Experiment No.1

Title:Implementation of Linear Regression Model

Part A:Implementation of Linear Regression Model on Synthetic data

Importing Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Creating Sample Data

```
#Independent Variable
x=np.array(np.arange(0,8))
x
```

```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

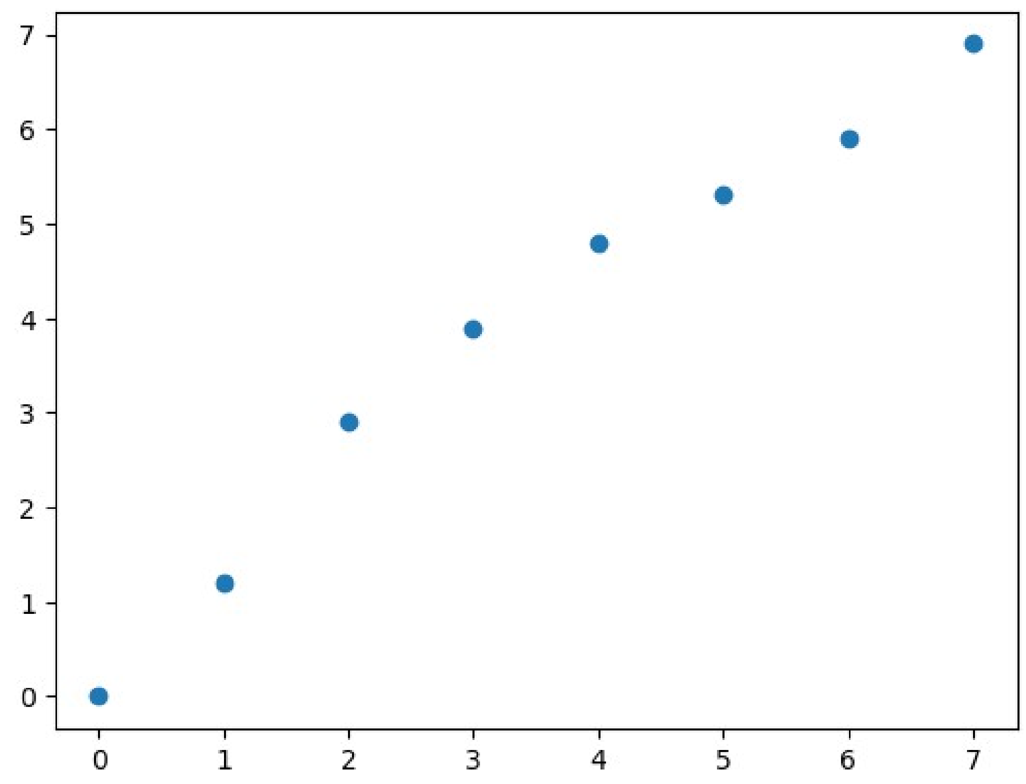
```
#Dependent Variable
y=np.array([0,1.2,2.9,3.9,4.8,5.3,5.9,6.9])
y
```

```
array([0. , 1.2, 2.9, 3.9, 4.8, 5.3, 5.9, 6.9])
```

Visualizing the data

```
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7355ae7ef230>
```



Linear Regression Model

```
#Importing the LinearRegression model
from sklearn.linear_model import LinearRegression
#making an instance of LinearRegression
model=LinearRegression()
#fitting the data to the model
model.fit(x.reshape(-1,1),y)#x should be 2d array while fitting the
data

LinearRegression()
```

Model Parameters (Slope and Intercept)

```
print(f'Slope :{model.coef_[0]}')
print(f'Intercept: {model.intercept_}')

