trained linear regression model.
* The predict() method takes the X\_test data as input and returns the predicted values for the target variable.
* The predicted values for the target variable are assigned to the variable y\_pred.
* plt.scatter() is a function provided by the matplotlib library for creating a scatter plot. It is used to plot individual data points in a two-dimensional space.
* plt.plot() is a function provided by the matplotlib library for creating a line plot. It is used to connect data points with a line.
* lr.predict(X\_train) uses the trained linear regression model (lr) to make predictions for the 'Salary' based on the 'Years of Experience' from the X\_train data.
* plt.title() is used to set the title of the plot**.**
* plt.xlabel() is used to set the label for the x-axis.
* plt.ylabel() is used to set the label for the y-axis.
* **plt.scatter(X\_test, y\_test, color='blue') -** This line creates a scatter plot using the test data.
* **plt.plot(X\_test, lr.predict(X\_test), color='red') -** This line creates a line plot based on the linear regression model's predictions.
* **plt.title('Salary vs Experience (Train set)') -** This line sets the title of the plot to 'Salary vs Experience (Train set)'.
* **plt.xlabel('Years of Experience') -** This line sets the label for the x-axis to 'Years of Experience'.
* **plt.ylabel('Salary') -** This line sets the label for the y-axis to 'Salary'.
* **plt.show() -** This line displays the plot on the screen.
* **from sklearn import metrics-**This line imports the metrics module from the sklearn (scikit-learn) library.
* **print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred)) -** calculates the Mean Absolute Error (MAE) between the actual target variable (y\_test) and the predicted target variable (y\_pred).
* **print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred)) -** calculates the Mean Squared Error (MSE) between the actual target variable (y\_test) and the predicted target variable (y\_pred).
* **print('RMSE:', np.sqrt(metrics.mean\_absolute\_error(y\_test, y\_pred))) -** calculates the Root Mean Squared Error (RMSE) based on the previously computed MAE. The np.sqrt function from the NumPy library is used to take the square root of the MAE value.

**Prepared by – SOUVIK SINGHA**