CS7CS4 Machine Learning
Week 2 Assignment

Michael Millard
Student ID: 24364218

October 5, 2025

Question (a)

(a)(i)

The dataset provided consists of two input features, X_1 and X_2, and a set of corresponding target values, y, which are a numeric label of either $+1$ or -1. A useful way to visualize this dataset is by plotting the input features against each other in a scatter plot with X_1 on the x-axis and X_2 on the y-axis using different markers to plot each point based on its label value, as shown in Figure 1 below.

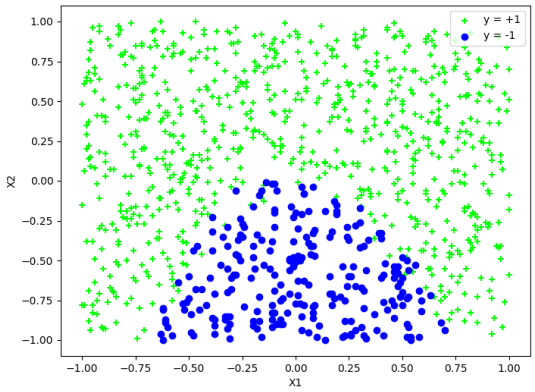

Figure 1: Scatter plot of input features X_1 vs X_2 with different markers based on the corresponding label y value at each point

In Figure 1 above, the label (or target value) corresponding to each pair of (x_1, x_2) input feature points determine the colour and type of the marker. If the data point corresponds to a label of $+1$, it is plotted as a green cross; else if it corresponds to a label of -1, it is plotted as a blue circle.

(a)(ii)

The dataset was used to train a logistic regression classifier model using the sklearn toolkit. The resulting model parameters were $[\theta_0, \theta_1, \theta_2] = [2.318229 \quad -0.153643 \quad 4.297776]$, which correspond to the y-intercept and weights of the input feature vector $[1 \quad x_1 \quad x_2]^T$ (the first element of 1 is added in order to allow the model equation to be expressed as a vector product $\boldsymbol{\theta}^T\boldsymbol{x}$). This vector product defines a plane in 3-dimensional space with the equation $y = 2.318229 - 0.153643X_1 + 4.297776X_2$. From these coefficient values it can be seen that changes in input feature X_2 result in a significantly greater change in y than changes in input feature X_1 do. This is a result of X_2's high coefficient of 4.297776, relative to X_1's small coefficient of -0.153643. It is also worth noting that as X_2 increases, y increases, since it has a positive coefficient. Whereas when X_1 increases, y decreases due to X_1's negative coefficient.

(a)(iii)

Figure 2 below shows the predictions made by the logistic regression classifier on the training data. The ground truth markers remain the same (green crosses for labels of $+1$ and blue circles for labels of -1),

the markers for predictions are a red cross for a predicted label of +1 and a purple circle for a prediction of -1. Also included in Figure 2 is the linear decision boundary, which is a straight line given by the

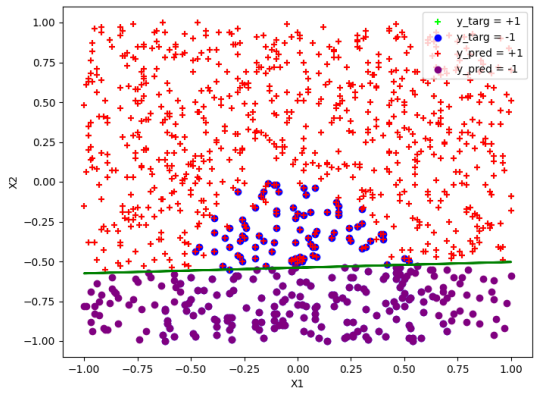

Figure 2: Scatter plot showing predicted values vs actual target values and the decision boundary for the trained logistic regression classifier

equation $X_2 = 0.035749X_1 - 0.539402$. The parameters of which are obtained by taking the equation of the trained model, $y = 2.318229 - 0.153643X_1 + 4.297776X_2$, equating it to zero and isolating X_2. Thereby expressing X_2 as a function of X_1, which is the only dependent variable in the equation and hence gives the equation of a line.