Slope :0.9511904761904761
Intercept: 0.5333333333333332
```

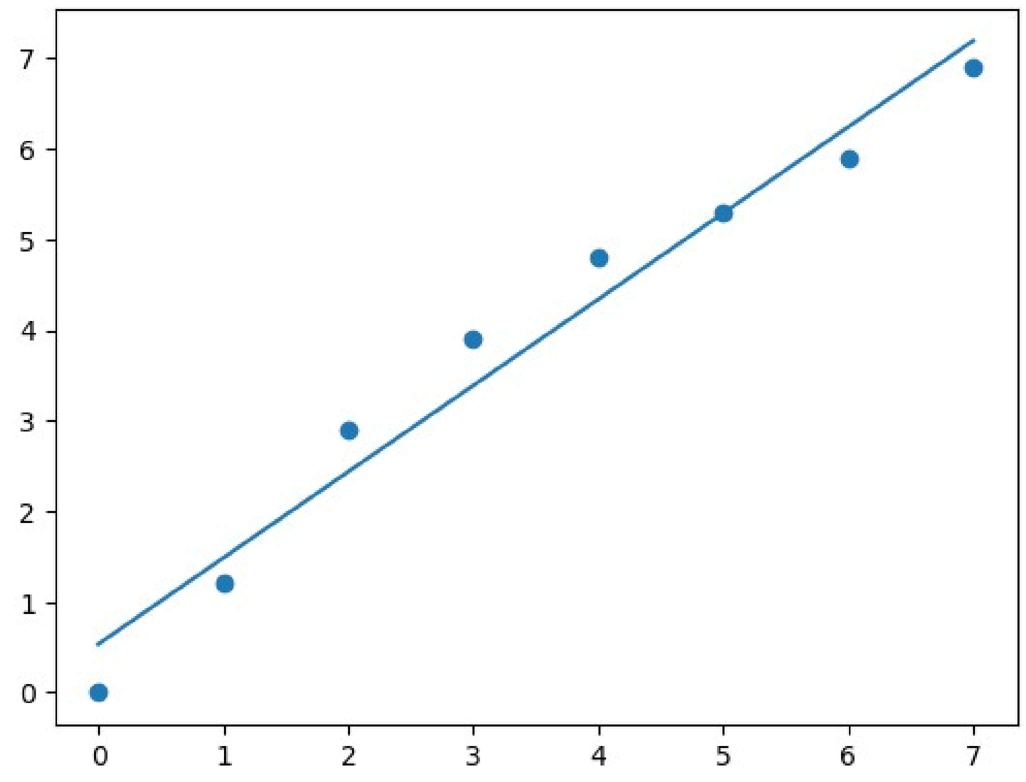
Predicting the Values

```
y_pred=model.predict(x.reshape(-1,1))
# y_pred=model.coef_[0]*x+model.intercept_# y=mx+c equation of
regression line
y_pred
```

```
array([0.53333333, 1.48452381, 2.43571429, 3.38690476, 4.33809524,
       5.28928571, 6.24047619, 7.19166667])
```

Visualizing the model output

```
plt.scatter(x,y)
plt.plot(x,y_pred)

[<matplotlib.lines.Line2D at 0x7355a601c770>]
```



Regression Performance metrics

1. Mean Absolute Error (MAE):

- **Definition**: MAE measures the average magnitude of errors in a set of predictions, without considering their direction. It is the average of the absolute differences between predicted and actual values.
- **Formula**:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- n is the number of observations.

---

2. Mean Squared Error (MSE):

- **Definition**: MSE measures the average of the squares of the errors, giving more weight to larger differences between predicted and actual values.
- **Formula**:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2$$

Where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- n is the number of observations.

---

3. R-squared (R²) Score:

- **Definition**: R², also called the coefficient of determination, represents the proportion of variance in the dependent variable that is predictable from the independent variables. It indicates how well the model explains the variability in the target variable.
- **Formula**:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n} \left(y_i - \acute{y}\right)^2}$$

Where:

- y_i is the actual value,
- $\hat{y}_i$ is the predicted value,
- $\acute{y}$ is the mean of the actual values,
- n is the number of observations.

```
from sklearn.metrics import
mean_absolute_error,mean_squared_error,r2_score
print('Mean Absolute Error :',mean_absolute_error(y,y_pred))
print('Mean Squarred Error :',mean_squared_error(y,y_pred))
print('R2 Value: ',r2_score(y,y_pred))
```

```
Mean Absolute Error : 0.36249999999999998
Mean Squarred Error : 0.14391369047619035
R2 Value:  0.9725878684807256
```

Part B:Implementation of Linear Regression Model on a dataset

```
#pandas library for dataset handling and manipulation
import pandas as pd
```

Import the dataset

