Experiment No -04

Simple Linear Regression Model

In [1]:
```
print('-------------------------------EXPERIEMENT-04-------------------
---------------')
print('NAME:  Pratyush Srivastava')
print('ROLL NO: 18SCSE1010128')
```

```
-------------------------------EXPERIEMENT-04-------------------------
--------
NAME:  Pratyush Srivastava
ROLL NO: 18SCSE1010128
```

Ordiniary Least Squares

In [2]:
```
#Importing Linear Regression Model From Sklearn Library
from sklearn import linear_model
reg = linear_model.LinearRegression()
```

In [3]:
```
#Training of raw data that is passes in fit function
reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
```

Out[3]:
```
LinearRegression()
```

In [4]:
```
#Coefficient of the Linear Regression Model
reg.coef_
```

Out[4]:
```
array([0.5, 0.5])
```

Ridge Regression

In [5]:
```
#Importing Linear Regression Model From Sklearn Library
from sklearn import linear_model
reg = linear_model.Ridge(alpha=.5)
```

In [6]:
```
#Training of raw data that is passes in fit function
reg.fit([[0, 0], [0, 0], [1, 1]], [0, .1, 1])
```

Out[6]:
```
Ridge(alpha=0.5)
```

In [7]:
```
#Coefficient of the Linear Regression Model
reg.coef_
```

Out[7]:
```
array([0.34545455, 0.34545455])
```

In [8]:
```
#Intercept of the Linear Regression Model
reg.intercept_
```

Out[8]:
```
0.13636363636363638
```

First Example of Ordinary Least Sqaure(OLS)

In [9]:
```
#Importing Numpy and Linear Regression Model
import numpy as np
from sklearn.linear_model import LinearRegression
```

In [10]:
```
#Taking X as raw input
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
# y = 1 * x_0 + 2 * x_1 + 3
```

In [11]:
```
#Taking y as output and after that train the model by using fit functio
n
```

```
y = np.dot(X, np.array([1, 2])) + 3
reg = LinearRegression().fit(X, y)
```

In [12]:
```
#Check the accuray of the model
reg.score(X, y)
```

Out[12]:
```
1.0
```

In [13]:
```
#Coefficient Of the model
reg.coef_
```

Out[13]:
```
array([1., 2.])
```

In [14]:
```
#Intercept of the Model
reg.intercept_
```

Out[14]:
```
3.0000000000000018
```

In [15]:
```
#Prediction on unseen data
reg.predict(np.array([[3, 5]]))
```

Out[15]:
```
array([16.])
```

Linear Regression Example with in-built Diabetes dataset

In [16]:
```
#Importing Libraries
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

In [17]:
```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
```

In [18]:
```
# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]
diabetes_X
```

Out[18]:
```
array([[ 0.06169621],
       [-0.05147406],
       [ 0.04445121],
       [-0.01159501],
       [-0.03638469],
       [-0.04069594],
       [-0.04716281],
       [-0.00189471],
       [ 0.06169621],
       [ 0.03906215],
       [-0.08380842],
       [ 0.01750591],
       [-0.02884001],
       [-0.00189471],
       [-0.02560657],
       [-0.01806189],
       [ 0.04229559],
       [ 0.01211685],
       [-0.0105172 ],
       [-0.01806189],
       [-0.05686312],
       [-0.02237314],
       [-0.00405033],
       [ 0.06061839],
       [ 0.03582872],
       [-0.01267283],
       [-0.07734155],
       [ 0.05954058],
       [-0.02129532],
       [-0.00620595],
       [ 0.04445121],
       [-0.06548562],
       [ 0.12528712],
       [-0.05039625],
       [-0.06332999],
       [-0.03099563],
```

```
[ 0.02289497],
[ 0.01103904],
[ 0.07139652],
[ 0.01427248],
[-0.00836158],
[-0.06764124],
[-0.0105172 ],
[-0.02345095],
[ 0.06816308],
[-0.03530688],
[-0.01159501],
[-0.0730303 ],
[-0.04177375],
[ 0.01427248],
[-0.00728377],
[ 0.0164281 ],
[-0.00943939],
[-0.01590626],
[ 0.0250506 ],
[-0.04931844],
[ 0.04121778],
[-0.06332999],
[-0.06440781],
[-0.02560657],
[-0.00405033],
[ 0.00457217],
[-0.00728377],
[-0.0374625 ],
[-0.02560657],
[-0.02452876],
[-0.01806189],
[-0.01482845],
[-0.02991782],
[-0.046085  ],
[-0.06979687],
[ 0.03367309],
[-0.00405033],
[-0.02021751],
[ 0.00241654],
```

