Assignment II - Rick Kabuto
CS-436/536-01: Intro to Machine Learning
Due: 06/23 23:59

1. **Get the data and data description from the UCI repository about 3 different flowers (“Iris-setosa”, “Iris-versicolor”, “Iris-virginica”).**
    - Source: https://archive.ics.uci.edu/ml/datasets/iris
    - Data contains 4 features/specifications (all in cm): sepal length, sepal width, petal length, and petal width
2. **Build a logistic regression model using 2 features of your choice (e.g., “sepal length” and “sepal width”) as input, and any 2 flowers of your choice (e.g., “Iris-setosa” and “Iris-virginica”) as binary classifiable output.**
    - [10 points] Plot the distribution of your selected data.
      More specifically, a 2D scatter plot with the axes being the 2 features.
    - [10 points] Split your data into 80% Training and 20% Testing.
    - [60 points (for model & training)] Plot the learning curve while your model is being trained (i.e., accuracy on training data vs. number of epochs).
        - Please feel free to use any value of weights and learning rate. Report your model weights upon completion of training.
    - [20 points] Report the accuracy on test data using the trained model.

Hint: If your model is not performing well for the flowers/features you selected, try changing your selection.

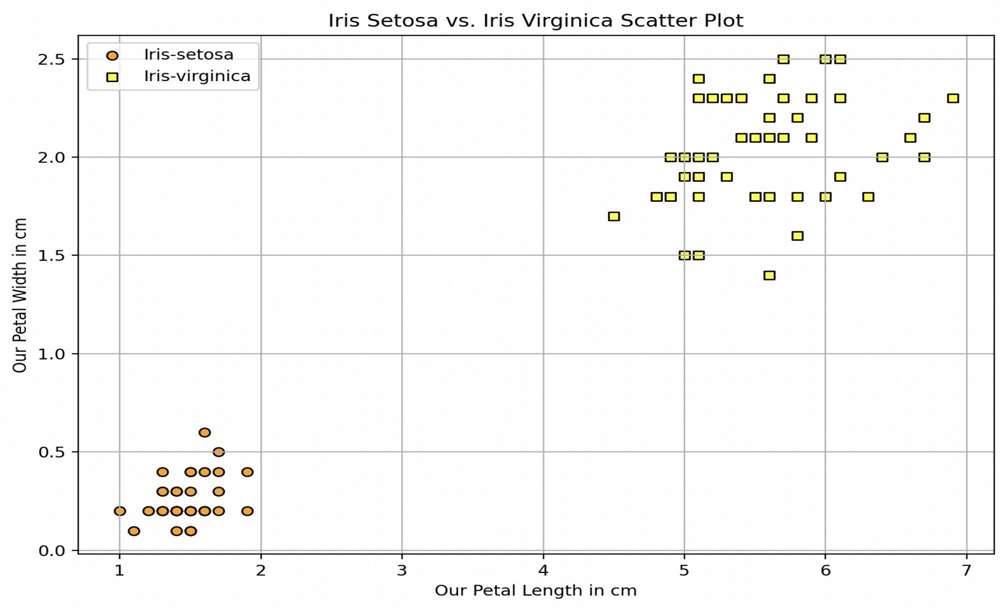
Submission

1. 5% deduction for every late day and maximum of 7 late days are allowed.
2. Please submit a single PDF file that includes all your code with comments, plots, and summary. Please be sure to address all the problems in the Description section.
3. Code should be computer-readable – NO SCREENSHOTS.

Build a logistic regression model using 2 features of your choice (e.g., “sepal length” and “sepal width”) as input, and any 2 flowers of your choice (e.g., “Iris-setosa” and “Iris-virginica”) as binary classifiable output.