```
df=pd.read_csv('archive/Salary_Data.csv')
df.head()
```

```
   YearsExperience  Salary
0              1.1   39343
1              1.3   46205
2              1.5   37731
3              2.0   43525
4              2.2   39891
```

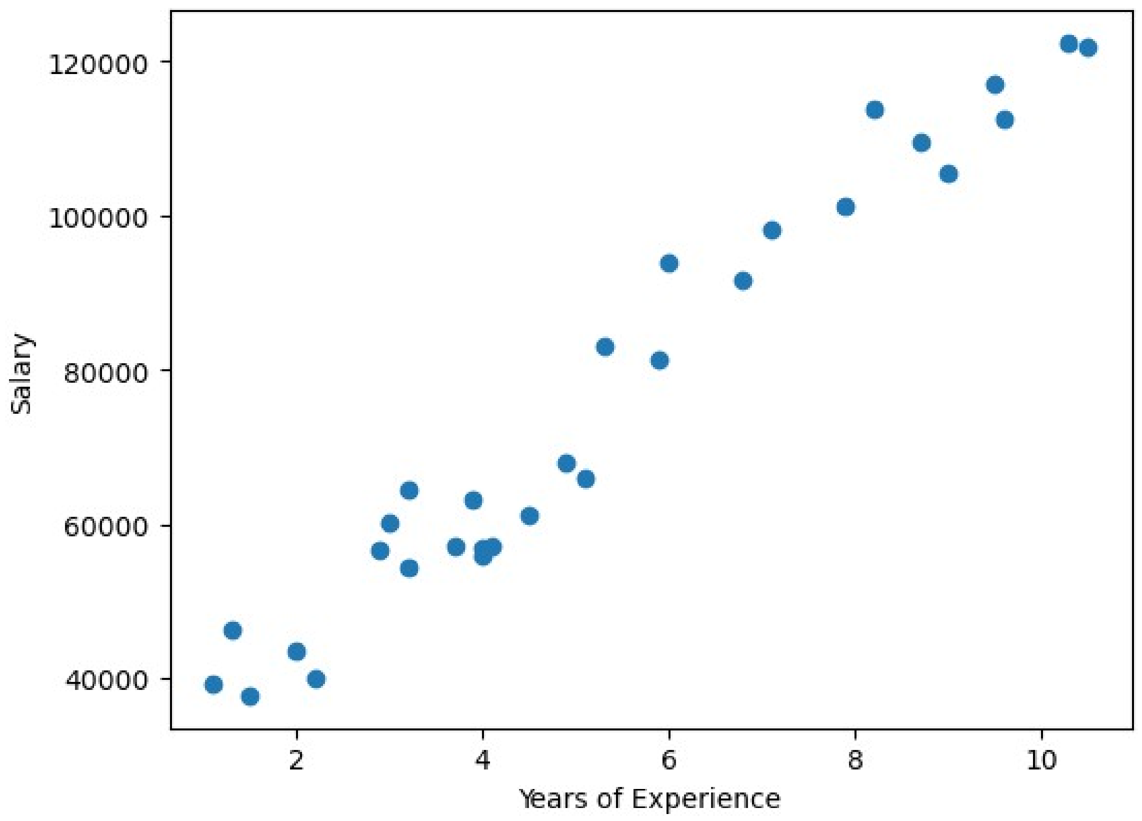
```
df.shape
```

```
(30, 2)
```

Visualizing the data

```
plt.scatter(df['YearsExperience'],df['Salary'])
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
```

```
Text(0, 0.5, 'Salary')
```



Separating data into Independent/Features Variable and Dependent/Target Variable

```
X=np.array(df['YearsExperience'])
Y=df['Salary']
```

Splitting the data into train 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=0.2,rando
m_state=0)
```

`train_test_split(X, Y, test_size=0.2, random_state=0)` splits the dataset `X` (features) and `Y` (labels) into training (80%) and testing (20%) subsets. The `test_size=0.2` determines the proportion of the data for testing, while `random_state=0` ensures reproducibility of the split. It returns four subsets: `X_train`, `X_test`, `Y_train`, and `Y_test`.

Linear Regression Model

```
#Importing the LinearRegression model
from sklearn.linear_model import LinearRegression
#making an instance of LinearRegression
model=LinearRegression()
#fitting the data to the model
model.fit(X_train.reshape(-1,1),Y_train)#X should be 2d array while
fitting the data
```

```
LinearRegression()
```

Model Parameters (Slope and Intercept)

