```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
%matplotlib inline
```

```
# Read the data from the Excel file into a DataFrame
data = pd.read_excel("/content/drive/MyDrive/dataset.xlsx")

# Display the first few rows of the DataFrame
print(data.head())
```

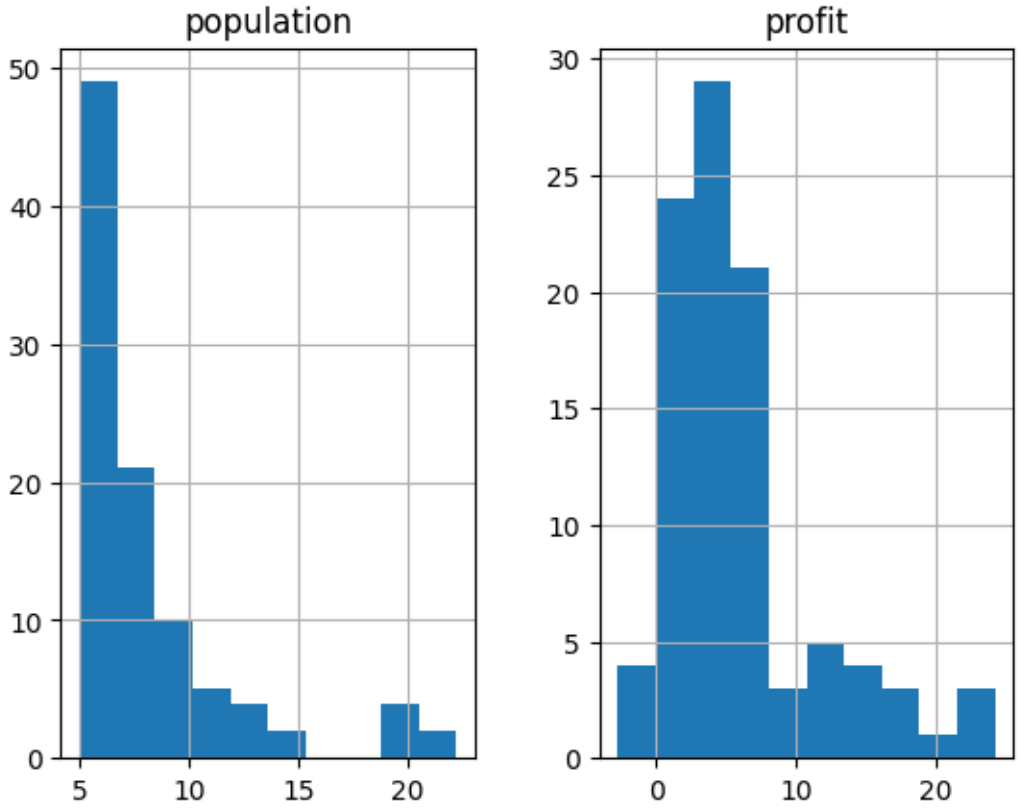
```
   population   profit
0      6.1101  17.5920
1      5.5277   9.1302
2      8.5186  13.6620
3      7.0032  11.8540
4      5.8598   6.8233
```

```
data.describe()
```

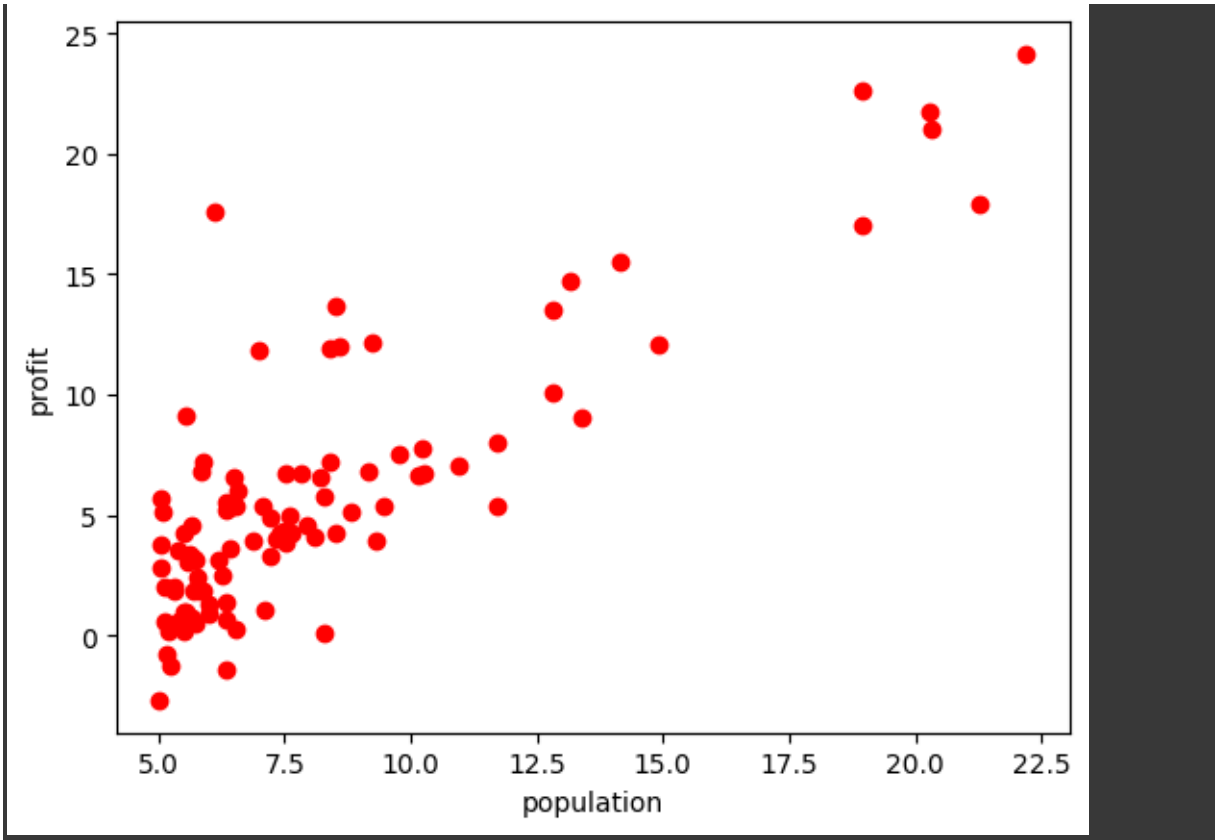
	population	profit
count	97.000000	97.000000
mean	8.159800	5.839135
std	3.869884	5.510262
min	5.026900	-2.680700
25%	5.707700	1.986900
50%	6.589400	4.562300
75%	8.578100	7.046700
max	22.203000	24.147000

```
cdata=data[["population","profit"]]
```

```
cd=cdata[["population","profit"]]
cd.hist()
```




```
plt.scatter(cd.population,cd.profit,color='r')
plt.xlabel("population")
plt.ylabel("profit")
plt.show()
```