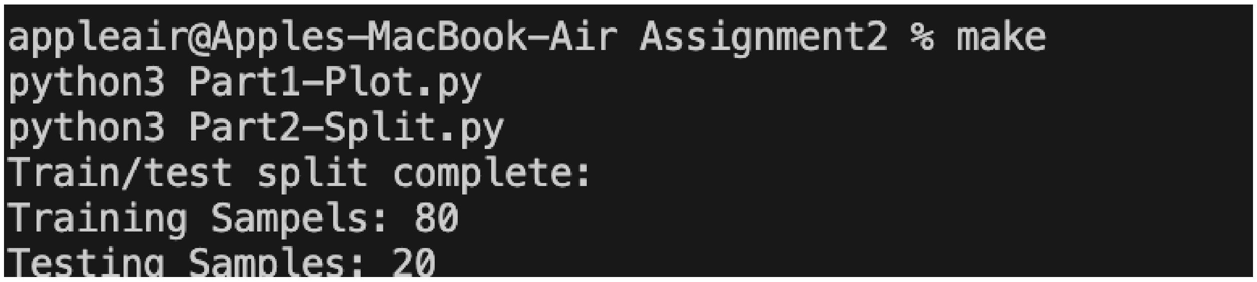
Part 1 - Plot the distribution of your selected data.

```
#-=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-
#Build a logistic regression model using 2 features of your choice as input, and any 2 flowers of your choice
#as binary classifiable output. [10 points] Plot the distribution of your selected data.
# More specifically, a 2D scatter plot with the axes being the 2 features.
#-=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-
import pandas as pd
import matplotlib.pyplot as plt
#-=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-
# Psuedocode:
# Step 1 - Read the data from the iris.data without the headers included and assign coloumn names
# Step 2 - Define the Plot and what the x and y axes are for each feature
# Step 3 - Create a figure and axes object for plotting
# Step 4 - Split the data by class for seperate plotting. Iris-Setosa = Orange Iris-Virgincia = Yellow
# Step 5 - Construct the graph - Label the axes, title the plot, and adjust to prevent label overlap etc.
#-=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-
df = pd.read_csv('iris.data', header=None)
df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
df = df[df['class'].isin(['Iris-setosa', 'Iris-virginica'])]

XAxis = 'petal_length'
YAxis = 'petal_width'

fig, ax = plt.subplots(figsize=(8,6))
DataSet = df[df['class'] == 'Iris-setosa']
DataVirg = df[df['class'] == 'Iris-virginica']

ax.scatter(DataSet[XAxis], DataSet[YAxis], label = 'Iris-setosa', marker = 'o', color = 'orange', edgecolor = 'k')
ax.scatter(DataVirg[XAxis], DataVirg[YAxis], label = 'Iris-virginica', marker = 's', color = 'yellow', edgecolor = 'k')
ax.set_xlabel('Our Petal Length in cm')
ax.set_ylabel('Our Petal Width in cm')
ax.set_title('Iris Setosa vs. Iris Virginica Scatter Plot')
ax.legend()
ax.grid(True)
fig.tight_layout()
plt.show()
```



Part 2 - Split your data into 80% Training and 20% Testing.

```
#Split your data into 80% Training and 20% Testing.
import pandas as pd
from sklearn.model_selection import train_test_split
#-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-
# PsuedoCode:
# Step 1 - Read the data from the iris.data without the headers included and assign coloumn name
# Step 2 - Set 0 = Setosa and Set 1 = Virgincia
# Step 3 - Extract the feature matrix and label array and split data into training sets(80/20 split)
# Step 4 - Print the result of the split
#-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-
iris_df = pd.read_csv('iris.data', header=None)
iris_df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
iris_df = iris_df.query("species == 'Iris-setosa' or species == 'Iris-virginica'")

iris_df['target'] = iris_df['species'].apply(lambda x: 0 if x == 'Iris-setosa' else 1)

selected_features = iris_df.loc[:, ['petal_length', 'petal_width']].to_numpy()
target_values = iris_df['target'].to_numpy()

TrainFeature, TestFeature, TrainLabel, TestLabel = train_test_split(selected_features, target_values, test_size=0.2,
random_state=42)

print('Train/test split complete:')
print(f"Training Sampels: {len(TrainFeature)}")
print(f"Testing Samples: {len(TestFeature)}")
```



