LAB 3A : Implement Simple Linear regression

Import Libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

import dataset

In [4]:
```
dataset = pd.read_csv("delivery_time_data.csv")
```

EDA Steps

In [5]:
```
dataset.head()
```

Out[5]:

	Distance	Delivery Time
0	11.55	75.32
1	28.55	181.90
2	22.09	140.41
3	18.16	112.23
4	5.10	32.92

In [6]:
```
dataset.shape
```

Out[6]:
```
(500, 2)
```

In [7]:
```
dataset.columns
```

Out[7]:
```
Index(['Distance', 'Delivery Time'], dtype='object')
```

In [8]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Distance       500 non-null    float64
 1   Delivery Time  500 non-null    float64
dtypes: float64(2)
memory usage: 7.9 KB
```

In [9]:
```
dataset.describe()
```

Out[9]:

	Distance	Delivery Time
count	500.000000	500.000000
mean	15.207480	96.268320
std	8.811485	53.159548
min	0.650000	5.660000
25%	7.615000	50.012500
50%	15.640000	98.490000
75%	22.807500	142.622500
max	29.790000	188.100000

Preprocessing Steps

In [10]:
```
# Step 1 : Seprate i/p Independent Var and
# o/p Dependent Var
X = dataset.iloc[:,:-1]
y = dataset.iloc[:,-1]
```

In [11]:
```
print(X.head())
```

```
   Distance
0     11.55
1     28.55
2     22.09
3     18.16
4      5.10
```

In [12]:
```
print(y.head())
```

```
0     75.32
1    181.90
2    140.41
3    112.23
4     32.92
Name: Delivery Time, dtype: float64
```

In [13]:
```
# Step 4 : Split data into training and testin
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split (X,y,
                                        test_size = 0.3,
                                        random_state = 0)
```

In [14]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(350, 1)
(150, 1)
```

Create the Regression model on training data

In [16]:
```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X_train,y_train)
```

Out[16]:

▾ LinearRegression

LinearRegression()

In [17]:
```
print("Intercept B0 = ",regressor.intercept_)
print("Coefficient B1= ",regressor.coef_)
```

```
Intercept B0 =  4.519663334710799
Coefficient B1=  [6.03556347]
```

From Above values our model regression line equation

y = 4.519663334710799 + 6.03556347 * X

By using this equation lets find Time for Distance =6

In [18]:
```
ynew = 4.519663334710799 + 6.03556347 * 6

print("Time for Distance 6 = ",ynew)
```

```
Time for Distance 6 =  40.7330441547108
```

In [19]:
```
ypred = regressor.predict(X_test)
```

In [20]:
```
print(ypred)
print(y_test)
```

```
[ 28.84298413  37.71526244  24.25595589 176.17108853 123.11848559
  40.18984346  80.32634056 166.03134189 137.18134849  94.69098163
 177.37820122  59.98649166 183.89660977 112.73731642  10.49487117
  51.65741406  47.372164   121.9113729  183.05163089  24.91986788
   9.16704721 178.70602518 174.96397583  48.27749852 125.47235535
 133.25823223  32.40396658 168.74734545 100.60583384  95.23418234
  70.00552702 166.45383133  48.09643161 167.29881022  20.21212837
  56.90835429  21.4795967  106.94317548  36.56850538 122.45457361
 125.11022154  63.06462903 154.68448256 108.75384452 144.8465141
 176.83500051  47.49287527 138.93166189 127.28302439  63.48711847
 104.710017   128.18835891  84.79265753  58.23617825  28.90333977
 155.77088398 116.47936577  47.85500908  54.73555144  38.37917442
  71.33335099 149.67496488  39.10344204 105.37392898 102.1147247
 136.81921468  11.21913879 179.06815899 151.78741209 137.36241539
  96.07916123 104.16681628  95.05311544 111.59055936  38.56024133
  89.13826324 163.73782777  13.1505191  117.927901   100.5454782
  16.71150155 136.15530269 184.31909921 158.72831009  70.91086154
 106.21890787 148.10571837 143.15655632  74.23042146 114.00478475
 124.62737646  97.46734083  57.69297754 121.85101726  32.04183277
  24.37666716  23.29026574  22.62635376  76.76535811   8.44277959
 142.0701549  159.87506715  42.90584703 101.51116836  17.85825861
 176.77464487 151.42527828  57.51191063  22.74706503  50.63136827
 161.92715873 161.74609182 132.71503151 114.54798546 111.40949245
 117.62612283  64.33209736 110.50415793 105.25321771 101.81294653
  33.00752293  40.431266   175.44682091  76.28251304 109.65917905
  39.0430864   65.47885442  76.82571375  80.99025255 176.71428924
 167.9627222  119.91963695  45.9839844  171.70477155  71.6954848
  88.65541816  17.55648044  91.00928791 168.92841236 153.05488042
  67.89307981 174.84326456 133.3789435   43.08691393 176.77464487
  35.30103705  35.36139269 142.49264434 172.85152861  35.30103705]
90      30.90
254     36.99
283     27.91
445    172.91
461    130.28
        ...
4       32.92
318     31.02
331    142.59
245    176.90
5       37.09
Name: Delivery Time, Length: 150, dtype: float64
```

