## Bansilal Ramnath Agarwal Charitable Trust’s

Vishwakarma Institute of Technology, Pune-37

*(An Autonomous Institute of Savitribai Phule Pune University)*



**Department of AIDS**

|  |  |
| --- | --- |
| **Division** | A |
| **Roll Number** | 78 |
| **PRN Number** | 12320056 |
| **Name** | Mayank Kulkarni |

**Title:** Implementation of Linear Regression.

**Theory:**

**What is Linear Regression?**

Linear regression is a statistical method that is used to predict a continuous dependent variable (target variable) based on one or more independent variables (predictor variables). This technique assumes a linear relationship between the dependent and independent variables, which implies that the dependent variable changes proportionally with changes in the independent variables. In other words, linear regression is used to determine the extent to which one or more variables can predict the value of the dependent variable.

**Types of Linear Regression**

**Simple Linear Regression**

This is the simplest form of linear regression, and it involves only one independent variable and one dependent variable. The equation for simple linear regression is:  
y=β0+β1X*y*=*β*0​+*β*1​*X*  
where:

* Y is the dependent variable
* X is the independent variable
* β0 is the intercept
* β1 is the slope

**Multiple Linear Regression**

This involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:  
y=β0+β1X1+β2X2+………βnXn*y*=*β*0​+*β*1​*X*1+*β*2​*X*2+………*βn*​*Xn*  
where:

* Y is the dependent variable
* X1, X2, …, Xn are the independent variables
* β0 is the intercept
* β1, β2, …, βn are the slopes

**The goal of the algorithm is to find the best Fit Line equation that can predict the values based on the independent variables.**

**What is the best Fit Line?**

Our primary objective while using linear regression is to locate the best-fit line, which implies that the error between the predicted and actual values should be kept to a minimum. There will be the least error in the best-fit line.

The best Fit Line equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variables.



**Evaluation Metrics for Linear Regression**

The strength of any linear regression model can be assessed using various evaluation metrics. These evaluation metrics usually provide a measure of how well the observed outputs are being generated by the model.

The most used metrics are,

1. Coefficient of Determination or R-squared (R2)
2. Root Mean Squared Error (RSME) and Residual Standard Error (RSE)

**Step 1: Importing Necessary Libraries**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split, learning\_curve  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score

**Step 2: Loading and Viewing the Dataset**

data = pd.read\_csv('Salary\_dataset.csv')

data.head()

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0

data.shape

(30, 3)

data = data.drop(columns=['Unnamed: 0'])

data

YearsExperience Salary  
0 1.2 39344.0  
1 1.4 46206.0  
2 1.6 37732.0  
3 2.1 43526.0  
4 2.3 39892.0  
5 3.0 56643.0  
6 3.1 60151.0  
7 3.3 54446.0  
8 3.3 64446.0  
9 3.8 57190.0  
10 4.0 63219.0  
11 4.1 55795.0  
12 4.1 56958.0  
13 4.2 57082.0  
14 4.6 61112.0  
15 5.0 67939.0  
16 5.2 66030.0  
17 5.4 83089.0  
18 6.0 81364.0  
19 6.1 93941.0  
20 6.9 91739.0  
21 7.2 98274.0  
22 8.0 101303.0  
23 8.3 113813.0  
24 8.8 109432.0  
25 9.1 105583.0  
26 9.6 116970.0  
27 9.7 112636.0  
28 10.4 122392.0  
29 10.6 121873.0

**Step 3: Feature Selection and Splitting Data**

 x: Independent variable (Years of Experience).

 y: Dependent variable (Salary).

 The data is split into training (80%) and testing (20%) sets.

X = data[['YearsExperience']]  
y = data['Salary']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Step 4: Model Initialization and Training**

 A LinearRegression model is initialized.

 The model is trained using the training data (X\_train and y\_train).

model = LinearRegression()  
model.fit(X\_train, y\_train)

LinearRegression()

**Step 5: Making Predictions**

* Predictions are made on both the training and testing data.

y\_train\_pred = model.predict(X\_train)  
y\_test\_pred = model.predict(X\_test)  
  
data['Predicted Salary'] = model.predict(X)  
data['Difference'] = data['Salary'] - data['Predicted Salary']

**Step 6: Model Evaluations**

 Mean **Squared Error (MSE)** is calculated for both the training and testing sets.