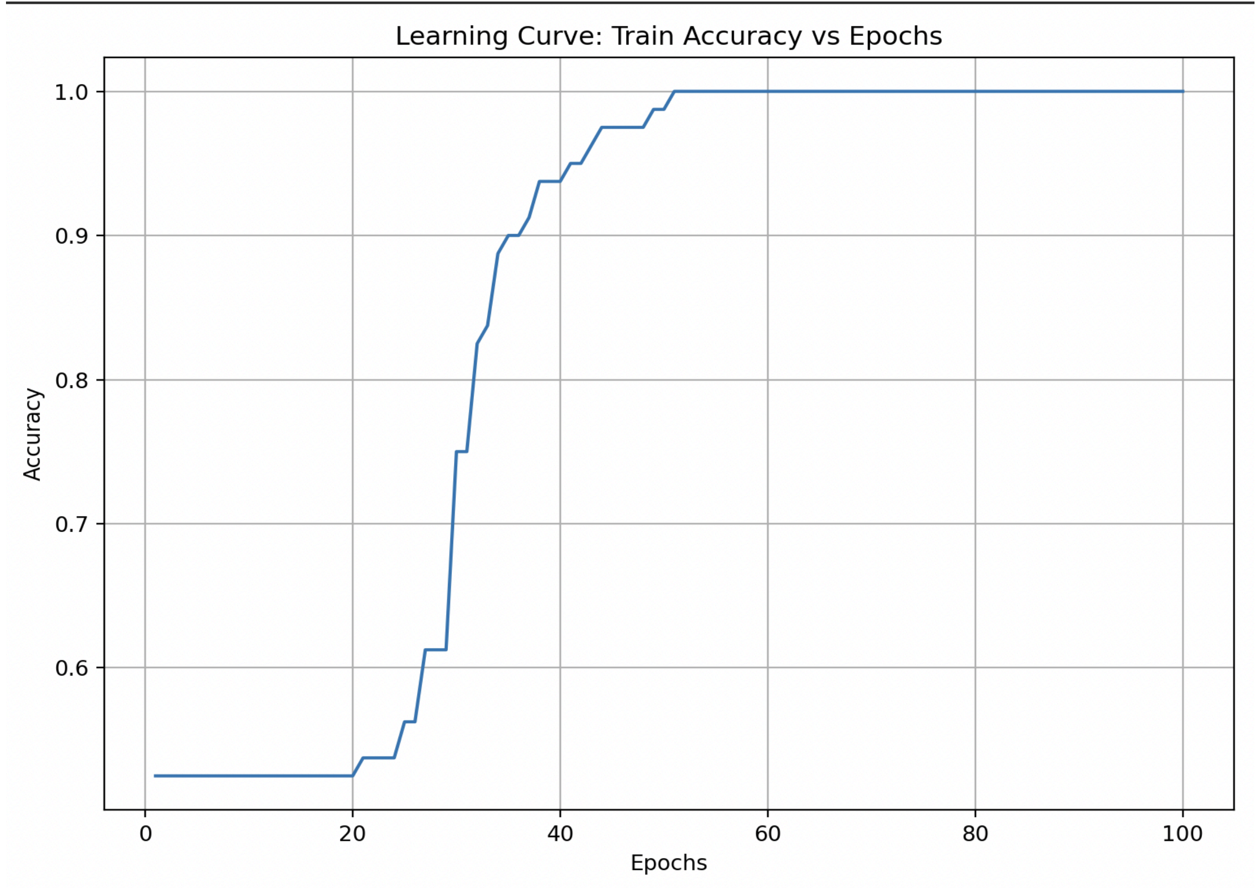
```
appleair@Apples-MacBook-Air Assignment2 % make
python3 Part1-Plot.py
python3 Part2-Split.py
Train/test split complete:
Training Sampels: 80
Testing Samples: 20
```