(a)(iv)

From the plot in Figure 2 it can be seen that this linear decision boundary is not well suited to the dataset in that the data is not linearly separable. As a result of this, many of the target values of +1 clustered below the centre of the plot and target values of -1 in the bottom left and bottom right corners are misclassified by the predictor. The accuracy of this logistic regression model is 84%, obtained using the accuracy_score function from the sklearn kit (see Appendix A). This indicates that the predictor is still correct for the vast majority of the cases. However, visual inspection of the data suggests a quadratic decision boundary may be better suited.

Question (b)

(b)(i)

For this question various linear support vector machine (SVM) classifiers were trained on the same dataset, each with a different penalty parameter given by $\frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{C}$, where C is a constant hyperparameter. The penalty is inversely proportional to the C value. SVM models were trained using 6 different values for C in the range [0.001, 100] with each iteration scaling the value of C by 10x. Since there are two input features, each iteration produced a linear SVM classifer of the form $\boldsymbol{\theta}^T\boldsymbol{x} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$. The various linear SVM models and their corresponding parameter values are given below in Table 1.

C	θ_0	θ_1	θ_2	$\frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{C}$	**Accuracy**
0.001	0.347334	-0.014827	0.362909	252.5637	0.777778
0.010	0.538418	-0.033092	0.904286	110.872281	0.833834
0.100	0.714055	-0.046017	1.350437	23.356728	0.842843
1.000	0.762760	-0.048220	1.457959	2.709774	0.843844
10.000	0.768670	-0.048483	1.470668	0.275607	0.843844
100.000	0.769271	-0.048509	1.471959	0.027608	0.843844

Table 1: Linear SVM model parameters for various penalty parameter values

It can be seen in Table 1 above that small values for C result in large penalty factors, whereas values over C = 1 yield very small penalty factors. Also worth noting is that the once C has increased past 0.01, the gains in model accuracy are negligible.

(b)(ii)

For each of the set of parameters tabulated above in Table 1, a new scatter plot was created using the trained linear SVM model in each case to predict the labels. In each instance, the linear decision boundary was computed by setting the model equation $y = \boldsymbol{\theta}^T \boldsymbol{x}$ equal to zero and isolating input feature X_2. Predictions of +1 are plotted as red crossed, predictions of -1 are plotted as purple circles, and the decision boundary is plotted in green, as shown in Figure 3 below.

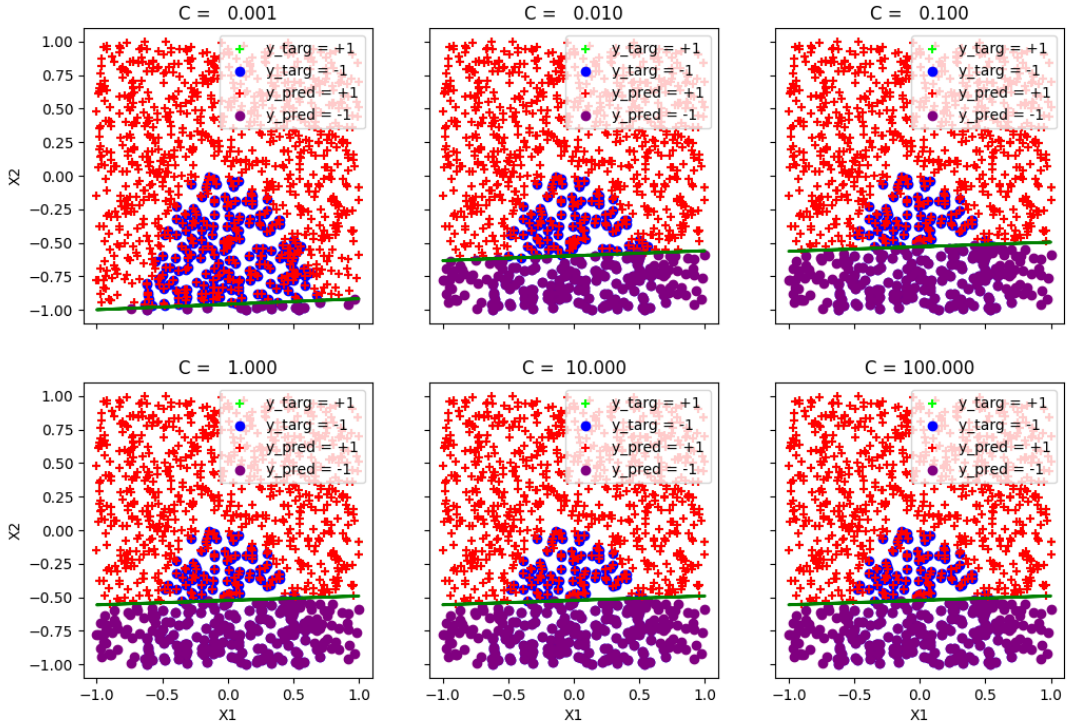

Figure 3: Various scatter plots showing the predictions of linear SVM models trained using different penalty parameters

