MACHINE

LEARNING

SALARY PREDICTION USING LINEAR REGRESSION

GROUP MEMBERS

DIVYA CHAUHAN

SAKSHI PARIHAR

INTRODUCTION

We have managed to build a simple linear model to predict salary based on years of working experience. Based on our linear model, we can conclude that our salary is grown with our years of working experience and there is a linear relationship between them. We can use our linear model to predict the salary by giving input of years of experience.

Although this is a little too naive to presume the salary is only dependent on the years of working experience, this is one of the great examples to demonstrate how we can develop a simple linear regression model to show a relationship between two variables. In fact, most phenomena in real life cannot be easily explained by a linear model. However, the understanding of building a linear model is the foundation to build a complex model.

THEORY

1. **Loading data:**
   - Import all the required libraries.
   - Use the *Pandas read_csv* function to read the CSV file. This function will return the data in a dataframe format.
   - Extract the column of *YearsExperience* and *Salary* and assign them to the variables X and y, respectively.

2. **Splitting data into a training set and a test set:**
   - Set apart 30% of the entire dataset as the test set and assign the training set and test set into four variables, respectively.

3. **Data Transformation:**
   - Use *Numpy reshape function* to transform the training set from 1-dimensional series to a 2-dimensional array.
   - Use *Numpy reshape function* to transform the test set from 1-dimensional series to a 2-dimensional array.

4. **Training Model:**
   - Use the *Scikit-Learn LinearRegression* function to create a model object.
   - Fit the training set to the model.

5. **Predicting Salary using Linear Model**
6. **Model Evaluation:**
   - We will use three types of quantitative metrics:

- Mean Square Error
- Explained Variance Score
- R2 Score

```
# import all the lib
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

[8] ✓ 0.3s

```
# read the dataset using pandas
data = pd.read_csv('C:\\Users\ASUS\\Downloads\\Salary_Data.csv')
```

[9] ✓ 0.4s

```
# This displays the top 5 rows of the data
data.head()
```

[10] ✓ 0.1s

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
# Provides some information regarding the columns in the data
data.info()
```

[11] ✓ 0.9s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

```
# this describes the basic stat behind the dataset used
data.describe()
```

[12] ✓ 0.8s

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000

```
memory usage: 608.0 bytes
```

```
# this describes the basic stat behind the dataset used
data.describe()
```

[12] ✓ 0.8s

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

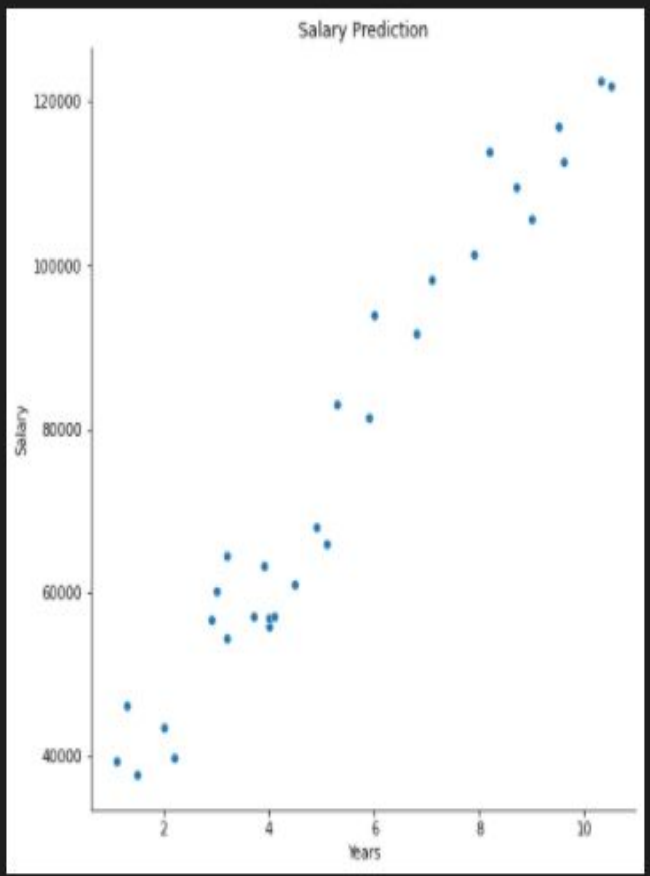
```
# These Plots help to explain the values and how they are scattered

plt.figure(figsize=(12,6))
sns.pairplot(data,x_vars=['YearsExperience'],y_vars=['Salary'],size=7,kind='scatter')
plt.xlabel('Years')
plt.ylabel('Salary')
plt.title('Salary Prediction')
plt.show()
```

