**Machine Learning Project Report**

Predict salary of an employee based on

Year of experience

Name –

Roll Number –

Subject –

Advisor –

* **Introduction**

In this project, I build a Simple Linear Regression model to study the linear relationship between Salary and Year of experience dataset for predicting salary based on Year of experience. I explained the basics of linear regression and its implementation in Python programming language using Scikit-learn. Scikit-learn is the popular machine learning library of Python programming language.

* **The problem statement**

The aim of building a machine learning model is to solve a problem and to define a metric to measure model performance. So, first of all I have to define the problem to be solved in this project. As described earlier, the problem is to model and investigate the linear relationship between Salary and Year of experience for predicting salary based on Year of experience

## Python libraries

I have Python and VScode installed on my system. It comes with most of the standard Python libraries I need for this project. The basic Python libraries used in this project are:-

* Numpy – It provides a fast numerical array structure and operating functions
* Pandas – It provides tools for data storage, manipulation and analysis tasks.
* Scikit-Learn – Scikit-learn is an open-source Python library that implements a range of machine learning, pre-processing, cross-validation, and visualization algorithms using a unified interface.
* Matplotlib – It is the basic plotting library in Python. It provides tools for making plots.

## Linear Regression

## Linear Regression is a statistical technique which is used to find the linear relationship between dependent and one or more independent variables. This technique is applicable for Supervised Learning Regression problems where we try to predict a continuous variable. Linear Regression can be further classified into two types – Simple and Multiple Linear Regression. In this project, I employ Simple Linear Regression technique where I have one independent and one dependent variable. It is the simplest form of Linear Regression where I fit a straight line to the data.

## Independent and Dependent Variables

## In this project, I refer Independent variable as Feature variable and Dependent variable as Target variable. These variables are also recognized by different names as follows: -

### **Independent variable**

Independent variable is also called Input variable(Year of experience) and is denoted by X. In practical applications, independent variable is also called Feature variable or Predictor variable. We can denote it as :-

Independent or Input variable (X) = Feature variable = Predictor variable

### **Dependent variable**

Dependent variable is also called Output variable(Salary) and is denoted by y. Dependent variable is also called Target variable or Response variable. It can be denote it as follows: -

Dependent or Output variable (y) = Target variable = Response variable

## Simple Linear Regression (SLR)

Simple Linear Regression (or SLR) is the simplest model in machine learning. It models the linear relationship between the independent and dependent variables.

In this project, there is one independent or input variable which represents the Year of experience data and is denoted by X. Similarly, there is one dependent or output variable which represents the Salary data and is denoted by y. We want to build a linear relationship between these variables. This linear relationship can be modelled by mathematical equation of the form:-

Y = β0 + β1\*X --------------- (1)

In this equation, X and Y are called independent and dependent variables respectively, β1 is the coefficient for independent variable and β0 is the constant term. β0 and β1 are called parameters of the model.

For simplicity, we can compare the above equation with the basic line equation of the form:-

y = ax + b ----------------- (2)

We can see that

slope of the line is given by, a = β1, and

intercept of the line by b = β0.

In this Simple Linear Regression model, we want to fit a line which estimates the linear relationship between X and Y.

## Exploratory data analysis

## First, I import the dataset into the dataframe with the standard read\_csv () function of pandas library and assign it to the dataset variable. Then, I conducted exploratory data analysis to get a feel for the data. I checked the dimensions of dataframe with the shape attribute of the dataframe. I viewed the top 5 rows of the dataframe with the pandas head() method. I viewed the dataframe summary with the pandas info() method and descriptive statistics with the describe() method.

## Intuition of Simple Linear Regression

## The Simple Linear Regression model starts with splitting the dataset into two sets – the training set and the test set. I import the class LinearRegression from sklearn.linear model and assign to an object of that class regressor and fit it on the training set with the fit method. In this step, the model learned the correlations between the training data (X\_train, y\_train). Now the model is ready to make predictions on the test data (X\_test). Hence, I predict on the test data using the predict method.

## Model slope and intercept term

The model slope is given by regressor.coef\_ and model intercept term is given by regressor.intercept\_. The estimated model slope and intercept values are 8692.01316868 and 29974.33208826927.

So, the equation of the fitted regression line is

y = 8692.01316868 \* x + 29974.33208826927

## Regression metrics for model performance

Now, it is the time to evaluate model performance.

For regression problems, there are two ways to compute the model performance.

RMSE (Root Mean Square Error) and R-Squared Value.

### RMSE -

RMSE is the standard deviation of the residuals. So, RMSE gives us the standard deviation of the unexplained variance by the model. It can be calculated by taking square root of Mean Squared Error. RMSE is an absolute measure of fit. It gives us how spread the residuals are, given by the standard deviation of the residuals. The more concentrated the data is around the regression line, the lower the residuals and hence lower the standard deviation of residuals. It results in lower values of RMSE. So, lower values of RMSE indicate better fit of data.

### R2 Score -

R2 Score is another metric to evaluate performance of a regression model. It is also called coefficient of determination. It gives us an idea of goodness of fit for the linear regression models. It indicates the percentage of variance that is explained by the model.

Mathematically,

R2 Score = Explained Variation/Total Variation

In general, the higher the R2 Score value, the better the model fits the data. Usually, its value ranges from 0 to 1. So, we want its value to be as close to 1. Its value can become negative if our model is wrong.