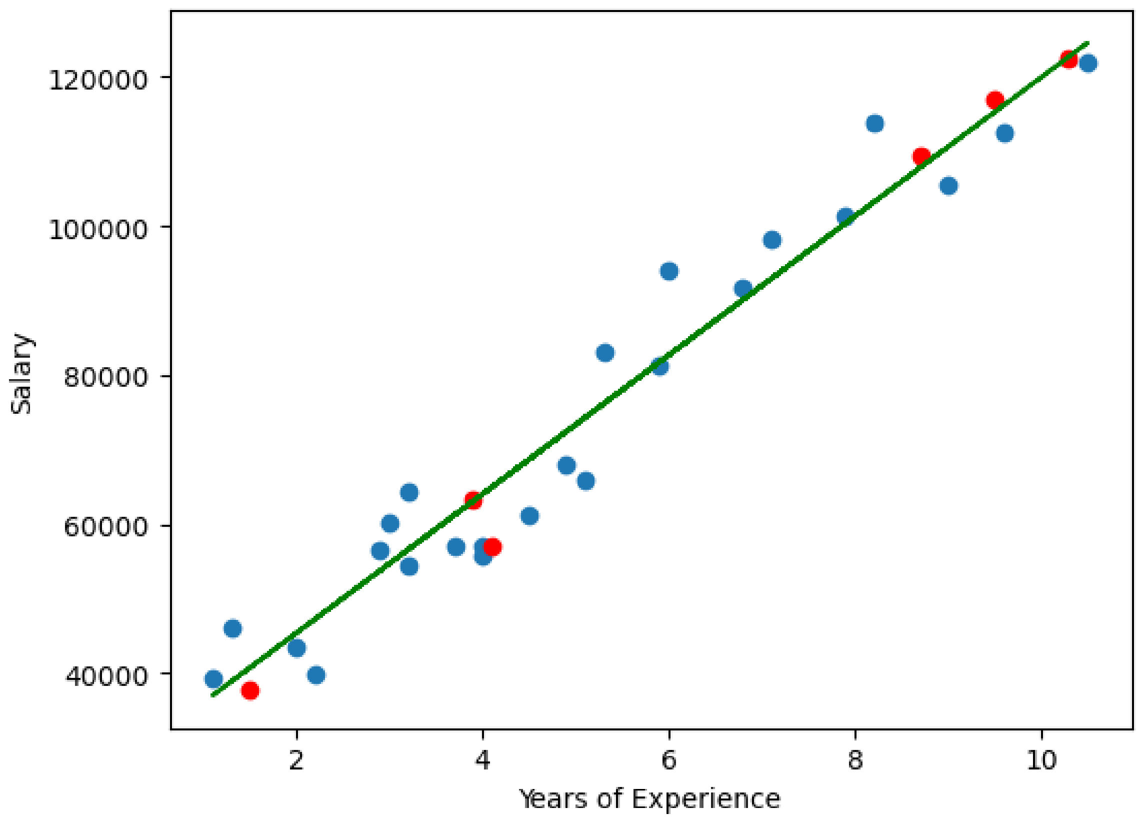
```
print(f'Slope :{model.coef_[0]}')
print(f'Intercept: {model.intercept_}')

Slope :9312.575126729187
Intercept: 26780.099150628186
```

Predicting the Salary

```
Y_pred=model.predict(X_test.reshape(-1,1))
Y_pred

array([ 40748.96184072, 122699.62295594,  64961.65717022,
63099.14214487,
       115249.56285456, 107799.50275317])
```

Visualizing the model

```
model_eq=model.coef_[0]*X_train+model.intercept_

plt.scatter(X_train,Y_train)#training data
plt.scatter(X_test,Y_test,c='red')#test data
plt.plot(X_train,model_eq,c='green')#linear regression model
plt.xlabel('Years of Experience')
plt.ylabel('Salary')

Text(0, 0.5, 'Salary')
```



Regression Model Performance metrics

```
from sklearn.metrics import
mean_absolute_error,mean_squared_error,r2_score
print('Mean Absolute Error :',mean_absolute_error(Y_test,Y_pred))
print('Mean Squarred Error :',mean_squared_error(Y_test,Y_pred))
print('R2 Value: ',r2_score(Y_test,Y_pred))

Mean Absolute Error : 2446.1723690465064
Mean Squarred Error : 12823412.298126562
R2 Value:  0.988169515729126
```

Conclusion:

Merits and Demerits of Linear Regression

Merits:

1. **Simplicity**: Linear regression is easy to understand and interpret, making it a good starting point for regression analysis.
2. **Efficiency**: Computationally efficient, especially for smaller datasets, and works well when the relationship between dependent and independent variables is linear.
3. **Low Variance**: With fewer parameters, it is less likely to overfit, especially in cases with fewer features.

4. **Good interpretability**: The coefficients in linear regression offer clear interpretations of the relationship between variables (i.e., the change in the dependent variable for a unit change in the independent variable).
5. **Widely used**: Suitable for many real-world problems, and often provides good results when assumptions are met.

Demerits:

1. **Assumes Linearity**: Linear regression assumes a linear relationship between the features and target variable, which might not always hold in real-world data.
2. **Sensitive to Outliers**: Outliers can have a significant impact on the regression model, leading to poor predictions.
3. **Multicollinearity**: If the independent variables are highly correlated, the model's coefficients can become unstable and difficult to interpret.
4. **Limited Flexibility**: It is not capable of capturing complex relationships in the data that are non-linear.
5. **Requires Assumptions**: Linear regression assumes normality of errors, homoscedasticity (constant variance of errors), and no multicollinearity, which might not be satisfied in real datasets.