[13] ✓ 0.5s

```
c:\Users\ASUS\anaconda3\lib\site-packages\seaborn\axisgrid.py:2076: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)

<Figure size 864x432 with 0 Axes>
```



SALARY_PREDICT.ipynb • | Salary_Data.csv | 1_qTDqjFtJ5dexi6_fIshpGA.jpeg

SALARY_PREDICT.ipynb > X = data['YearsExperience']

+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ... | base (Python 3.9.12)

```python
X = data['YearsExperience']
X.head()
```

[25] ✓ 0.4s Python

```
0    1.1
1    1.3
2    1.5
3    2.0
4    2.2
Name: YearsExperience, dtype: float64
```

```python
y = data['Salary']
y.head()
```

[26] ✓ 0.4s Python

```
0    39343.0
1    46205.0
2    37731.0
3    43525.0
4    39891.0
Name: Salary, dtype: float64
```

```python
# Import Segregating data from scikit learn
from sklearn.model_selection import train_test_split
# the data for train and test
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.7,random_state=100)
```

[16] ✓ 0.2s Python

```python
# Create new axis for x column
X_train = X_train[:,np.newaxis]
X_test = X_test[:,np.newaxis]
# Importing Linear Regression model from scikit learn
from sklearn.linear_model import LinearRegression
# Fitting the model
lr = LinearRegression()
lr.fit(X_train,y_train)
```

[17] ✓ 0.6s Python

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_19140\1482792533.py:2: FutureWarning: Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a future version.  Convert to a numpy array before indexing instead.
  X_train = X_train[:,np.newaxis]
C:\Users\ASUS\AppData\Local\Temp\ipykernel_19140\1482792533.py:3: FutureWarning: Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a future version.  Convert to a numpy array before indexing instead.
  X_test = X_test[:,np.newaxis]

LinearRegression()
```

```
LinearRegression()
```

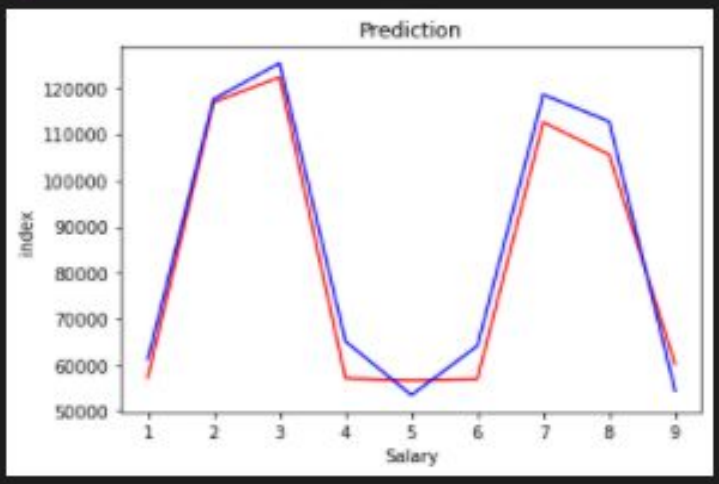
```
# Predicting the Salary for the Test values
y_pred = lr.predict(X_test)
```

[18] ✓ 0.7s

```
# Plotting the actual and predicted values

c = [i for i in range (1,len(y_test)+1,1)]
plt.plot(c,y_test,color='r',linestyle='-')
plt.plot(c,y_pred,color='b',linestyle='-')
plt.xlabel('Salary')
plt.ylabel('index')
plt.title('Prediction')
plt.show()
```

[19] ✓ 0.2s