Accuracy of model

In [21]:
```
# 1. Training Accuracy

print("Training accuracy =",
      regressor.score(X_train,y_train))
```

```
Training accuracy = 0.99678490852808
```

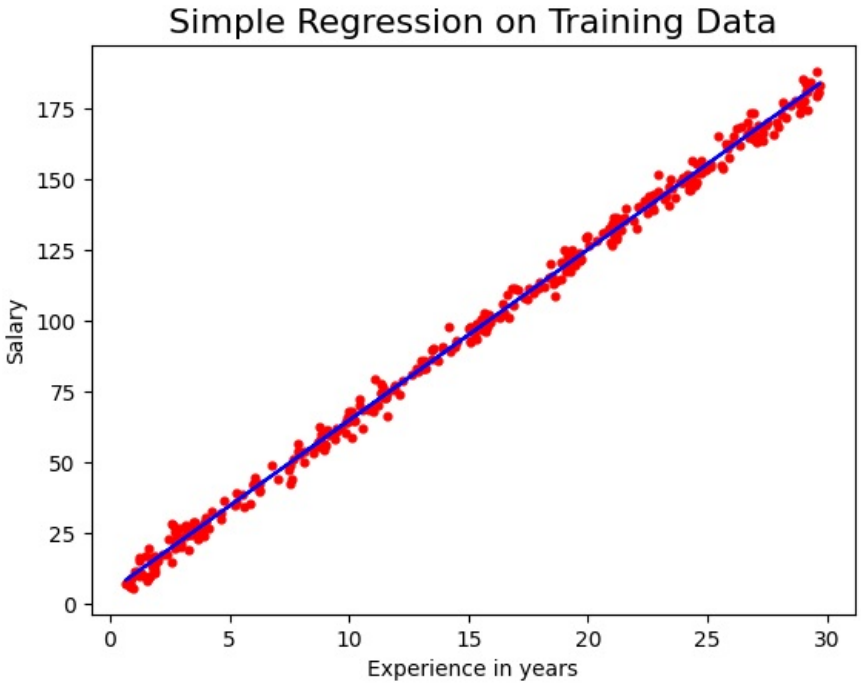
In [22]:
```
# 2. Testing Accuracy

print("Testing accuracy =",
      regressor.score(X_test,y_test))
```

```
Testing accuracy = 0.99676990148807
```

Visualizing the Model

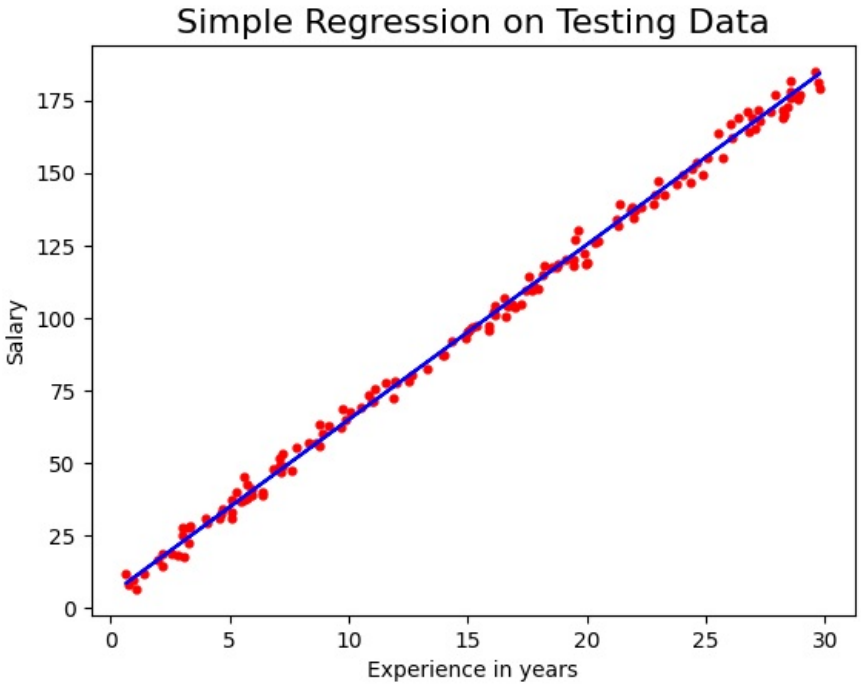
1. Visualize the training data

In [23]:
```
plt.scatter(X_train,y_train,color = 'red', s= 12)
plt.plot(X_train, regressor.predict(X_train),
        color = 'blue')
plt.title("Simple Regression on Training Data", size= 16)
plt.xlabel("Experience in years")
plt.ylabel("Salary")
plt.show()
```



2. Visualize the testing data

In [24]:
```
plt.scatter(X_test,y_test,color = 'red', s= 12)
plt.plot(X_test, regressor.predict(X_test),
        color = 'blue')
plt.title("Simple Regression on Testing Data", size= 16)
plt.xlabel("Experience in years")
plt.ylabel("Salary")
plt.show()
```



Find R^2 Score of model

In [25]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[25]:
```
0.99676990148807
```

In [27]:
```
from sklearn.metrics import median_absolute_error
median_absolute_error(y_train, regressor.predict(X_train))
```

Out[27]:
```
np.float64(2.017053470150735)
```

median_absolute_error(y_test, regressor.predict(X_test))

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js