```
[-0.03099563],
[ 0.02828403],
[-0.03638469],
[-0.05794093],
[-0.0374625 ],
[ 0.01211685],
[-0.02237314],
[-0.03530688],
[ 0.00996123],
[-0.03961813],
[ 0.07139652],
[-0.07518593],
[-0.00620595],
[-0.04069594],
[-0.04824063],
[-0.02560657],
[ 0.0519959 ],
[ 0.00457217],
[-0.06440781],
[-0.01698407],
[-0.05794093],
[ 0.00996123],
[ 0.08864151],
[-0.00512814],
[-0.06440781],
[ 0.01750591],
[-0.04500719],
[ 0.02828403],
[ 0.04121778],
[ 0.06492964],
[-0.03207344],
[-0.07626374],
[ 0.04984027],
[ 0.04552903],
[-0.00943939],
[-0.03207344],
[ 0.00457217],
[ 0.02073935],
[ 0.01427248],
```

```
[ 0.11019775],
[ 0.00133873],
[ 0.05846277],
[-0.02129532],
[-0.0105172 ],
[-0.04716281],
[ 0.00457217],
[ 0.01750591],
[ 0.08109682],
[ 0.0347509 ],
[ 0.02397278],
[-0.00836158],
[-0.06117437],
[-0.00189471],
[-0.06225218],
[ 0.0164281 ],
[ 0.09618619],
[-0.06979687],
[-0.02129532],
[-0.05362969],
[ 0.0433734 ],
[ 0.05630715],
[-0.0816528 ],
[ 0.04984027],
[ 0.11127556],
[ 0.06169621],
[ 0.01427248],
[ 0.04768465],
[ 0.01211685],
[ 0.00564998],
[ 0.04660684],
[ 0.12852056],
[ 0.05954058],
[ 0.09295276],
[ 0.01535029],
[-0.00512814],
[ 0.0703187 ],
[-0.00405033],
[-0.00081689],
```

```
[-0.04392938],
[ 0.02073935],
[ 0.06061839],
[-0.0105172 ],
[-0.03315126],
[-0.06548562],
[ 0.0433734 ],
[-0.06225218],
[ 0.06385183],
[ 0.03043966],
[ 0.07247433],
[-0.0191397 ],
[-0.06656343],
[-0.06009656],
[ 0.06924089],
[ 0.05954058],
[-0.02668438],
[-0.02021751],
[-0.046085  ],
[ 0.07139652],
[-0.07949718],
[ 0.00996123],
[-0.03854032],
[ 0.01966154],
[ 0.02720622],
[-0.00836158],
[-0.01590626],
[ 0.00457217],
[-0.04285156],
[ 0.00564998],
[-0.03530688],
[ 0.02397278],
[-0.01806189],
[ 0.04229559],
[-0.0547075 ],
[-0.00297252],
[-0.06656343],
[-0.01267283],
[-0.04177375],
```

```
[-0.03099563],
[-0.00512814],
[-0.05901875],
[ 0.0250506 ],
[-0.046085  ],
[ 0.00349435],
[ 0.05415152],
[-0.04500719],
[-0.05794093],
[-0.05578531],
[ 0.00133873],
[ 0.03043966],
[ 0.00672779],
[ 0.04660684],
[ 0.02612841],
[ 0.04552903],
[ 0.04013997],
[-0.01806189],
[ 0.01427248],
[ 0.03690653],
[ 0.00349435],
[-0.07087468],
[-0.03315126],
[ 0.09403057],
[ 0.03582872],
[ 0.03151747],
[-0.06548562],
[-0.04177375],
[-0.03961813],
[-0.03854032],
[-0.02560657],
[-0.02345095],
[-0.06656343],
[ 0.03259528],
[-0.046085  ],
[-0.02991782],
[-0.01267283],
[-0.01590626],
[ 0.07139652],
```

```
[-0.03099563],
[ 0.00026092],
[ 0.03690653],
[ 0.03906215],
[-0.01482845],
[ 0.00672779],
[-0.06871905],
[-0.00943939],
[ 0.01966154],
[ 0.07462995],
[-0.00836158],
[-0.02345095],
[-0.046085  ],
[ 0.05415152],
[-0.03530688],
[-0.03207344],
[-0.0816528 ],
[ 0.04768465],
[ 0.06061839],
[ 0.05630715],
[ 0.09834182],
[ 0.05954058],
[ 0.03367309],
[ 0.05630715],
[-0.06548562],
[ 0.16085492],
[-0.05578531],
[-0.02452876],
[-0.03638469],
[-0.00836158],
[-0.04177375],
[ 0.12744274],
[-0.07734155],
[ 0.02828403],
[-0.02560657],
[-0.06225218],
[-0.00081689],
[ 0.08864151],
[-0.03207344],
```