## Conclusion

## Here, I build a regression model and check the model RMSE which is equal to 4918.4200and R2 score is 0.9738

## In business decisions, the benchmark for the R2 score value is 0.7. It means if R2 score value >= 0.7, then the model is good enough to deploy on unseen data whereas if R2 score value < 0.7, then the model is not good enough to deploy. Our R2 score value has been found to be 0.9738. It means that this model explains 97.38 % of the variance in our dependent variable. So, the R2 score value confirms that the model is good enough to deploy because it does provide good fit to the data.

## Checking for Overfitting and Underfitting

## Underfitting - Underfitting means our model performs poorly fit on the training data. It means the model does not capture the relationships between the training data.

## **Underfitting – High bias and low variance**

Techniques to reduce underfitting:

1. Increase model complexity
2. Increase the number of features, performing feature engineering
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better results.

## **Overfitting - Overfitting** means our model performs overfitted on the training data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too many details and noise

## **Overfitting – High variance and low bias**

Techniques to reduce overfitting:

1. Increase training data.
2. Reduce model complexity.
3. Early stopping during the training phase

I calculate training set score as 0.9588. Similarly, I calculate test set score as 0.9738. The training set score is very good. So, the model learn the relationships appropriately from the training data. Thus, the model performs well on the training data. It is a clear sign of best fitting. Hence, I validated my finding that the linear regression model provide good fit to the data.

* **#Code**

**# Simple Linear Regression**

*## Importing the libraries*

## import numpy as np

## import matplotlib.pyplot as plt

## import pandas as pd

## ***## Importing the dataset***

## **dataset = pd.read\_csv('Salary\_Data.csv')**

## **X = dataset.iloc[:, :-1].values**

## **y = dataset.iloc[:, -1].values**

## ***##Exploratory data analysis***

## **print(dataset.shape)**

## **(35, 2)**

## **print(dataset.head())**

## **YearsExperience Salary**

## **0 1.1 39343**

## **1 1.3 46205**

## **2 1.5 37731**

## **3 2.0 43525**

## **4 2.2 39891**

## **print(dataset.info())**

## **<class 'pandas.core.frame.DataFrame'>**

## **RangeIndex: 35 entries, 0 to 34**

## **Data columns (total 2 columns):**

## **# Column Non-Null Count Dtype**

## **--- ------ -------------- -----**

## **0 YearsExperience 35 non-null float64**

## **1 Salary 35 non-null int64**

## **dtypes: float64(1), int64(1)**

## **memory usage: 624.0 bytes**

## **None**

## **print(dataset.describe())**

## YearsExperience Salary

## count 35.000000 35.000000

## mean 6.308571 83945.600000

## std 3.618610 32162.673003

## min 1.100000 37731.000000

## 25% 3.450000 57019.000000

## 50% 5.300000 81363.000000

## 75% 9.250000 113223.500000

## max 13.500000 139465.000000

## *## Splitting the dataset into the Training set and Test set*

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

## X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 0)

## *## Training the Simple Linear Regression model on the Training set*

## from sklearn.linear\_model import LinearRegression

## regressor = LinearRegression()

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

## LinearRegression()

## *## Predicting the Test set results*

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

## *## Compute model slope and intercept*

## a = regressor.coef\_

## b = regressor.intercept\_,

## print("Estimated model slope, a:" , a)

## print("Estimated model intercept, b:" , b)

## Estimated model slope, a: [8692.01316868]

## Estimated model intercept, b: (29974.33208826927)

## *##Regression metrics*

## from sklearn.metrics import mean\_squared\_error

## mse = mean\_squared\_error(y\_test, y\_pred)

## rmse = np.sqrt(mse)

## print("RMSE value: {:.4f}".format(rmse))

## RMSE value: 4918.4200

## from sklearn.metrics import r2\_score

## print ("R2 Score value: {:.4f}".format(r2\_score(y\_test, y\_pred)))

## R2 Score value: 0.9738

## *## Visualising the Training set results*

## plt.scatter(X\_train, y\_train, color = 'red')

## plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

## plt.title('Salary vs Experience (Training set)')

## plt.xlabel('Years of Experience')

## plt.ylabel('Salary')

## plt.show()

## C:\Users\Sahil\Desktop\1.png

## *## Visualising the Test set results*

## plt.scatter(X\_test, y\_test, color = 'red')

## plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

## plt.title('Salary vs Experience (Test set)')

## plt.xlabel('Years of Experience')

## plt.ylabel('Salary')

## plt.show()

## C:\Users\Sahil\Desktop\2.png

## *##Checking for Overfitting and Underfitting*

## print("Training set score:{:.4f}".format(regressor.score(X\_train,y\_train)))

## print("Test set score: {:.4f}".format(regressor.score(X\_test,y\_test)))

## Training set score: 0.9588

## Test set score: 0.9738

## *##Sample estimations*

## new\_pred\_salary = regressor.predict([[15]])

## print('The predicted salary of an employee with 15 years experience is ', new\_pred\_salary)

## The predicted salary of an employee with 15 years experience is [160354.52961852]

## new\_pred\_salary = regressor.predict([[12]])

## print('The predicted salary of an employee with 12 year’s experience is ', new\_pred\_salary)

## The predicted salary of an employee with 12 years experience is [160354.52961852]