(b)(iii)

It can be seen in Table 1 above that increasing C results in increases of the model parameters $\boldsymbol{\theta} = [\theta_0 \quad \theta_1 \quad \theta_2]^T$ and simultaneous decreases in the value of the penalty factor $\frac{\theta^T \theta}{C}$. The higher the value of C, the smaller the penalty factor and the better the SVM model performed, as is evident in Figure 3 and Table 1 above. However, this only holds true for increases in small values of C (e.g. C = 0.001 to C = 0.01). Once the value of C has exceeded 0.01, the changes in the model parameters and resulting model performance are minimal, as shown in Table 1. This is consistent with the similar predictions and decision boundaries present in the sub-figures from C = 0.01 to C = 100 in Figure 3 above.

(b)(iv)

The model parameters for the logistic regression model were $\boldsymbol{\theta} = [2.318229 \quad -0.153643 \quad 4.297776]^T$ and the parameters of the best performing SVM model (C = 100.0, i.e. the smallest penalty factor) were $\boldsymbol{\theta} = [0.769271 \quad -0.048509 \quad 1.471959]^T$. The logistic regression model and the best performing model were 84% and 84.4844%, respectively. As such, the models performed to a very similar standard with the best performing SVM model slightly outperforming the logistic regression model (by a mere 0.4844%).

Question (c)

(c)(i)

For this question, two new feature parameters, X_3 and X_4, were created as the square of each element in X_1 and X_2, respectively (i.e. $X_3 = X_1^2$ and $X_4 = X_2^2$). Using these four input features

and the same original labels as training data, a new logistic regression classifier was trained using the sklearn logisitic regression toolkit and the resulting model parameters were $\boldsymbol{\theta} = [0.178761 \quad -0.746359 \quad 20.692164 \quad 40.582676 \quad -0.037979]^T$. Writing out the full model equation gives $y = \boldsymbol{\theta}^T \boldsymbol{x} = 0.178761 - 0.746359X_1 + 20.692164X_2 + 40.582676X_3 - 0.037979X_4$. From these coefficient values, it is evident that changes in X_2 and X_3 yield far greater changes in y than changes in X_1 and X_4 do. Additionally, increases in X_2 and X_3 result in increases in y, whereas increases in X_1 and X_4 cause y to decrease.

(c)(ii)

The predictions made by the newly trained logistic regression classifier are plotted below in Figure 4. Again, label predictions of +1 are plotted as red crosses and label predictions of -1 are plotted as purple circles.

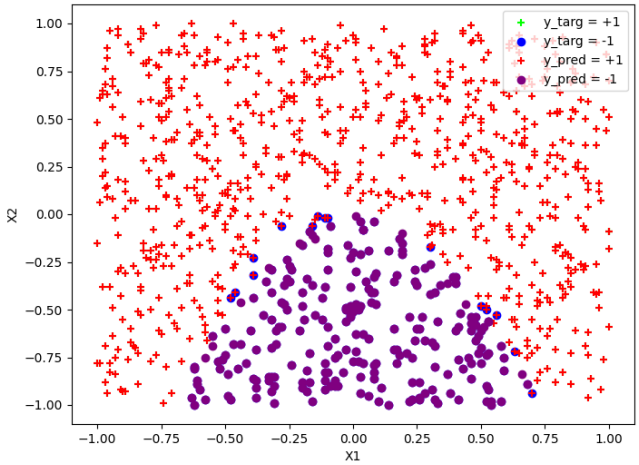

Figure 4: Scatter plot of actual target values and predictions made by the logistic regression model with four input features

It is immediately evident in Figure 4 above that the accuracy of this new logistic regression model is significantly more accurate than any of the previously trained models in this assignment. This new model does not try to fit a linear decision boundary on this non-linearly separable dataset, but rather a quadratic decision boundary, which upon inspection, is a significantly better fit. This is confirmed by the model having an accuracy of 97.10%, which was obtained using the sklearn's accuracy_score function. As such, this model is the best performing model on this dataset and outperforms the second best performing model (the SVM model with the highest C value) by 12.62%.

