Regression Homework

This is the first assignment for CAP 4630 and we will develop two basic models with regression. You will use "**Tasks**" and "**Hints**" to finish the work. **(Total 100 Points)**

Task Overview:

- Singal Variable Nonlinear Regression
- Multiple Variable Linear Regression

1 - Packages

Import useful packages for scientific computing and data processing. **(5 Points)**

Tasks:

1. Import numpy and rename it to np.
2. Import pandas and rename it to pd.
3. Import the pyplot function in the libraray of matplotlib and rename it to plt.

References:

- numpy is the fundamental package for scientific computing with Python.
- matplotlib is a famous library to plot graphs in Python.

Attention:

1. After this renaming, you will use the new name to call functions. For example, **numpy** will become **np** in the following sections.

```
# Coding here
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from google.colab import files
```

2 - Data Preparation

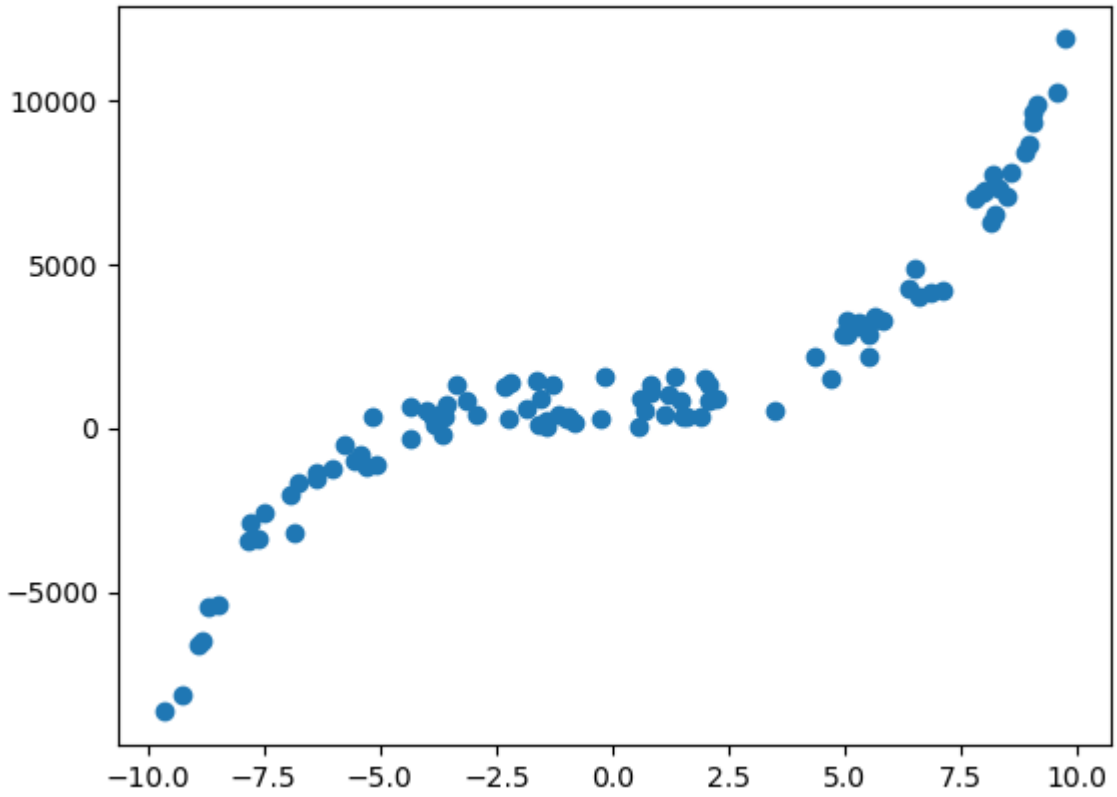
Prepare the data for regression task. **(10 Points)**

Tasks:

1. Load data for nonlinear regression.
2. Generate the scatter plot of the data.

Hints:

1. The data file is "data_nonlinear.csv".
2. The data format is as follows: 1st column is X and 2nd column is Y.
3. You may follow the example in class.




```
# Coding here
from google.colab import files
uploaded1 = files.upload()

from google.colab import files
uploaded2 = files.upload()

import io
data = pd.read_csv(io.BytesIO(uploaded1['data_nonlinear.csv']))
X = data.iloc[:, 0]
Y = data.iloc[:, 1]
plt.scatter(X, Y)
plt.show
```

Choose Files data_nonlinear.csv

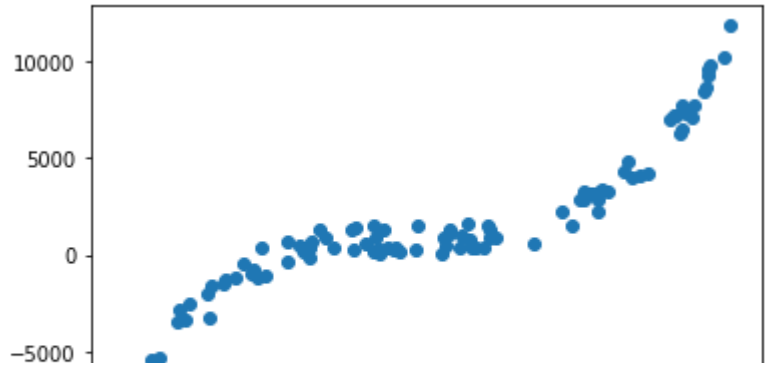
- **data_nonlinear.csv**(application/vnd.ms-excel) - 2556 bytes, last modified: 9/17/2021 - 100% done

```
Saving data_nonlinear.csv to data_nonlinear.csv
```

Choose Files data_two_variables.csv

- **data_two_variables.csv**(application/vnd.ms-excel) - 3674 bytes, last modified: 9/17/2021 - 100% done

```
Saving data_two_variables.csv to data_two_variables.csv
<function matplotlib.pyplot.show>
```



3 - Single Variable Nonlinear Regression

Develop a regression model, estimate coefficients with data, and derive the relationship. **(30 Points)**

Tasks:

1. Establish a relationship between Y and X with a cubic function.
2. Compute MSE loss with observation-prediction pairs.
3. Implement **Gradient Descent (GD)** to achieve optimal solution with the learning rate of **0.000001 (1e-6)** and **10000 (1e4)** epochs.
4. Print out the optimal solution at final step.

Hints:

1. Given the example of linear regression in class, modify the function to an equation for a spline with coefficients of **a** , **b**, **c** and **d** for cubic, qudractic, linear, and constant term.
2. Initialize the model with zero. For example, a=0, b=0, c=0 and d=0.
3. It may take **10-15 seconds** to finish the running for 10000 steps. Be patient.
4. For debugging, the results of **a**, **b**, **c**, **d** for first five steps are as follows:

Epoch 0 : 2.8045093168662314 0.15006631239563697 0.04047903434004733 0.0030023401200892003
Epoch 1 : 4.905935374329749 0.2803623842843468 0.07068280026181122 0.0057565282228493
Epoch 2 : 6.480417434500056 0.395779237410925 0.09318576969022647 0.008323648642107889
Epoch 3 : 7.65996806232127 0.4999280146312246 0.10991745268097952 0.010749486523089888