 The **R² score** is computed to measure the model's goodness of fit.

train\_mse = mean\_squared\_error(y\_train, y\_train\_pred)  
test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)  
r2 = r2\_score(y\_test, y\_test\_pred)  
  
print(f'Mean Squared Error (Training): {train\_mse}')  
print(f'Mean Squared Error (Testing): {test\_mse}')  
print(f'R^2 Score: {r2}')

Mean Squared Error (Training): 27102249.731261354  
Mean Squared Error (Testing): 49830096.855908334  
R^2 Score: 0.9024461774180498

**Step 7: Adding Predictions and Differences to Data**

 The predicted salaries for all data points are added to the dataset.

 The difference between the actual and predicted salaries is also computed.

data['Predicted Salary'] = model.predict(X)  
data['Difference'] = data['Salary'] - data['Predicted Salary']

**Step 8: Saving the Updated Datasets**

* The updated dataset with predictions and differences is saved to a CSV file.

output\_file\_path = 'output.csv'   
data.to\_csv(output\_file\_path, index=False)  
  
print(data.head())

YearsExperience Salary Predicted Salary Difference  
0 1.2 39344.0 35688.779867 3655.220133  
1 1.4 46206.0 37573.542932 8632.457068  
2 1.6 37732.0 39458.305996 -1726.305996  
3 2.1 43526.0 44170.213658 -644.213658  
4 2.3 39892.0 46054.976722 -6162.976722

**Step 9: Plotting Actual vs Predicted Salary**

 The dataset is sorted based on YearsExperience for smooth plotting.

 Two lines are plotted: one for the actual salary and one for the predicted salary.

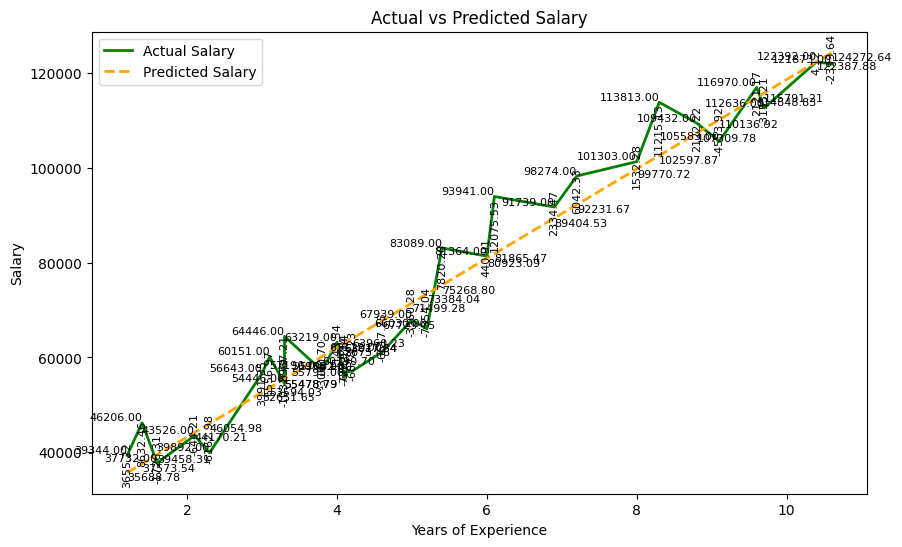
sorted\_indices = np.argsort(data['YearsExperience'])  
sorted\_X = data['YearsExperience'].iloc[sorted\_indices]  
sorted\_Y = data['Salary'].iloc[sorted\_indices]  
sorted\_pred = data['Predicted Salary'].iloc[sorted\_indices]  
  
plt.figure(figsize=(10, 6))  
plt.plot(sorted\_X, sorted\_Y, color='green', linewidth=2, label='Actual Salary', linestyle='-')  
plt.plot(sorted\_X, sorted\_pred, color='orange', linewidth=2, label='Predicted Salary', linestyle='--')  
  
**Step 10: Annotating the Plot**

Each point on the graph is annotated with the actual salary, predicted salary, and the difference between them.