(c)(iii)

A baseline predictor was created using sklearn's DummyClassifier model with the *strategy* parameter set to "most_frequent", which is simply a model that predicts the most common occurring label in the dataset for all inputs. This simple model achieved an accuracy of 76%, which is significantly lower than the newly trained logistic regression model's accuracy of 97.10%. As such, this model both outperforms the baseline predictor by 21.10% and outperforms all other models trained in this assignment by at least 12.62%; making it the best suited model to make predictions on this type of data.

(c)(iv)

Since there are four input features, the logistic regression model has five coefficients: four coefficients for the input features and one intercept value. Since the model equation is

$$y = \boldsymbol{\theta}^T \boldsymbol{x} = 0.1788 - 0.7464X_1 + 20.6922X_2 + 40.5827X_3 - 0.03798X_4,$$

the decision boundary occurs when the above expression equated to zero and X_2 is isolated.

$$\therefore 0.1788 - 0.7464x_1 + 20.6922x_2 + 40.5827x_3 - 0.03798x_4 = 0$$

Since $x_3 = x_1^2$ and $x_4 = x_2^2$, we can sub these in and solve for the decision boundary.

$$\therefore 0.1788 - 0.7464x_1 + 40.5827x_1^2 + 20.6922x_2 - 0.03798x_2^2 = 0$$

Ignoring x_2^2 term since it has a very small coefficient gives

$$\therefore x_2 = (-0.1788/20.6922) + (0.7464/20.6922)x_1 - (40.5827/20.6922)x_1^2$$

Subbing in y = -1 and solving for the roots of the equation gives the range of X_1 values to plot the parabola with

$$\therefore (-0.1788/20.6922) + (0.7464/20.6922)x_1 - (40.5827/20.6922)x_1^2 = -1$$
$$\therefore [(-0.1788/20.6922) + 1] + (0.7464/20.6922)x_1 - (40.5827/20.6922)x_1^2 = 0$$

From which we find: $x_{1lower} = -0.7018$ and $x_{1upper} = 0.7202$. The resulting parabolic decision boundary is shown below in Figure 5.

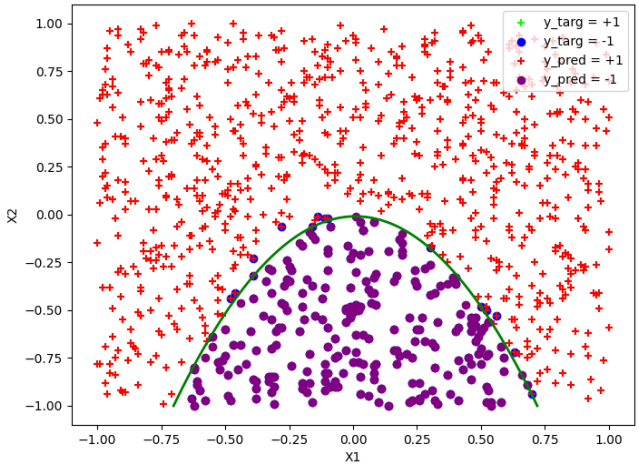

Figure 5: Scatter plot with parabolic decision boundary

References

All Python code used in this assignment was based on snippets taken from the first two weeks of lectures provided for this course. This includes the use of the sklearn toolkit, the numpy and pandas libraries, as well as plotting using matplotlib.

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

I consent / do not consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

I agree that this thesis will not be publicly available, but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement. **Please consult with your supervisor on this last item before agreeing, and delete if you do not consent**

Signed: Michael Millard Date: 05/10/2024

A Imports and Training Data Setup Appendix

This appendix contains the code that imports the relevant libraries and training data required for the assignment.

```
# ---------------------------------------------------------------------- #

# CS7CS4 Machine Learning
# Week 2 Assignment
#
# Name:           Michael Millard
# Student ID:     24364218
# Due date:       05/10/2024

# ---------------------------------------------------------------------- #

# Imports

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score

# ---------------------------------------------------------------------- #

# Read in data and set labels

# NB: dataset csv file must be in same directory as this solution
labels = ["X1", "X2", "y"]
df = pd.read_csv("ML_W2_Dataset.csv", names=labels)
print("Dataframe head:")
print(df.head())

# Split data frame up into X and y
X1 = df["X1"]
X2 = df["X2"]
X = np.column_stack((X1, X2))
y = df["y"]

# ---------------------------------------------------------------------- #
```

