Artificial Intelligence Midterm Project

In this project, you will build a regression model and a classification model from scratch. Please follow the instructions closely, and only use Python's Numpy, Pandas, and matplotlib library to complete this project. Using functions from `sklearn` is not allowed.

Part I dues on Monday, March 22nd at 11:59 PM. **Part II** dues on Monday, April 12th at 11:59 PM.

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

Part I: A Regression Model

In this part, please build a multilinear regression model that extracts the relationship between housing prices and other relevant variables. The training data is shown in the table below:

```
data1 = pd.DataFrame({
    "YearBuilt": [1974, 1996, 1968, 1962, 1960],
    "YearSold": [2015, 2017, 2020, 2010, 2016],
    "Bedrooms": [3, 10, 4, 5, 6],
    "TotalArea": [1500, 4000, 1700, 2500, 2000],
    "Quality": [7.5, 6, 4, 5.5, 5],
    "Price": [358500, 452600, 352100, 341300, 342200]
})

data1
```

	YearBuilt	YearSold	Bedrooms	TotalArea	Quality	Price
0	1974	2015	3	1500	7.5	358500
1	1996	2017	10	4000	6.0	452600
2	1968	2020	4	1700	4.0	352100
3	1962	2010	5	2500	5.5	341300
4	1960	2016	6	2000	5.0	342200

Task 1: Data Transformation (10 pts)

Create a new column named "Age" that represents the age of each house when it was sold.

```
# Your Code Here
```

Task 2: Train a Multilinear Model (20 pts)

Assume that the price can be expressed as a linear combination of age, bedrooms, total area, and quality:

$$Price = \theta_0 + \theta_1 \cdot Age + \theta_2 \cdot Bedrooms + \theta_3 \cdot TotalArea + \theta_4 \cdot Quality.$$

Apply the normal equation to find the best values for the parameters:

1. Construct matrix $\mathbf{X}$ and $\mathbf{y}$ (the matrices are defined in Week 6 notebook and Chapter 4 of the textbook).
2. Calculate the parameter vector using the normal equation $\theta = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$

```
# Your Code Here
```

Task 3: Make A Prediction (10 pts)

Suppose that there is another house with the following attribute:

- YearBuilt: 1985
- YearSold: 2021
- Bedrooms: 6
- Total Area: 2500
- Quality: 5.5

Use the parameter values that you have calculated to make a prediction on its sale price.

```
# Your Code Here
```

Part II: A Classification Model

In this part, we will build a logistic regression model and evaluate its performance on the classifying the data. The dataset is as follows:

```
data2 = pd.DataFrame([[5.0, 2.0, 1],
                      [6.2, 3.4, 1]
```

```
                        [6.2, 3.4, 1],
                        [4.9, 3.6, 0],
                        [6.2, 2.2, 1],
                        [5.7, 3.0, 1],
                        [4.8, 3.4, 0],
                        [5.0, 3.4, 0]],
                       columns=["x1", "x2", "class"])
data2
```

	x1	x2	class
0	5.0	2.0	1
1	6.2	3.4	1
2	4.9	3.6	0
3	6.2	2.2	1
4	5.7	3.0	1
5	4.8	3.4	0
6	5.0	3.4	0

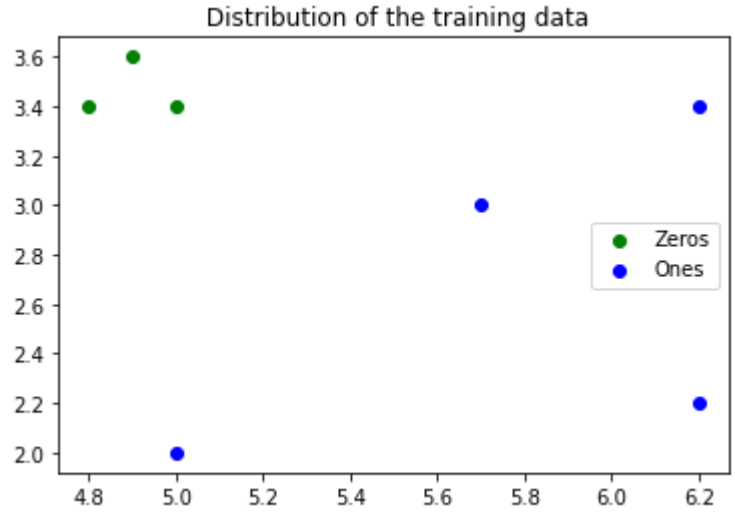
Task 1: Data Visualization (10 pts)