Part 3 - Plot the learning curve while your model is being trained (i.e., accuracy on training data vs. number of epochs). Please feel free to use any value of weights and learning rate. Report your model weights upon completion of training.

```
#-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-
# Part 3 - Plot the learning curve while your model is being trained (i.e., accuracy
# on training data vs. number of epochs). Please feel free to use any value of weights
# and learning rate. Report your model weights upon completion of training.
#-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
#-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-=--=-=-=-=-
```

```
# Psuedocode:
# Step 1) Read the CSV file and Iris dataset and construct Coloumns
# Step 2) Set Setosa = 0 and Set Virgincia = 1
# Step 3) Extract the petal features and add a bias as the first feature
# Step 4) Split the data into training and test sets which we did from Part2
# Step 5) Construct the functions related to our logistic regression
#       a) Construct our signmoid function
#       b) Using our weight vector, create the probabilites of prediction
#       c) Compute the classification accuracy for our predictions
# Step 6) Train our logistic regression using a gradient decscent
# Step 7) Output the final model weights and Plot our accuracy
# Step 8) Evaluate the accuracy
#-=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=--=-=-=-=-
df = pd.read_csv('iris.data', header=None)
df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
df = df[df['class'].isin(['Iris-setosa', 'Iris-virginica'])]
df['target'] = df['class'].map({'Iris-setosa': 0, 'Iris-virginica': 1})

features = df[['petal_length', 'petal_width']].to_numpy()
targets = df['target'].to_numpy()
features = np.insert(features, 0, 1, axis=1)
TrainFeature, TestFeature, TrainLabel, TestLabel = train_test_split(features, targets, test_size=0.2, random_state=42)

def sigmoid(value):
  return 1 / (1 + np.exp(-value))
def predict(data, weight_vec):
  return np.array([sigmoid(np.dot(row, weight_vec)) for row in data])
def calculateAccuracy(data, labels, weight_vec):
  predicted_probs = predict(data, weight_vec)
  predictions = (predicted_probs >= 0.5).astype(int)
  return np.mean(predictions == labels)

LearningRate = 0.1
Iterations = 100
W = np.zeros(features.shape[1])
AccuracyMeasurment = []

for CurrentEpoch in range(Iterations):
  PredictProb = predict(TrainFeature, W)
  TrueLoss = []
  for i, _ in enumerate(TrainFeature[0]):
    partial = np.sum((PredictProb - TrainLabel) * TrainFeature[:, i] / len(TrainLabel))
    TrueLoss.append(partial)
  W -= LearningRate * np.array(TrueLoss)
  ComputedAccuracy = calculateAccuracy(TrainFeature, TrainLabel, W)
  AccuracyMeasurment.append(ComputedAccuracy)

print('Train Complete')
print('Final model Weights:')
```

```
for i, weight in enumerate(W):
  print(f"  w{i}: {weight:.4f}")

fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(range(1, Iterations + 1), AccuracyMeasurment, label = 'Training Accuracy')
ax.set_xlabel('Epochs')
ax.set_ylabel('Accuracy')
ax.set_title('Learning Curve: Train Accuracy vs Epochs')
ax.grid(True)
fig.tight_layout()
plt.show()

TrueAccuracy = calculateAccuracy(TestFeature, TestLabel, W)
print(f"\nTest Accuracy: {TrueAccuracy * 100:.2f}%")
```



```
Final model Weights:
w0: -1.5016
w1: 0.4183
w2: 0.7721
Test Accuracy: 100.00%
```

Part 4 - [20 points] Report the accuracy on test data using the trained model.

After analyzing the graphs, the accuracy using the train model achieved a testing accuracy of 100%. Although getting our Machine Learning to reach 100% test accuracy is out of the norm, in this specific case, 100% accuracy is realistic and expected. This is because the dataset is small and clean, and the classes I chose, which were Iris setosa and Iris virginica, are linearly separable using petal length and petal width. Thus, these features create distinct clusters with no overlap, making the classification task ideal for logistic regression models.

The weights that were outputted during the train were the bias(w0) = -1.5016, the petal length(w1) being 0.4183, and the petal width(w2) being 0.7721. These weights tell us that there was a clean decision boundary that recognizes and responds well to new and unknown data. Furthermore, the model's perfect testing accuracy indicates that it successfully captured the patterns in the data and made reliable predictions, proving our test to be successful.