for i in range(len(sorted\_X)):  
 actual = sorted\_Y.iloc[i]  
 predicted = sorted\_pred.iloc[i]  
 error = actual - predicted  
 plt.text(sorted\_X.iloc[i], actual, f"{actual:.2f}", ha='right', va='bottom', fontsize=8)  
 plt.text(sorted\_X.iloc[i], predicted, f"{predicted:.2f}", ha='left', va='top', fontsize=8)  
 plt.text(sorted\_X.iloc[i], actual - (error / 2), f"{error:.2f}", ha='center', va='center', fontsize=8, rotation=90)

**Step 11: Plotting the Regression Line**

The graph is labeled, and the regression line is displayed to show the relationship between YearsExperience and Salary.  
  
plt.xlabel('Years of Experience')  
plt.ylabel('Salary')  
plt.title('Actual vs Predicted Salary')  
plt.legend()  
plt.show()



 Observations: The actual and predicted salaries are closely aligned, indicating that the model performs well in predicting salaries based on years of experience. However, there is some variability in the predictions, particularly at certain points where the actual salary deviates significantly from the predicted line.

 Detail: The graph includes detailed salary values at each data point, which shows that while the overall trend is captured, there are fluctuations in the exact predictions.

sorted\_indices = np.argsort(data['YearsExperience'])  
sorted\_X = data['YearsExperience'].iloc[sorted\_indices]  
sorted\_Y = data['Salary'].iloc[sorted\_indices]  
sorted\_pred = data['Predicted Salary'].iloc[sorted\_indices]  
  
plt.figure(figsize=(10, 6))  
plt.plot(sorted\_X, sorted\_Y, color='green', linewidth=2, label='Actual Salary', linestyle='-')  
plt.plot(sorted\_X, sorted\_pred, color='orange', linewidth=2, label='Predicted Salary', linestyle='--')  
  
plt.xlabel('Years of Experience')  
plt.ylabel('Salary')  
plt.title('Actual vs Predicted Salary (Regression Line)')  
plt.legend()  
plt.show()



 Observation: This graph illustrates a more general trend comparison between actual and predicted salaries without the individual salary values. The predicted salary line (dashed orange) closely follows the actual salary line (green), showing a good fit with the data. The model seems to capture the overall trend very well, indicating strong linearity in the relationship between experience and salary.

 Detail: The lack of detailed salary values makes this graph cleaner and easier to interpret the overall trend.

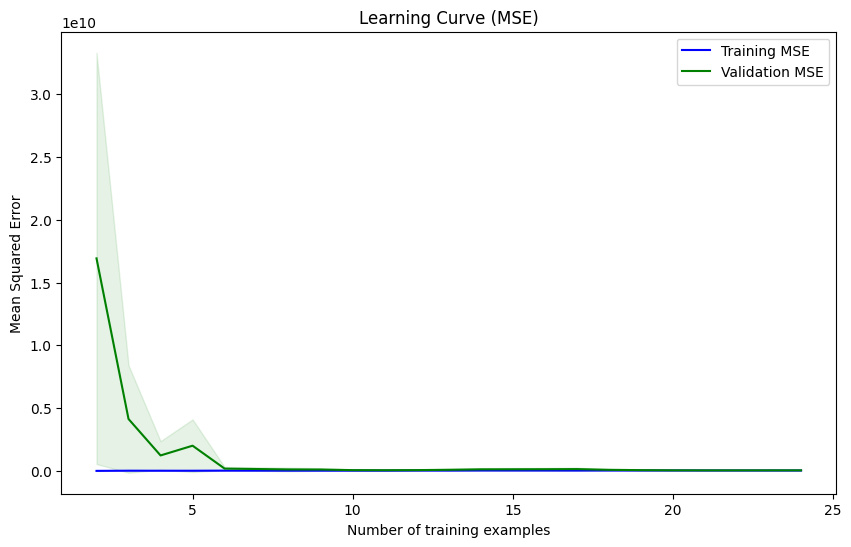
**Step 12: Generating a Learning Curve**

A learning curve is generated to show how the model's performance (in terms of MSE) changes as the size of the training set increases.