B Question (a) Appendix

This appendix contains the code used to answer question (a) of the assignment.

```
# (a)(i)

# Create scatter plot ('+' for +1 labels, '-' for -1 labels)
plt.figure(figsize=(8, 6))
plt.scatter(X1[y == 1], X2[y == 1], marker='+', color='lime', label='y = +1')
plt.scatter(X1[y == -1], X2[y == -1], marker='o', color='blue', label='y = -1')

# Label axes and add legend
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend(loc=1) # TR corner
```

```
# Save and show the plot
plt.savefig("scatter_lin_reg_a_i.png")
plt.show()

# ---------------------------------------------------------- #

# (a)(ii)

# Logistic regression model (no penalty)
model = LogisticRegression(penalty=None, solver='lbfgs')
model.fit(X, y)
theta0 = model.intercept_.item()
theta1, theta2 = model.coef_.T
theta1, theta2 = theta1.item(), theta2.item() # Convert from np.array to float
print("\nLogistic regression params:")
print("theta0, theta1, theta2 = %8f, %8f, %8f"%(theta0, theta1, theta2))

# ---------------------------------------------------------- #

# (a)(iii)

# Create scatter plot ('+' for +1 labels, '-' for -1 labels)
plt.figure(figsize=(8, 6))

# Actual target values
plt.scatter(X1[y == 1], X2[y == 1], marker='+', color='lime', label='y_targ = +1')
plt.scatter(X1[y == -1], X2[y == -1], marker='o', color='blue', label='y_targ = -1')

# Add the trained logistic regression model's predictions on the training data to the pl
y_pred = model.predict(X)

# Predicted values
plt.scatter(X1[y_pred == 1], X2[y_pred == 1], marker='+', color='red', label='y_pred = +
plt.scatter(X1[y_pred == -1], X2[y_pred == -1], marker='o', color='purple', label='y_pre

# Decision boundary x2 = m * x1 + c
m = -theta1 / theta2
c = -theta0 / theta2
y_db = m * X1 + c
plt.plot(X1, y_db, color='green', linewidth=2)

# Label axes and add legend
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend(loc=1)

# Save and show the plot
plt.savefig("scatter_db_lin_reg_a_iii.png")
plt.show()

# ---------------------------------------------------------- #
```

C Question (b) Appendix

This appendix contains the code used to answer question (b) of the assignment.

```
# (b)(i) and (ii)

print("\nSVM params for various penalty param (C) values:")
```

```
# Create subplots grid
num_rows, num_cols = 2, 3
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(12, 8), sharex=True, s
axes = axes.ravel() # Convert axes to 1D array

# Sweep through range of C values [0.001, 100], increasing C by 10x each iteration (6 it
# At each step, train the model with the new penalty parameter using the same training d
# Add subplot at each step
for i in range(0, 6):
    C_ = 0.001 * (10**i) # Underscore to differentiate between this var and LinearSVC pa
    model = LinearSVC(C=C_).fit(X, y)
    theta0 = model.intercept_.item()
    theta1, theta2 = model.coef_.T
    theta1, theta2 = theta1.item(), theta2.item()
    print("%1d. C, theta0, theta1, theta2 = %7.3f, %8f, %8f, %8f"%(i, C_, theta0, theta1

    axes[i].scatter(X1[y == 1], X2[y == 1], marker='+', color='lime', label='y_targ = +1
    axes[i].scatter(X1[y == -1], X2[y == -1], marker='o', color='blue', label='y_targ =

    # Add the trained logistic regression model's predictions on the training data to th
    y_pred = model.predict(X)
    axes[i].scatter(X1[y_pred == 1], X2[y_pred == 1], marker='+', color='red', label='y_
    axes[i].scatter(X1[y_pred == -1], X2[y_pred == -1], marker='o', color='purple', labe

    # Decision boundary x2 = m * x1 + c
    m = -theta1 / theta2
    c = -theta0 / theta2
    y_db = m * X1 + c
    axes[i].plot(X1, y_db, color='green', linewidth=2)

    # Label axes and add legend
    axes[i].title.set_text('C = %7.3f'%(C_))
    # Only set labels on bottom row for x-axis and left col for y-axis
    if (i / num_cols >= (num_rows - 1)):
        axes[i].set_xlabel('X1')
    if (i % num_cols == 0):
        axes[i].set_ylabel('X2')
    axes[i].legend(loc=1)

# Save and show the plot
plt.savefig("scatter_plots_svm_b_ii.png")
plt.show()

# ---------------------------------------------------------------------- #
```