```
# plotting the error
c = [i for i in range(1,len(y_test)+1,1)]
plt.plot(c,y_test-y_pred,color='green',linestyle='-')
plt.xlabel('index')
plt.ylabel('Error')
plt.title('Error Value')
plt.show()
```

[20] ✓ 0.2s

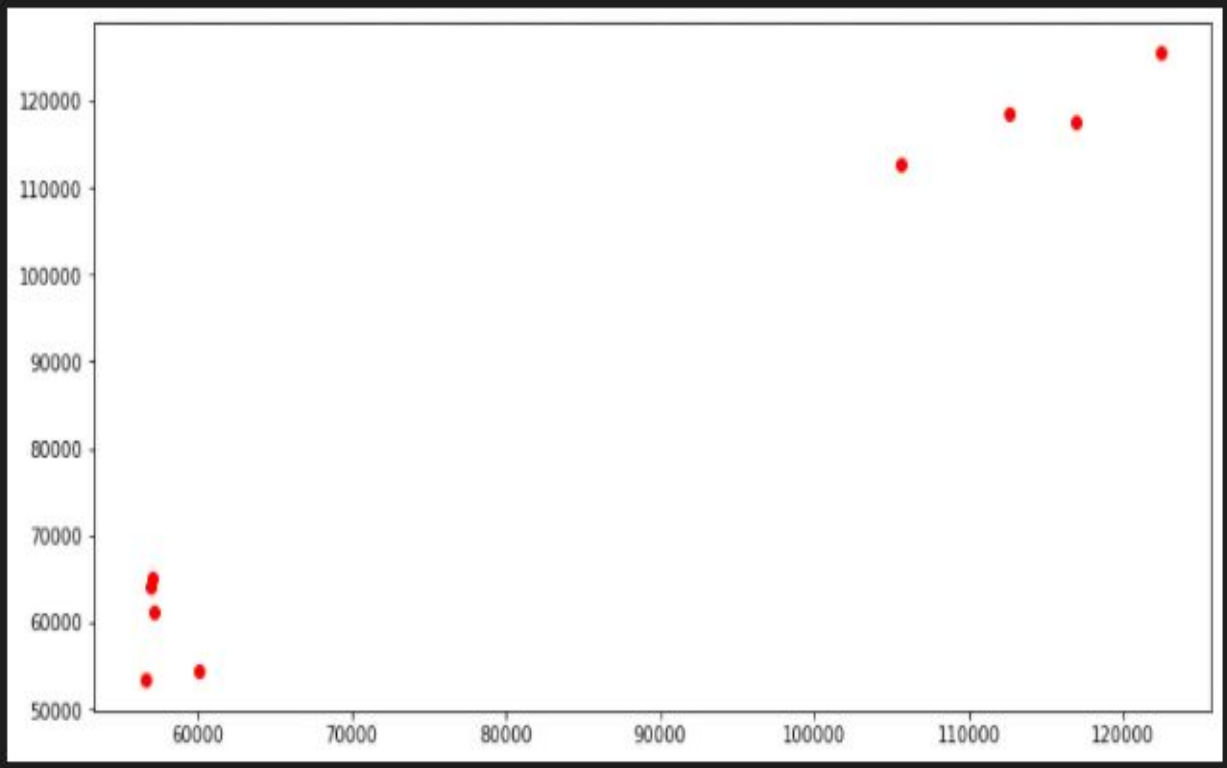
```
# Importing metrics for the evaluation of the model
from sklearn.metrics import r2_score,mean_squared_error
# calculate Mean square error
mse = mean_squared_error(y_test,y_pred)
# Calculate R square vale
rsq = r2_score(y_test,y_pred)
print('mean squared error :',mse)
print('r square :',rsq)
```

[21] ✓ 0.9s

```
mean squared error : 30310299.043402452
r square : 0.9627668685473267
```

```
# Just plot actual and predicted values for more insights
plt.figure(figsize=(12,6))
plt.scatter(y_test,y_pred,color='r',linestyle='-')
plt.show()
```

[22] ✓ 0.2s




```
# Intecept and coeff of the line
print('Intercept of the model:',lr.intercept_)
print('Coefficient of the line:',lr.coef_)
```

[23] ✓ 0.6s

```
Intercept of the model: 25202.887786154883
Coefficient of the line: [9731.20383825]
```

TECH USED

1)PYTHON
2)SK LEARN
3)NUMPY
4)PANDAS
5)SEABORN
6)KAGGLE
7)GOOGLE

SALARY PREDICTION

BIBLIOGRAPHY