Simple Linear Regressions

Equation:

$$y = b_0 + b_1 X_1$$

Where:

- `y` is the dependent variable
- b_0 is the y-intercept
- b_1 is the slope coefficient
- X_1 is the independent

To understand this concept with an example.
- Lets say we want to predict how the yield of a crop based on the amount of fertilizer(in kilograms) is added.
- In this case, we will have our equation setup like such:

$$CropYield[y] = b_0 + b_1 Fertilizer[kg]$$

- Lets say the farmer has data throughout the season on how much crop yielded based on the amount of fertilizer used. Plotting this data gives us a scatter plot. How do we apply our linear regression to this data which will accurately predict how the yield from amount of fertilizer.
- The challenge is, from the scatterplot, we have a lot of options for the slop line. There can be multiple slope lines which go through our data points, but to determine the best one we use a method called **Ordinary Least Squares**

Ordinary Least Squares

- This method allows us to choose the best possible linear regression line.

How does it work in theory?

- Every data point is projected onto the line, where:
  y_i is the actual data. In our example, it would be the actual yield the farmer has for the amount of fertilizer used.
  $\hat{y_i}$ is the projected value. In our example, it would the yield predicted by the linear regression based on the fertilizer used.

There will be a difference between the two values and that is completely normal. The goal is to find a line where every data point has the lowest difference between y_i and $\hat{y}_i$.

- The difference between y_i and $\hat{y}_i$ is called the `residual`, where $\epsilon_i = y_i - \hat{y}_i$
- The best equation
  $y = b_0 + b_1 X_1$
  b_0 , b_1 such that:
  SUM $(y_i - \hat{y}_i)^2$ is minimized

  So for every data point, find the residual, square the value, and finally add all the squared values for each data point together. The iteration with the lowest total sum, is the best linear regression line.

Applying Simple Linear Regressions

- As a necessary first step, we must first pre-process the data.
  - **Import Libraries**

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

  - **Import Dataset**

```
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

  - **Splitting Dataset into Training and Testing Sets**

```
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 Data on our Linear Regression Model**
  Now we have to call our Linear Regression model on the Training Set

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
        ```

To train our training sets to this model, we use the `.fit()` function. The
parameters required are the `X_train` (which consists the features of the dataset),
```
```

```
and `y_train` contains the dependent variable training set.

- **Predicting Data using Model**
Getting predictions using our model which was training on the training dataset is
fairly simple.
```python
y_pred = regressor.predict(X_test)
```

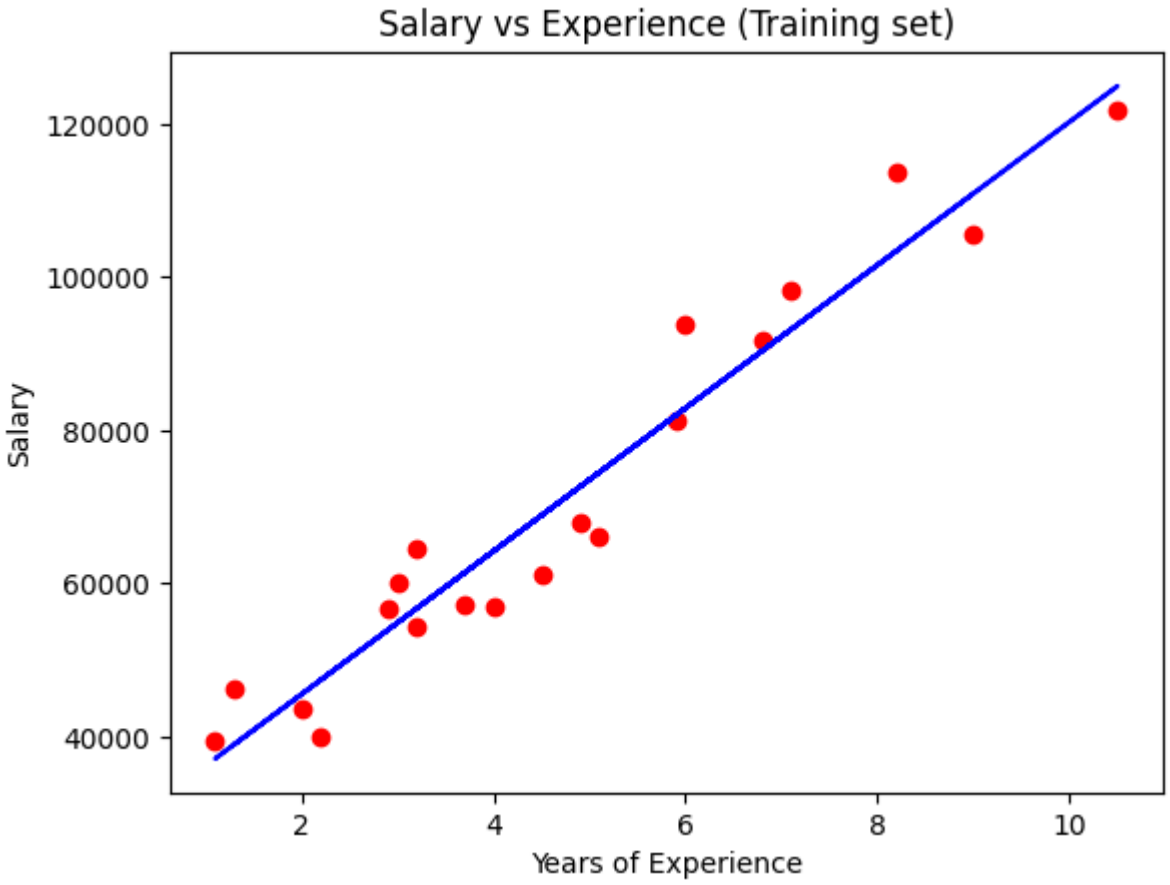
We use the `predict()` method to get the predictions for the dataset, need to pass in test dataset, `X_test`.

- **Visualizing Predictions**
  Lets start off by taking a look at our training set result.
  We can now use the `matplotlib` to generate charts to view our predictions