Epoch 4 : 8.543527816733905 0.5957208253596222 0.12232397430880633 0.013068360586717544

```
# Coding here
a = 0
b = 0
c = 0
d = 0

L = 0.000001  # Learning Rate
epochs = 10000 # Number of iterations to perform gradient descent

n = float(len(X)) # Number of elements in X column

# Performing the gradient descent
for i in range(epochs):
  Y_pred = a*X**3 + b*X**2 + c*X + d # Current predicted value of Y
  D_a = (-2/n) * sum(X**3 * (Y - Y_pred)) # Derivative wrt a
  D_b = (-2/n) * sum(X**2 * (Y - Y_pred)) # Derivative wrt b
  D_c = (-2/n) * sum(X * (Y - Y_pred)) # Derivative wrt c
  D_d = (-2/n) * sum(Y - Y_pred) # Derivative wrt d
  a = a - L * D_a # Update a
  b = b - L * D_b # Update b
  c = c - L * D_c # Update c
  d = d - L * D_d # Update d
  if i >= 0 and i <= 4:
    print("Epoch", str(i) + ":" + str(a), b, c, d)
```

```
Epoch 0:2.8045093168662314 0.15006631239563697 0.04047903434004733 0.0030023401200892003
Epoch 1:4.905935374329749 0.2803623842843468 0.07068280026181122 0.0057565282228493
Epoch 2:6.480417434500056 0.395779237410925 0.09318576969022647 0.008323648642107889
Epoch 3:7.65996806232127 0.49998280146312246 0.10991745268097952 0.010749486523089888
Epoch 4:8.543527816733905 0.5957208253596222 0.12232397430880633 0.013068360586717544
```

4 - Prediction Results

Derive prediction function and generate estmated results. **(5 Points)**

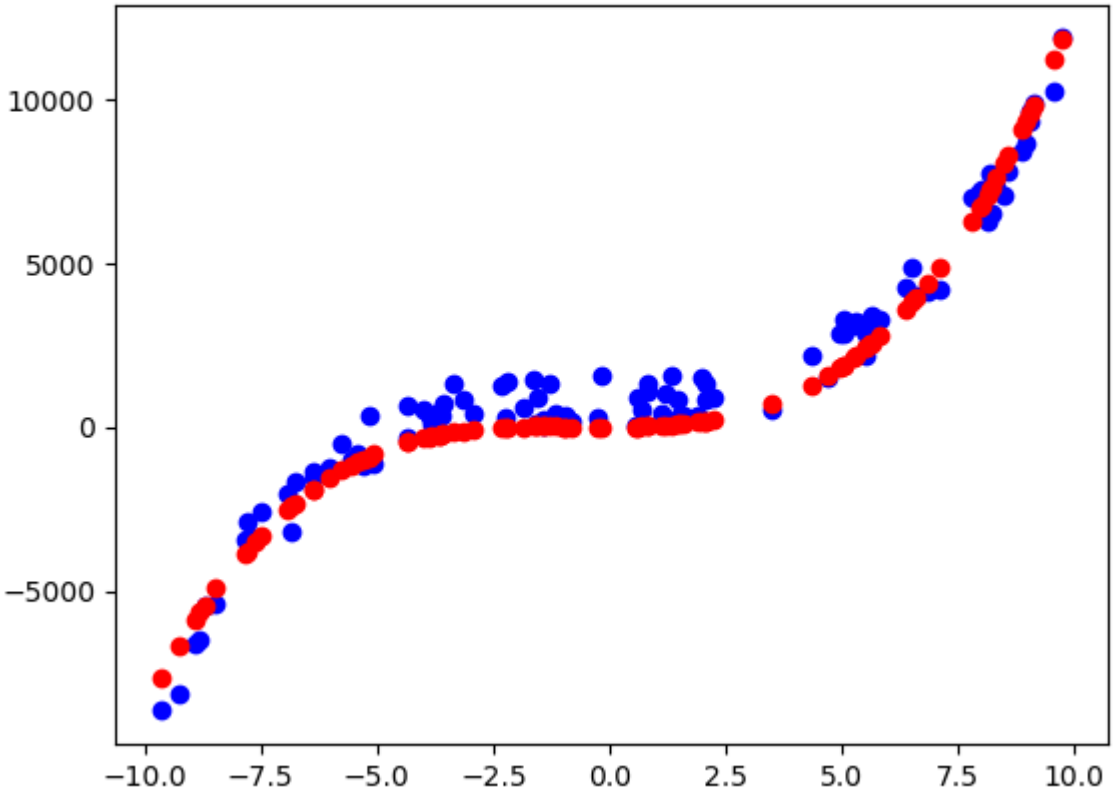
Tasks:

1. Derive prediction function with the obtained coefficients above.
2. Generate scatter plots for original data pairs X-Y and prediction results X-Y_Pred in the same figure.