Visualize the data as a scatter plot. Show class 0 records as green dots and class 1 records as blue dots. Display the following items:

- Title of the plot: Distribution of the training data
- Label for x axis: x1
- Label for y axis: x2
- Legend

```
# Your Code Here
# Let draw the scatter plot

index_0 = (data2['class'] == 0)
index_1 = (data2['class'] == 1)

plt.scatter(data2.loc[index_0, 'x1'], data2.loc[index_0, 'x2'], c = 'green', label = 'Zeros')
plt.scatter(data2.loc[index_1, 'x1'], data2.loc[index_1, 'x2'], c = 'blue', label = 'Ones')

plt.title(" Distribution of the training data")
plt.xlabel("x1")
plt.xlabel("x2")

plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f3560450a90>
```



Task 2: Apply A Logistic Regression Model (10 pts)

Suppose that you are given a logistic regression model with explicity paramter values:

$p = \sigma(\mathbf{x} \cdot \theta^T)$.

where

- p: the probability that the point belongs to class 1.
- $\mathbf{x} = (1, x_1, x_2)$.
- $\theta = (\theta_0, \theta_1, \theta_2) = (-2.15, 0.92, -0.82)$.
- $\sigma(t) = \frac{1}{1+e^{-t}}$

Find the model's prediction on the following test set:

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

```
data3 = pd.DataFrame([[5.1, 3.4, 0],
                      [6.5, 2.8, 1],
                      [5.8, 2.7, 1],
                      [4.6, 3.1, 0]],
                     columns = ["x1", "x2", "class"])

data3
```

	x1	x2	class
0	5.1	3.4	0

```
import numpy as np

# Assume the parameter vector is as follows:
# Assuming that there are three input variables
theta = np.array([-2.15,0.92,-0.82])

# Now calculate y for input x
# x = np.array([1,data3.iloc[:,0],data3.iloc[:,1]])
x = np.hstack([np.ones([4, 1]), data3.loc[:, ['x1', 'x2']]])

t = x.dot(theta)

y = 1 / (1 + np.e ** (-t))


# y.iloc[:,0]
print("y:", y)
```

```
y: [0.43880828 0.82259081 0.72551796 0.38698582]
```

```
predictions = []

for i in y:
  if i < .5:
    prediction = 0
    predictions.append(prediction)
  else:
    prediction = 1
    predictions.append(prediction)

predictions = np.array(predictions)
print(predictions)
```

```
[0 1 1 0]
```

Task 3: Model Evaluation (40 pts)

Calculate the following model metrics regarding the performance on the test set:

- classification accuracy
- precision score
- recall score
- F-1 score

```
# Classification Accuracy
array1 = np.array(data3['class'])
array2 = np.array(predictions)
print(array1)
print(array2)

# count the number of pairs that have identical values
#USE A LOOP TO MANULLY COMPARE THE ERRORS
count = 0
for i in range(len(array1)):
 actual = array1[i]
 pred = array2[i]
 if actual == pred:
  count = count + 1
print(count)

accuracy = count / len(array1)
print("Accuracy:",accuracy)
```

```
[0 1 1 0]
[0 1 1 0]
4
Accuracy: 1.0
```

```
# first we need the number of true positives, false positives, false negatives
num_true_positives = 0
for i in range(len(array1)):
 label = array1[i]
 pred = array2[i]
 if (label == 1 and pred == 1):
  num_true_positives = num_true_positives +1
print(num_true_positives)
```

```
2
```

```
num_false_positives = 0
for i in range(len(array1)):
 label = array1[i]
 pred = array2[i]
 if (label == 0 and pred == 1):
  num_false_positives = num_false_positives +1
print(num_false_positives)
```

```
0
```

```
num_false_neg = 0
for i in range(len(array1)):
 label = array1[i]
 pred = array2[i]
 if (label == 1 and pred == 0):
  num_false_neg = num_false_neg +1
print(num_false_neg)
```

```
0
```

```
# Precision Score
precision = num_true_positives/num_true_positives + num_false_positives
print(precision)
```

```
1.0
```

```
# Recall Score
recall = num_true_positives/ num_true_positives + num_false_neg
print(recall)
#calculate recall: num tru pos / num true pos + num false neg
```

```
1.0
```

```
# F-1 Score
F1_Score = 2*(recall * precision) / (recall + precision)
print(F1_Score)
```

```
1.0
```