```
[ 0.03043966],
[ 0.00888341],
[ 0.00672779],
[-0.02021751],
[-0.02452876],
[-0.01159501],
[ 0.02612841],
[-0.05901875],
[-0.03638469],
[-0.02452876],
[ 0.01858372],
[-0.0902753 ],
[-0.00512814],
[-0.05255187],
[-0.02237314],
[-0.02021751],
[-0.0547075 ],
[-0.00620595],
[-0.01698407],
[ 0.05522933],
[ 0.07678558],
[ 0.01858372],
[-0.02237314],
[ 0.09295276],
[-0.03099563],
[ 0.03906215],
[-0.06117437],
[-0.00836158],
[-0.0374625 ],
[-0.01375064],
[ 0.07355214],
[-0.02452876],
[ 0.03367309],
[ 0.0347509 ],
[-0.03854032],
[-0.03961813],
[-0.00189471],
[-0.03099563],
[-0.046085  ],
```

```
 0.00133873],
 0.06492964],
 0.04013997],
-0.02345095],
 0.05307371],
 0.04013997],
-0.02021751],
 0.01427248],
-0.03422907],
 0.00672779],
 0.00457217],
 0.03043966],
 0.0519959 ],
 0.06169621],
-0.00728377],
 0.00564998],
 0.05415152],
-0.00836158],
 0.114509  ],
 0.06708527],
-0.05578531],
 0.03043966],
-0.02560657],
 0.10480869],
-0.00620595],
-0.04716281],
-0.04824063],
 0.08540807],
-0.01267283],
-0.03315126],
-0.00728377],
-0.01375064],
 0.05954058],
 0.02181716],
 0.01858372],
-0.01159501],
-0.00297252],
 0.01750591],
-0.02991782],
```

```
[-0.02021751],
[-0.05794093],
[ 0.06061839],
[-0.04069594],
[-0.07195249],
[-0.05578531],
[ 0.04552903],
[-0.00943939],
[-0.03315126],
[ 0.04984027],
[-0.08488624],
[ 0.00564998],
[ 0.02073935],
[-0.00728377],
[ 0.10480869],
[-0.02452876],
[-0.00620595],
[-0.03854032],
[ 0.13714305],
[ 0.17055523],
[ 0.00241654],
[ 0.03798434],
[-0.05794093],
[-0.00943939],
[-0.02345095],
[-0.0105172 ],
[-0.03422907],
[-0.00297252],
[ 0.06816308],
[ 0.00996123],
[ 0.00241654],
[-0.03854032],
[ 0.02612841],
[-0.08919748],
[ 0.06061839],
[-0.02884001],
[-0.02991782],
[-0.0191397 ],
[-0.04069594],
```

```
[ 0.01535029],
[-0.02452876],
[ 0.00133873],
[ 0.06924089],
[-0.06979687],
[-0.02991782],
[-0.046085  ],
[ 0.01858372],
[ 0.00133873],
[-0.03099563],
[-0.00405033],
[ 0.01535029],
[ 0.02289497],
[ 0.04552903],
[-0.04500719],
[-0.03315126],
[ 0.097264  ],
[ 0.05415152],
[ 0.12313149],
[-0.08057499],
[ 0.09295276],
[-0.05039625],
[-0.01159501],
[-0.0277622 ],
[ 0.05846277],
[ 0.08540807],
[-0.00081689],
[ 0.00672779],
[ 0.00888341],
[ 0.08001901],
[ 0.07139652],
[-0.02452876],
[-0.0547075 ],
[-0.03638469],
[ 0.0164281 ],
[ 0.07786339],
[-0.03961813],
[ 0.01103904],
[-0.04069594],
```