Hint:

1. You may follow the example in class materials.

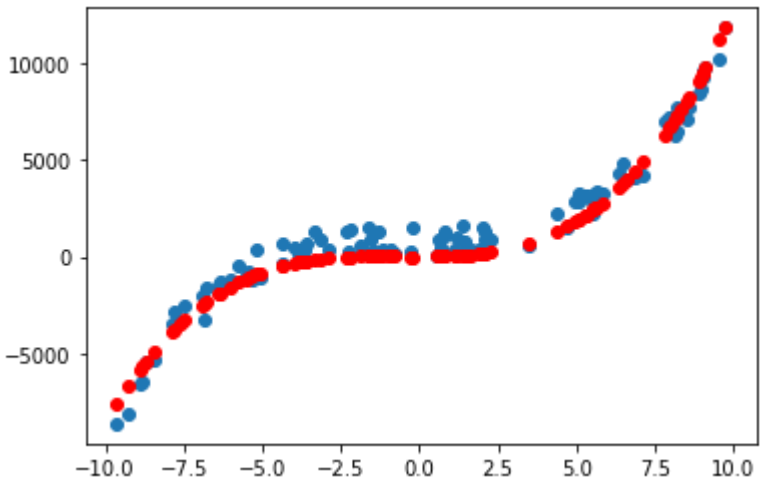
2. An example is shown below.




```
# Coding here
Y_pred = a*X**3 + b*X**2 + c*X + d # Current predicted value of Y

plt.scatter(X, Y)
plt.scatter(X, Y_pred, color='red') # predicted
plt.show
```

```
<function matplotlib.pyplot.show>
```



5 - Multiple Variables Linear Regression

5.1 Data Preparation

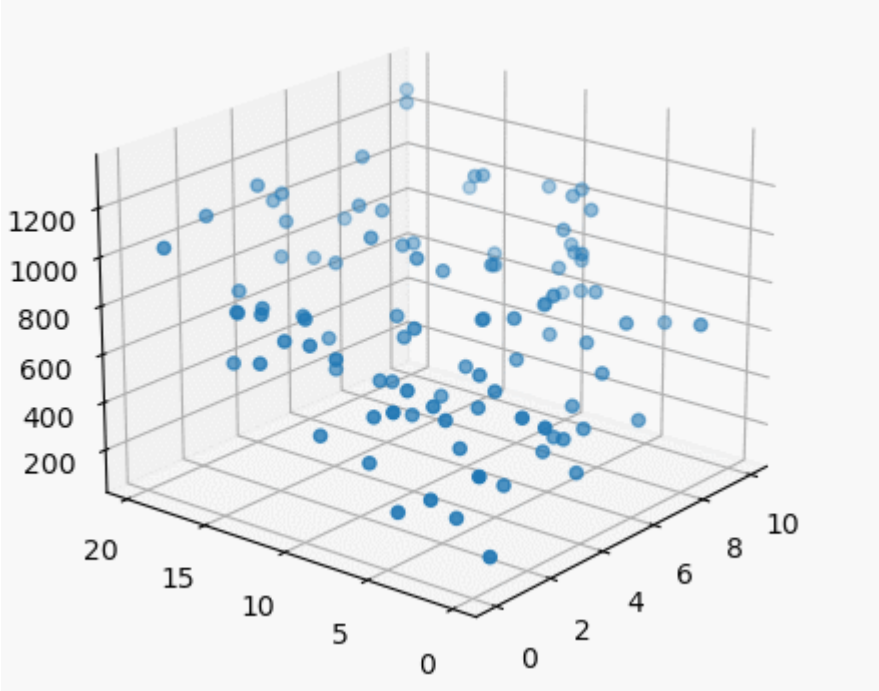
Prepare the data for regression task. **(10 Points)**