Creating train and test dataset Train/Test Split involves splitting the dataset into training and testing sets that are mutually exclusive. After which, you train with the training set and test with the testing set. This will provide a more accurate evaluation on out-of-sample accuracy because the testing dataset is not part of the dataset that have been used to train the model. Therefore, it gives us a better understanding of how well our model generalizes on new data.

This means that we know the outcome of each data point in the testing dataset, making it great to test with! Since this data has not been used to train the model, the model has no knowledge of the outcome of these data points. So, in essence, it is truly an out-of-sample testing.

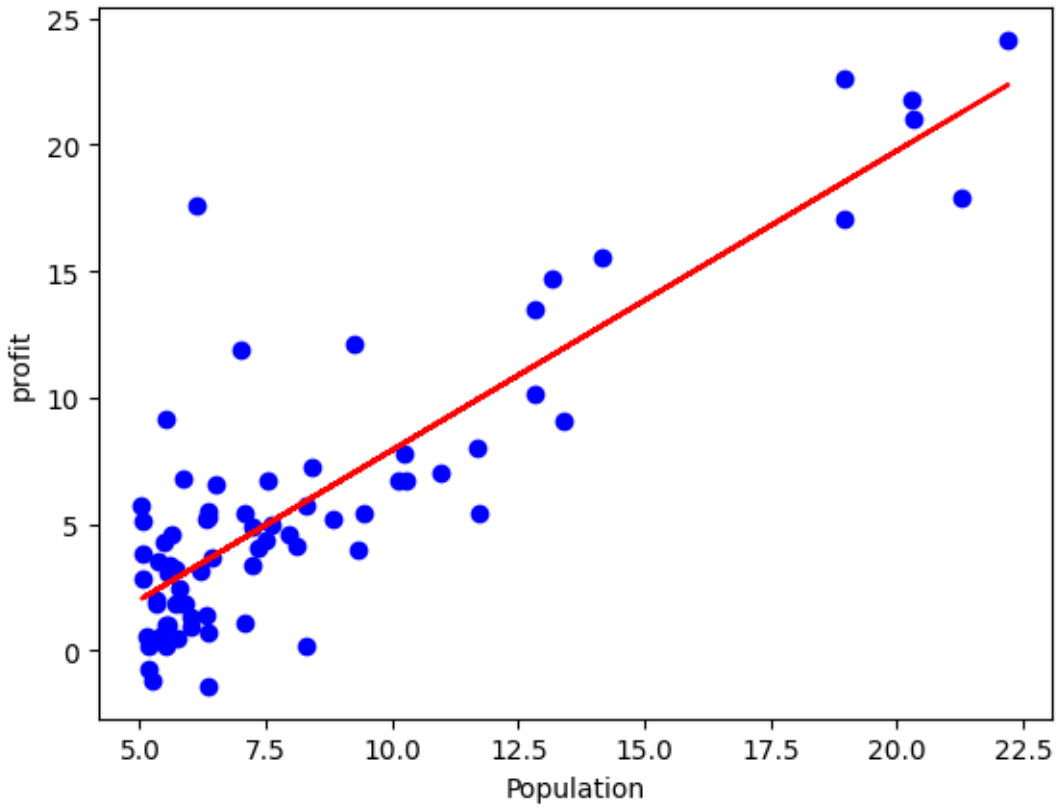
Let's split our dataset into train and test sets. 80% of the entire dataset will be used for training and 20% for testing. We create a mask to select random rows using np.random.rand() function:

```
msk=np.random.rand(len(data))<0.8
train=cd[msk]
test=cd[~msk]
```

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['population']])
```

```
train_y = np.asanyarray(train[['profit']])
regr.fit(train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
plt.scatter(train.population, train.profit,  color='blue')
#plt.scatter(data.population, data.profit,  color='yellow')
#plt.plot(data.population,data.profit,color="y")
plt.plot(train_x, regr.coef_[0][0]*train_x + regr.intercept_[0], '-r')
plt.xlabel("Population")
plt.ylabel("profit")
```




```
from sklearn.metrics import r2_score

test_x = np.asanyarray(test[['population']])
test_y = np.asanyarray(test[['profit']])
test_y_ = regr.predict(test_x)
```

```
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ -
test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ -
test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y , test_y_) )
```

```
Mean absolute error: 2.15
Residual sum of squares (MSE): 8.84
R2-score: 0.47
```