D Question (c) Appendix

This appendix contains the code used to answer question (c) of the assignment.

```
# (c)(i)

print("\nNew logistic regression params (4 input features):")

# Create new input features and stack them
X3 = X1**2
X4 = X2**2
X = np.column_stack((X1, X2, X3, X4))

# Logistic Regression
model = LogisticRegression(penalty=None, solver='lbfgs')
```

```
model.fit(X, y)
theta0 = model.intercept_.item()
theta1, theta2, theta3, theta4 = model.coef_.T
theta1, theta2, theta3, theta4 = theta1.item(), theta2.item(), theta3.item(), theta4.iter
print("theta0, theta1, theta2, theta3, theta4 = %8f, %8f, %8f, %8f, %8f"%(theta0, theta1

# ---------------------------------------------------------- #

# (c)(ii)

# Create scatter plot ('+' for +1 labels, '-' for -1 labels)
plt.figure(figsize=(8, 6))
plt.scatter(X1[y == 1], X2[y == 1], marker='+', color='lime', label='y_targ = +1')
plt.scatter(X1[y == -1], X2[y == -1], marker='o', color='blue', label='y_targ = -1')

# Add the trained logistic regression model's predictions on the training data to the pl
y_pred = model.predict(X)
plt.scatter(X1[y_pred == 1], X2[y_pred == 1], marker='+', color='red', label='y_pred = +
plt.scatter(X1[y_pred == -1], X2[y_pred == -1], marker='o', color='purple', label='y_pre

# Label axes and add legend
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend(loc=1) # Upper right

# Save and show the plot
plt.savefig("scatter_new_lin_reg_c_ii.png")
plt.show()

# ---------------------------------------------------------- #

# (c)(iii)

# https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html
# Performance of most common class predictor
most_common_classifier = DummyClassifier(strategy="most_frequent")
most_common_classifier.fit(X, y)
y_base = most_common_classifier.predict(X)
print("\nMost common accuracy = %4.2f"%(accuracy_score(y, y_base)))

# Performance of logistic regression model
print("Logistic regression accuracy = %4.2f"%(accuracy_score(y, y_pred)))

# ---------------------------------------------------------- #

# (c)(iv)

# Create scatter plot ('+' for +1 labels, '-' for -1 labels)
plt.figure(figsize=(8, 6))
plt.scatter(X1[y == 1], X2[y == 1], marker='+', color='lime', label='y_targ = +1')
plt.scatter(X1[y == -1], X2[y == -1], marker='o', color='blue', label='y_targ = -1')

# Add the trained logistic regression model's predictions on the training data to the pl
plt.scatter(X1[y_pred == 1], X2[y_pred == 1], marker='+', color='red', label='y_pred = +
plt.scatter(X1[y_pred == -1], X2[y_pred == -1], marker='o', color='purple', label='y_pre

# Decision boundary 2: x2 = a*x1^2 + b*x1 + c
a = -theta3 / theta2
b = -theta1 / theta2
c = -theta0 / theta2
print("a = %8f, b = %6f, c = %8f"%(a, b, c))
```

```
# Find x_range
d = c + 1
x_lower = (-b + np.sqrt(b**2 - 4*a*d))/(2*a)
x_upper = (-b - np.sqrt(b**2 - 4*a*d))/(2*a)
x_lower, x_upper = x_lower.item(), x_upper.item()
print("x_lower = %8f, x_upper = %8f"%(x_lower, x_upper))

# Create linspaced points to plot decision boundary
X_range = np.linspace(x_lower, x_upper, 100)
y_db = a * X_range ** 2 + b * X_range + c
plt.plot(X_range, y_db, color='green', linewidth=2)

# Label axes and add legend
plt.xlabel('X1')
plt.ylabel('X2')
plt.legend(loc=1) # Upper right

# Save and show the plot
plt.savefig("scatter_db_new_lin_reg_c_iii.png")
plt.show()
```

————————————————————————————————

—— END OF ASSIGNMENT ——