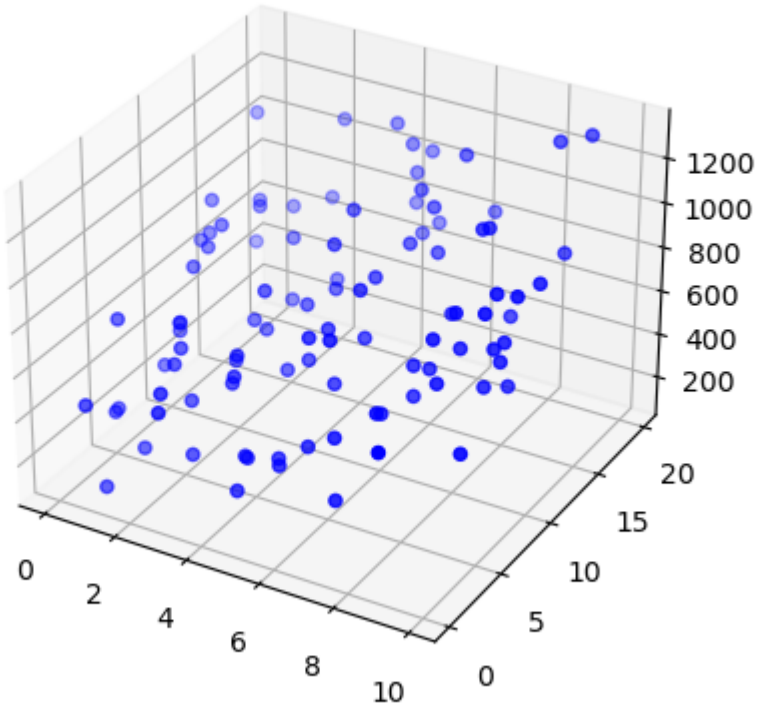
Tasks:

1. Load data for multiple variable linear regression.
2. Generate the 3D scatter plot of the data.

Hints:

1. The data file is "data_two_variables.csv".
2. The data format is as follows: 1st column is X1, 2nd column is X2, and 3rd colum is Y.
3. You may use "mplot3d" in the toolkit of "mpl_toolkits" and import "Axes3D" to faciliate 3D scatter plot. More details can be found in the reference of https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html
4. [Optional, NO Credit]You may rotate the figure you generated by using "%matplotlib qt" before you plot it. Remember to install the related package by "!pip install PyQt5". Only work on Jupyter(locally). Does not work on Google Colab. Reference Website

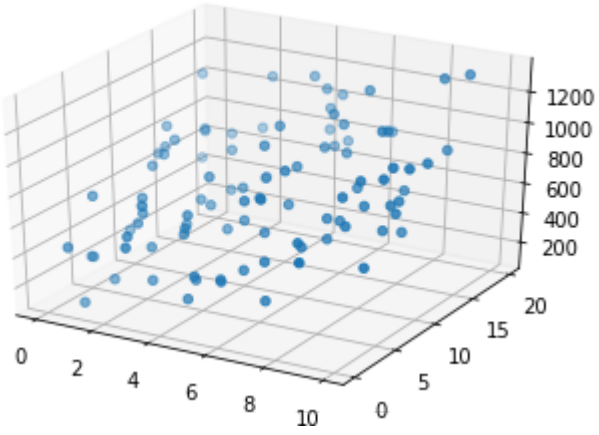





```
# Coding here
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

```
data = pd.read_csv(io.BytesIO(uploaded2['data_two_variables.csv']))
X1 = data.iloc[:, 0]
X2 = data.iloc[:, 1]
Y = data.iloc[:, 2]
ax.set_yticks([0, 5, 10, 15, 20])
ax.scatter(X1, X2, Y)
plt.show
```

```
<function matplotlib.pyplot.show>
```



5.2 Linear Regression

Develop a regression model, estimate coefficients with data, and derive the relationship. **(30 Points)**

Tasks:

1. Establish a linear function to describe the relationship among Y, X1, and X2.
2. Compute MSE loss with observation-prediction pairs.
3. Implement **Gradient Descent (GD)** to achieve optimal solution with the learning rate of **0.001 (1e-3)** and **10000 (1e4)** epochs.
4. Print out the optimal solution at final step.

Hints:

1. Given the example of linear regression in class, modify the function to a linear equation with two independent variables X1 and X2. The coefficients of X1 and X2 are **m1** and **m2**, respectively. The constant term is **m3**.
2. Initialize the model with zero. For example, m1=0, m2=0, and m3=0.
3. It may take **10-15 seconds** to finish the running for 10000 steps. Be patient.
4. For debugging, the results of **m1**, **m2**, and **m3** for first five steps are as follows:

Epoch 0: 7.43847600018326 15.595631430047339 1.4265844915879997
Epoch 1: 12.954483113402425 26.731746959534096 2.481143659135288
Epoch 2: 17.084193849045587 34.664109745712814 3.2680146970514863

Epoch 3: 20.213137348549306 40.2953527521597 3.8622050343066556
Epoch 4: 32.734943422646175 34.69592128962032 222.91661391579638

```
# Coding here
m1 = 0
m2 = 0
m3 = 0

L = 0.001  # Learning Rate
epochs = 10000 # Number of iterations to perform gradient descent

n = float(len(X1)) # Number of elements in X1 column

# Performing the gradient descent
for i in range(epochs):
  Y_pred = m1*X1 + m2*X2 + m3 # Current predicted value of Y
  D_m1 = (-2/n) * sum(X1 * (Y - Y_pred)) # Derivative wrt m1
  D_m2 = (-2/n) * sum(X2 * (Y - Y_pred)) # Derivative wrt m2
  D_m3 = (-2/n) * sum(Y - Y_pred) # Derivative wrt m3
  m1 = m1 - L * D_m1 # Update m1
  m2 = m2 - L * D_m2 # Update m2
  m3 = m3 - L * D_m3 # Update m3
  if i >= 0 and i <= 4:
    print("Epoch", str(i) + ": " + str(m1), m2, m3)
```

```
Epoch 0: 7.43847600018326 15.595631430047339 1.4265844915879997
Epoch 1: 12.954483113402425 26.731746959534096 2.481143659135288
Epoch 2: 17.084193849045587 34.664109745712814 3.2680146970514863
Epoch 3: 20.213137348549306 40.2953527521597 3.8622050343066556
Epoch 4: 22.618552798604984 44.274269323103674 4.317638791453634
```

5.3 - Prediction Results

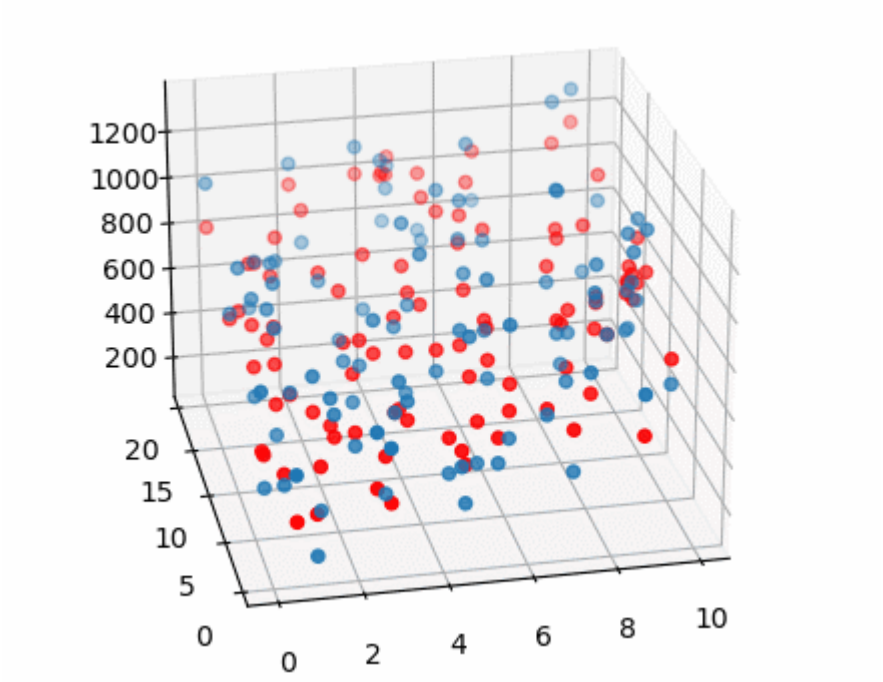
Derive prediction function and generate estmated results. **(10 Points)**

Tasks:

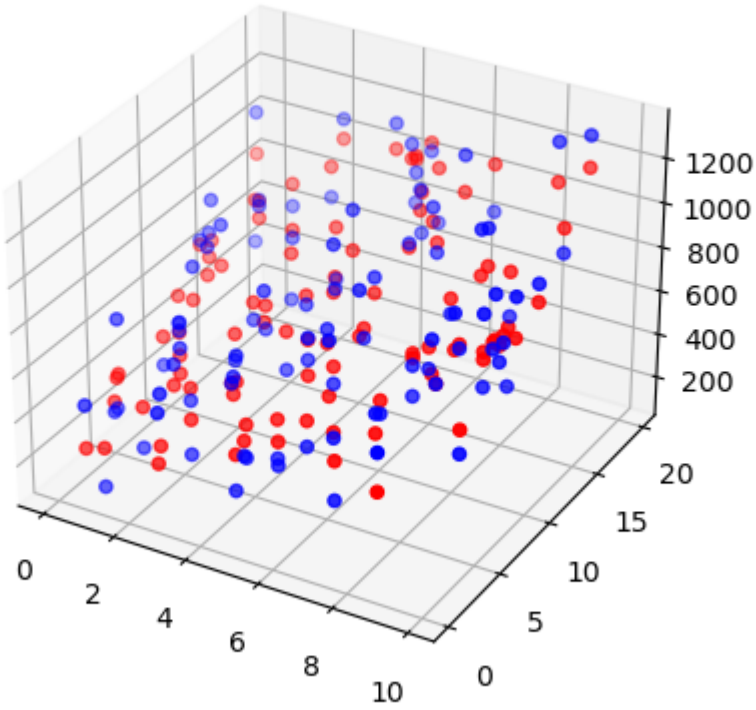
1. Derive prediction function with the obtained coefficients above.
2. Generate 3D scatter plots for original data pairs X-Y and prediction results X-Y_Pred in the same figure.

Hint:

1. You may follow the example above.



2. An example is shown below.



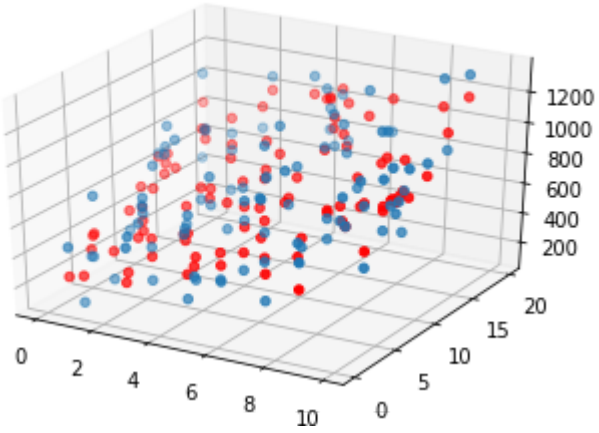
New Section

```
# Coding here
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

Y_pred = m1*X1 + m2*X2 + m3 # Current predicted value of Y

ax.set_yticks([0, 5, 10, 15, 20])
ax.scatter(X1, X2, Y)
ax.scatter(X1, X2, Y_pred, c='red') #predicted
```

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fb41a75d250>
```



✓ 0s completed at 1:23 PM