train\_sizes, train\_scores, test\_scores = learning\_curve(  
 model, X, y, cv=5, scoring='neg\_mean\_squared\_error', train\_sizes=np.linspace(0.1, 1.0, 20)  
)  
  
train\_scores\_mean = -train\_scores.mean(axis=1)  
train\_scores\_std = train\_scores.std(axis=1)  
test\_scores\_mean = -test\_scores.mean(axis=1)  
test\_scores\_std = test\_scores.std(axis=1)  
  
**Step 13: Plotting the Learning Curve**

The learning curve is plotted to visualize the training and validation MSEs.

plt.figure(figsize=(10, 6))  
plt.plot(train\_sizes, train\_scores\_mean, label='Training MSE', color='blue')  
plt.plot(train\_sizes, test\_scores\_mean, label='Validation MSE', color='green')  
  
plt.fill\_between(train\_sizes, train\_scores\_mean - train\_scores\_std, train\_scores\_mean + train\_scores\_std, alpha=0.1, color='blue')  
plt.fill\_between(train\_sizes, test\_scores\_mean - test\_scores\_std, test\_scores\_mean + test\_scores\_std, alpha=0.1, color='green')  
  
plt.xlabel('Number of training examples')  
plt.ylabel('Mean Squared Error')  
plt.title('Learning Curve (MSE)')  
plt.legend()  
plt.show()



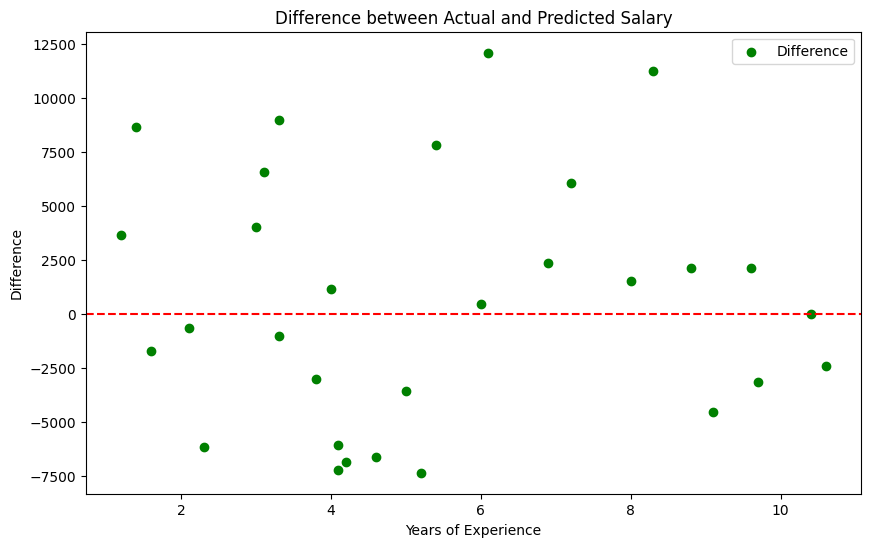
 Observation: The learning curve shows that the Mean Squared Error (MSE) for both training and validation decreases rapidly with the number of training examples and then stabilizes. The low and almost overlapping lines of training and validation MSE indicate that the model is well-trained without significant overfitting or underfitting issues.

 Detail: The validation MSE line is very close to the training MSE, which suggests that the model generalizes well to unseen data. The initial sharp drop in MSE suggests that the model quickly learns the relationship between the variables early in the training process.

**Step 14: Plotting the Difference between Actual and Predicted Salary**

A scatter plot is created to show the difference between the actual and predicted salaries as a function of years of experience.

plt.figure(figsize=(10, 6))  
plt.scatter(data['YearsExperience'], data['Difference'], color='green', label='Difference')  
plt.axhline(y=0, color='red', linestyle='--')  
plt.xlabel('Years of Experience')  
plt.ylabel('Difference')  
plt.title('Difference between Actual and Predicted Salary')  
plt.legend()  
plt.show()



 Errors **Vary:** Differences range widely, with both overpredictions and underpredictions.

 No **Clear Pattern:** Errors are scattered across all experience levels without a consistent trend.

 Outliers**:** Some predictions are significantly off, with extreme overpredictions and underpredictions.

 Slight **Overprediction Bias:** More points are above the zero line, especially at lower experience levels, indicating a tendency to overpredict salaries.

**Conclusion:** Thus, we have successfully implemented the linear regression on the salary dataset.