```
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()
```

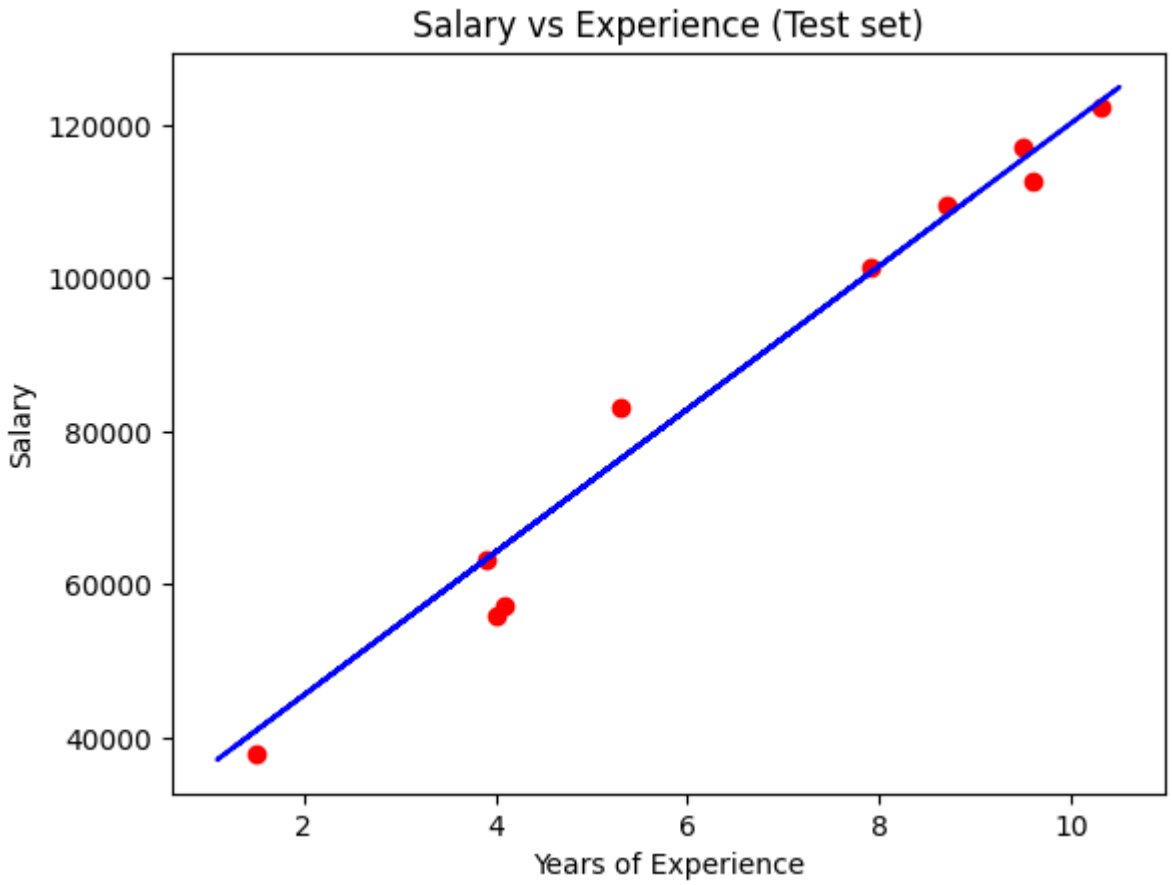
Where we create the scatter plot, using the `.scatter()` where parameters is `X-axis`, `Y-axis`, and the color of scattered plots
We than use the `.plot()`, to display the predicted values. In this case, our X-Coordinates are the same as from training set, but the Y-Coordinates are the predicted values for given x-coordinate. So, we pass in the parameters as `X_train`, `regressor.predict(X_train)`, `color`.
We can define the `title` of the graph, `xlabel` and `ylabel` and display it. Viewing this graph below, we see the best possible line of fit given our dataset, this is predicted by our linear
```

regression model.



To view our test set result:
Where we switch out the scatter plot with the data with the test set(also treated as real data). We do not need to change our datasets in the regression model as it is generating the line of best fit (predictions) based on the training and is unique.

```
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()
```



To get a prediction based on an input:

```
print(regressor.predict([[12]]))
```

To get the b_0, the y-intercept & b_1, the slope coefficient for our models:

```
print(regressor.coef_) #B_1
print(regressor.intercept_)#B_0
```

Note we observe clean and accurate results in the above graphs as the dataset is linear in relation. This is not the case for all datasets. Use Models carefully.

Next Steps -> Regression Topics

Multiple Linear Regression