```
[-0.03422907],
[ 0.00564998],
[ 0.08864151],
[-0.03315126],
[-0.05686312],
[-0.03099563],
[ 0.05522933],
[-0.06009656],
[ 0.00133873],
[-0.02345095],
[-0.07410811],
[ 0.01966154],
[-0.01590626],
[-0.01590626],
[ 0.03906215],
[-0.0730303 ]])
```

In [19]:
```
# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
```

In [20]:
```
# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
```

In [21]:
```
# Create linear regression object
regr = linear_model.LinearRegression()
```

In [22]:
```
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
```

Out[22]:
```
LinearRegression()
```

In [23]:
```
# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
```

In [24]:
```
# The coefficients
```

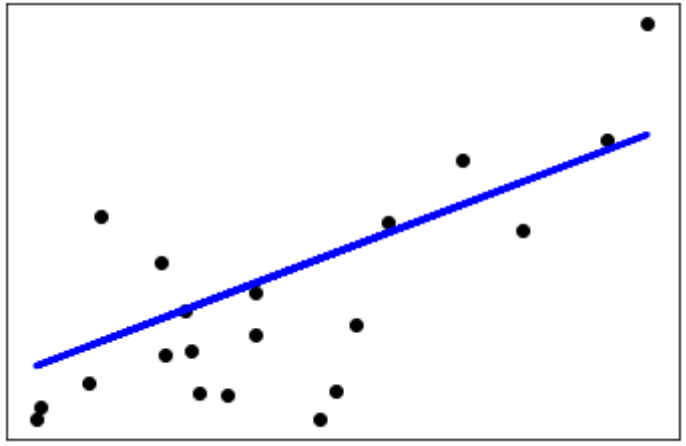
```
print('Coefficients: \n', regr.coef_)
```

```
Coefficients:
 [938.23786125]
```

In [25]:
```
print('Mean squared error: %.2f'% mean_squared_error(diabetes_y_test, d
iabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'% r2_score(diabetes_y_test, d
iabetes_y_pred))
```

```
Mean squared error: 2548.07
Coefficient of determination: 0.47
```

In [26]:
```
# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



In this model We have taken only one features then then the Mean squared error is 2548.07 ,

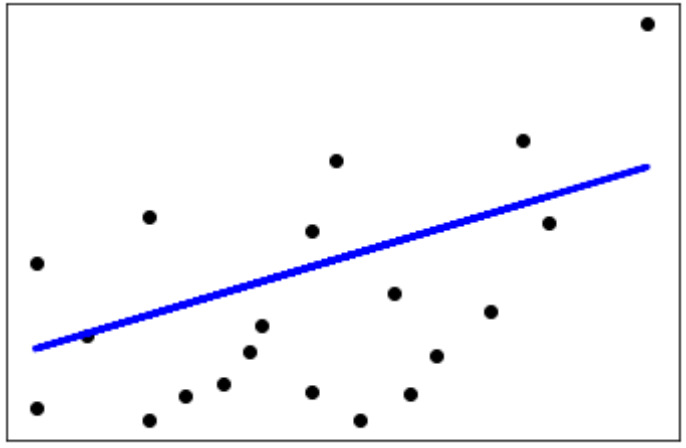
Coefficients is [938.23786125] and Coefficient of determination: 0.47

Change the features

In [27]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 3]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print('Mean squared error: %.2f'  % mean_squared_error(diabetes_y_test,
 diabetes_y_pred))
```

```
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f' % r2_score(diabetes_y_test,
diabetes_y_pred))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

```
Coefficients:
 [709.19471785]
Mean squared error: 4058.41
Coefficient of determination: 0.16
```



In the above data we have taken only 1 features and while i have taken 2 features then the Mean squared error is 4058.41 , Coefficients is [709.19471785] and Coefficient of determination: 0.16

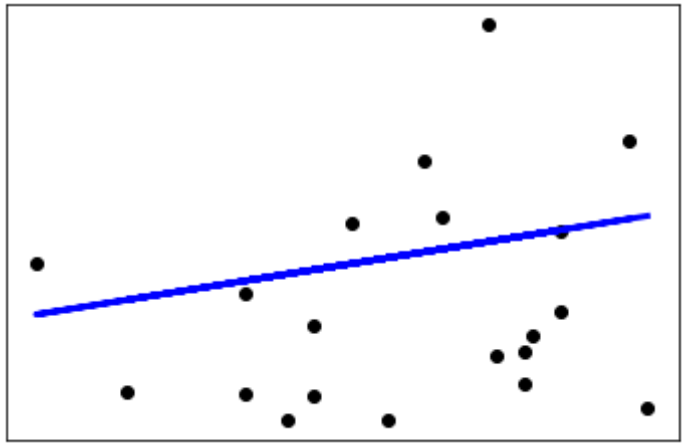
Increase the feature size to 2 and discuss the

output

In [28]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 4]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print('Mean squared error: %.2f' % mean_squared_error(diabetes_y_test,
diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f' % r2_score(diabetes_y_test,
diabetes_y_pred))
```

```
# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

```
Coefficients:
 [352.82770178]
Mean squared error: 5608.70
Coefficient of determination: -0.16
```



In the above data we have taken only 2 features and when i have taken 3 features then the Mean squared error is 5608.70 , Coefficients is [352.82770178] and Coefficient of determination: -0.16

In [ ]: