Answers to Week2 Assignment

dataset # id:15--15-15

Question a

a.i

The scatter plot shows the data, where X_1 and X_2 are the two features from the data. There are two types of target values on the plot, where the green plus markers are +1, and the blue dot marker is -1.

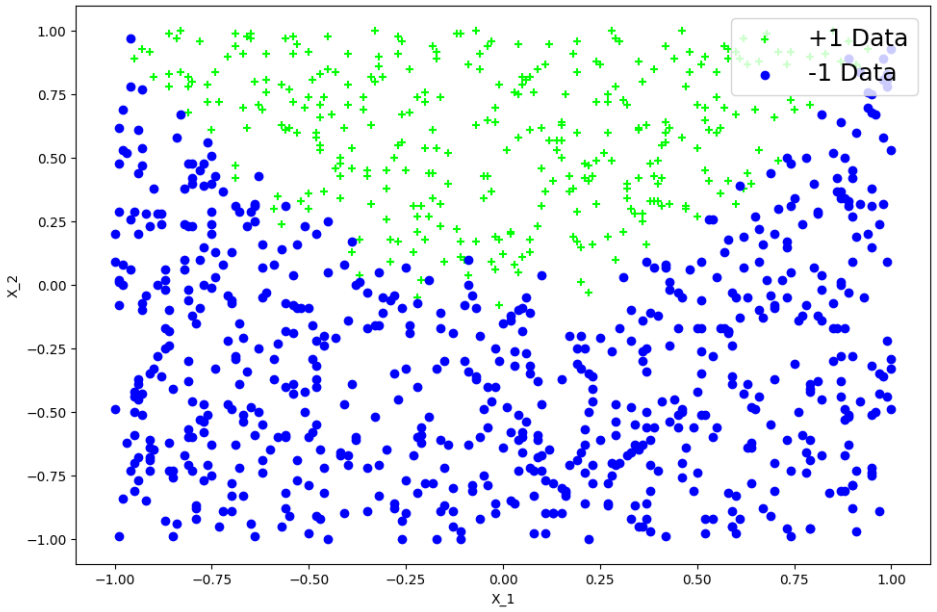

Figure 1: Scatter plot of data

a.ii

The data is split to 80% of training data and 20% of test data by using `train_test_split` function. Then the logistic regression model is trained on the training data with 2 features, where the linear model is $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$, and the trained parameter values are $\theta_0 = -1.51989916,\ \theta_1 = 0.02916994,\ \theta_2 = 4.92295755.$

Since θ_2 is larger than θ_1 then greater weight is placed by the model on feature x_2 than feature x_1, and since θ_2 is positive and large then increases in feature x_2 will cause output y to increase. Also, since θ_0 is negative negative then the output y is negative when $x_1 = 0, x_2 = 0$.

a.iii

The predicted data is based on the test data spited before. The plot shows the data, the predicted data, and decision boundary decided by the model. The red up triangle marker is +1 prediction, the orange down triangle marker is -1 prediction, and the black straight line is the decision boundary. There are some of red up triangles being misclassified into the area of -1 points(blue dot markers), and orange down triangles being misclassified into the area of +1 points(green plus markers).

The logistic regression mode is $y = -1.51989916 + 0.02916994x_1 + 4.92295755x_2$. When $y > 0$, the target value is +1, and when $y < 0$, the target value is -1. When $y = 0$, the equation becomes to $0 = 0.02916994x_1 + 4.92295755x_2 + -1.51989916$, which can be solved to the relationship between x_1 and x_2, and that is the decision boundary.

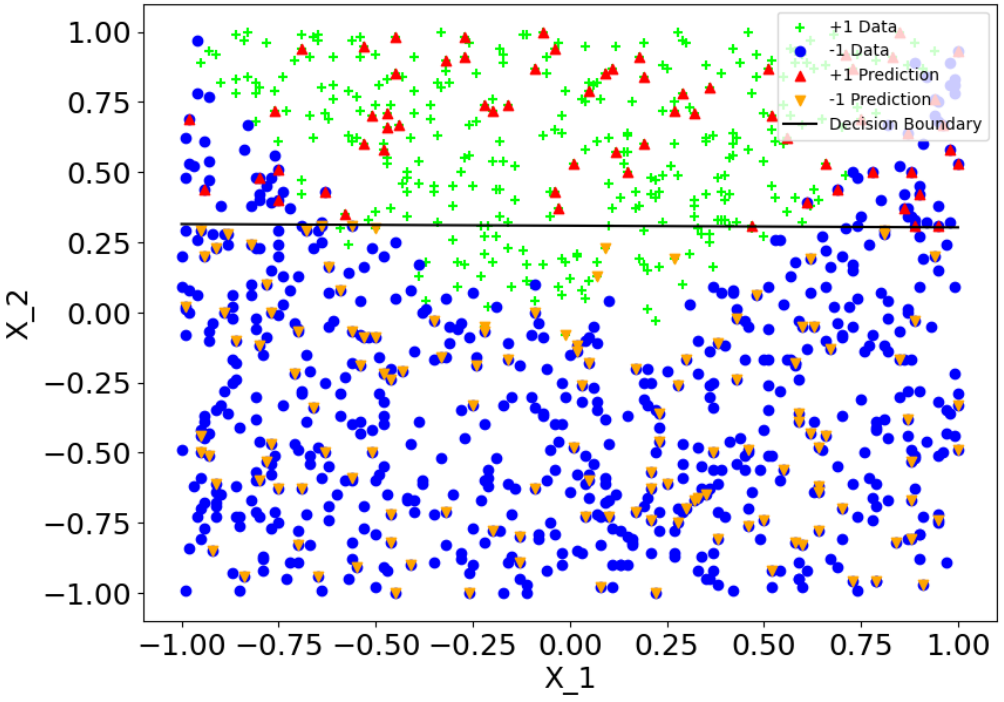

Figure 2: Plot of data and logistic regression prediction and decision boundary

a.iv

The accuracy of the model can be obtained by using the function `accuracy_score`, the accuracy score of this model on the test data is 0.87. It can be seen from the Figure 2 that there are blue dot markers above the decision boundary, and there are green plus markers below the decision boundary. These are all the points being misclassified.

Question b

b.i

1. With C at 0.001, the linear SVM model is expressed as
   $y = -0.22077764 + -0.0001033x_1 + 0.476492718x_2$.
   The weight of feature x_2 (0.476492718) is higher than that of feature x_1 (-0.0001033), indicating that x_2 has greater influences on the output of the model. Since the weight associated with feature x_1 is negative, it contributes negatively to the prediction, while x_2 is positive so it contributes positively to the prediction.

2. With C at 1, the linear SVM model is expressed as
   $y = -0.60378094 + -0.02286386x_1 + 1.79922921x_2$.
   The weight of feature x_2 (1.79922921) is higher than that of feature x_1 (-0.02286386), indicating that x_2 has greater influences on the output of the model. Since the weight associated with feature x_1 is negative, it contributes negatively to the prediction, while x_2 is positive so it contributes positively to the prediction.

3. With C at 100, the linear SVM model is expressed as
   $y = -0.61020377 + -0.02347584x_1 + 1.81675655x_2$
   The weight of feature x_2 (1.81675655) is higher than that of feature x_1 (-0.02347584), indicating that x_2 has greater influences on the output of the model. Since the weight associated with feature x_1 is negative, it contributes negatively to the prediction, while x_2 is positive so it contributes positively to the prediction.

b.ii

The following plots are the data, prediction by Linear SVM, and decision boundary with C equal to 0.01, 1, 100. As we can see, as the penalty increases, the area of green plus being misclassed is less and less as C increases, the decision boundary line goes down and down. Which means the predictions gets more accurate. But overall, there are misclassification in all three plots, there are red up triangle in the area of blue dot markers, and orange down triangle in the area of green plus markers.

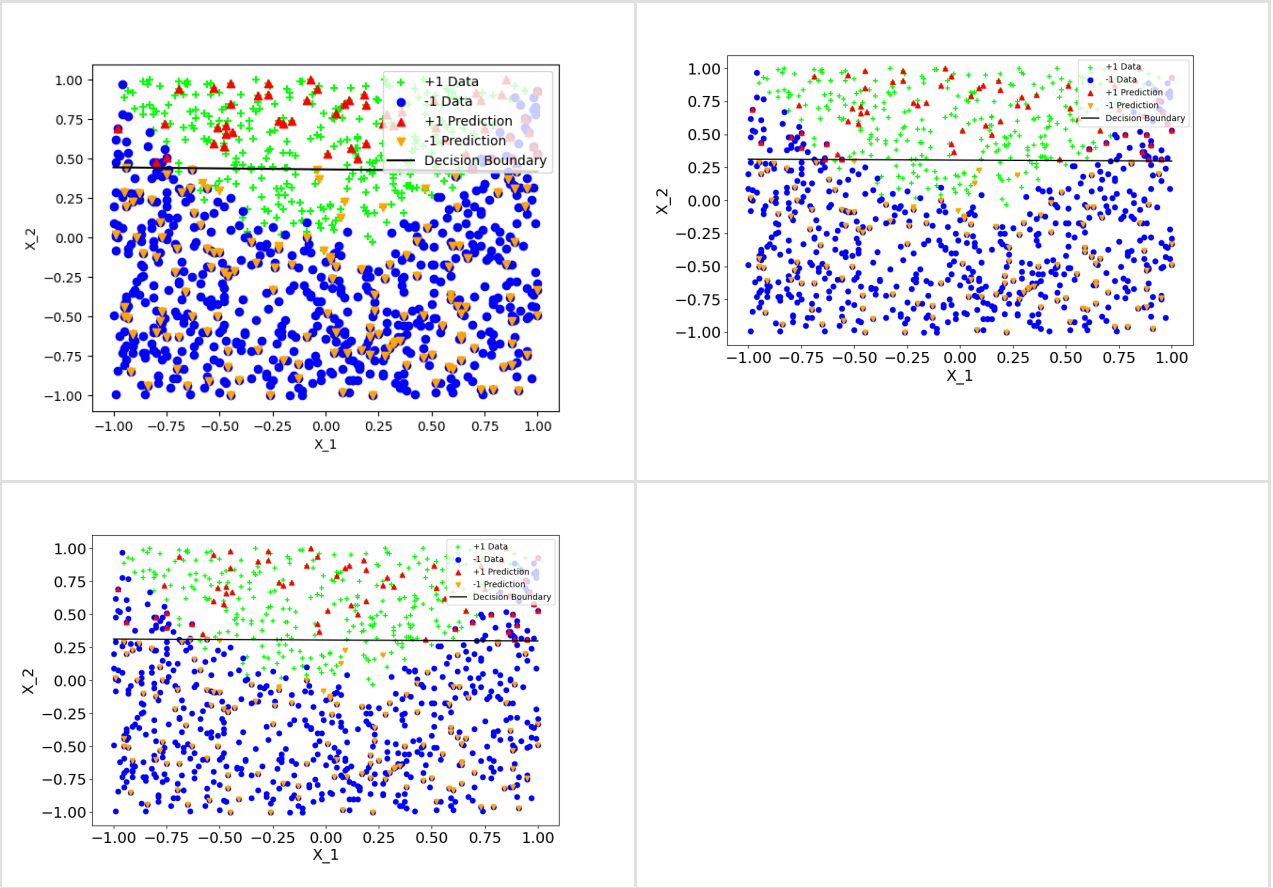

Figure 3: Plot of data and prediction and decision boundary with penalty at 0.001, 1 and 100

From cost function of SVM which is

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\max(0, 1 - y^{(i)}\theta^T x^{(i)}) + \theta^T\theta/C$$

It can be seen that as C increases, the regularization term $\theta^T\theta/C$ decreases. This means that the model will place more emphasis on minimizing the hinge loss $\max(0, 1 - y^{(i)}\theta^T x^{(i)})$, which encourages it to correctly classify all training data points as much as possible.

- Impact on model parameters:
  As C increases, the penalty for misclassifying becomes more significant. To avoid the penalties, the model adjusts by allowing the weights θ to become larger, thereby fitting the training data more closely.

- Impact on predictions:
  With higher C, the model becomes more rigid, trying to make fewer classification errors. This results in more accurate predictions but with the chance of overfitting.

b.iv

The accuracies of Linear SVM at 3 different C (0.001, 1, 100) are 0.89, 0.86, 0.87, which are quite same as the accuracy of logistic regression from question a.

Question C

c.i

The features are inserted with the squares of original two features x_1 and x_2. Then the logistic regression model can be expressed as

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2$$

with
$\theta_0 = -0.11426041, \theta_1 = -0.07595665, \theta_2 = 6.4928731, \theta_3 = -6.230209, \theta_4 = 0.66148383$

c.ii

The plot shows the data points and predictions by logistic regression model trained on 4 features. Almost all of the red up triangle markers locate in the area of green plus markers, so as most of the orange down triangle makers are in the area of blue dot markers. The predictions made by this model are significantly more accurate than previous models.

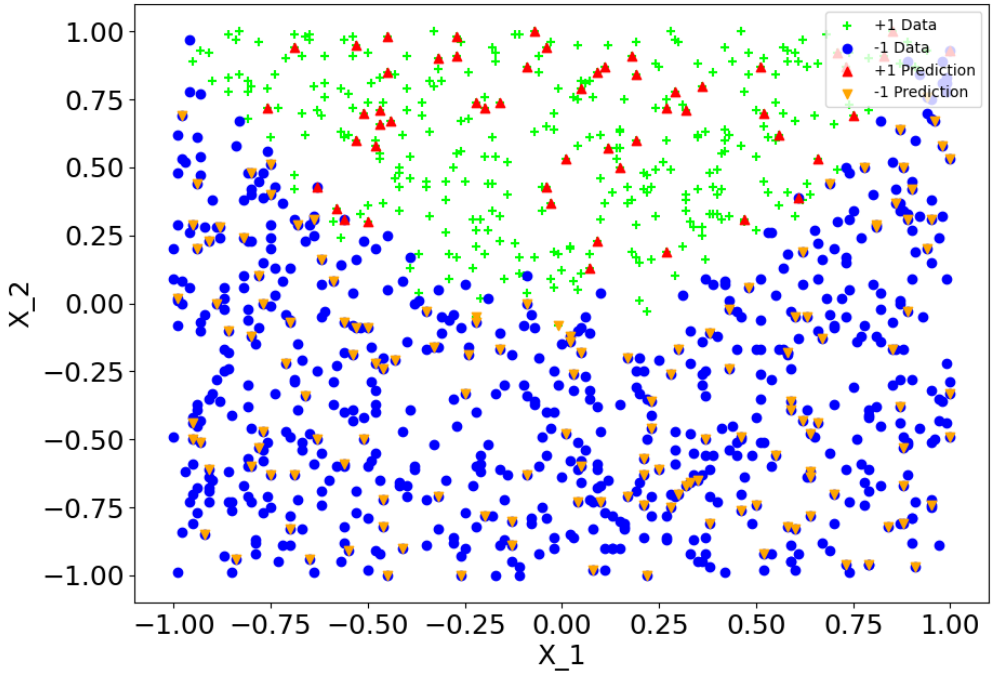

Figure 4: Plot of data and predictions

c.iii

The accuracy of this model is `0.97`, and that of a `DummyClassifier` trained on the 4 features is `0.74`. This logistic regression model is much more accurate than the dummy classifier, indicating that it successfully captures the relationship between features and target values. Which means the relationship between features and target values does involve the square of original features.

c.iv

The logistic regression prediction is:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2$$

and set y to 0 for the decision boundary, and rearrange the equation, it can be expressed as:

$$\theta_4 x_2^2 + \theta_2 x_2 = -(\theta_0 + \theta_1 x_1 + \theta_3 x_1^2)$$

And we see x_1 as known values, so we can apply the root equation to get the relationship between x_1 and x_2:

$$x_2 = \frac{-\theta_2 + -\sqrt{\theta_2^2 - 4\theta_4(\theta_0 + \theta_1 x_1 + \theta_3 x_1^2)}}{2\theta_4}$$

Then creating a set of points for x_1 at the ranges of (-1,1), and apply get the corresponding values of x_2

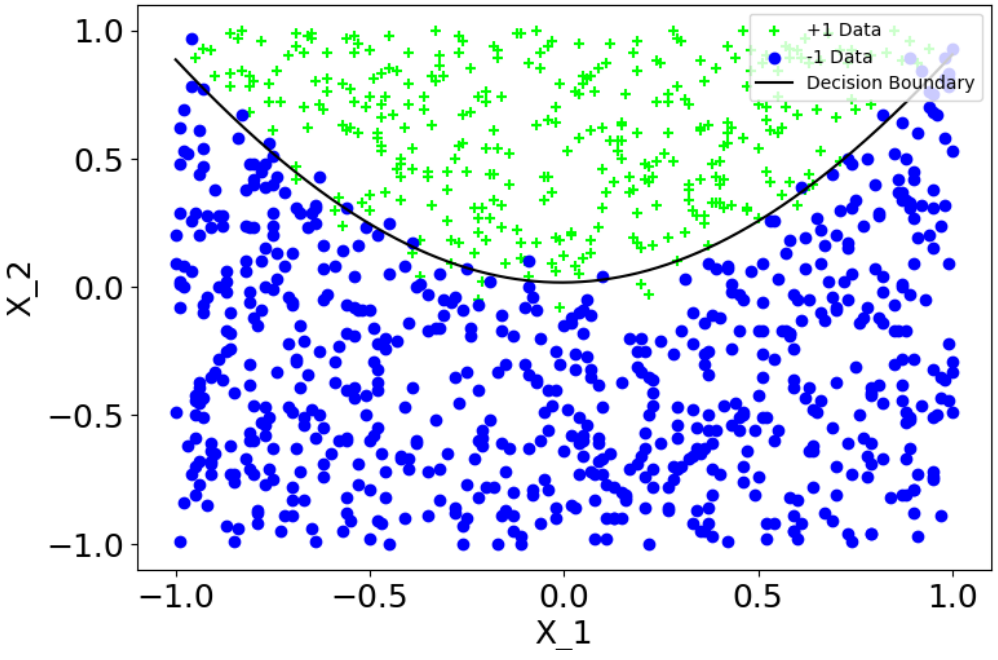

Figure 5: plot of decision boundary

Appendix

```
import matplotlib.pyplot as plt

# association function of plotting linear decision boundary of a model
def pltLinearBoundary(model):
    x1 = np.linspace(-1, 1, 100)
    # y = intercept + weight1*X1 + weight1*X2
    # the desicion boundary is the line where y = 0
    # 0 = intercept + weight1*X1 + weight2*X2
    # X2 = - (intercept + weight1*X1) / weight2
    plt.plot(x1, -(model.intercept_ + model.coef_[0]
[0]*x1)/model.coef_[0][1], color='black', label='Decision Boundary')

# association function of plotting predictions of a model
def pltPredictions(x_test, y_pred):
    predict_postive = x_test[y_pred == 1]
    predict_negative = x_test[y_pred == -1]
    plt.scatter(predict_postive[:, 0], predict_postive[:, 1],
color="red", marker='^', label='+1 Prediction')
    plt.scatter(predict_negative[:, 0], predict_negative[:, 1],
color="orange", marker='v', label='-1 Prediction')

import numpy as np
import pandas as pd
df = pd.read_csv("week2.csv",sep=',')
x1=df.iloc[:,0]
x2=df.iloc[:,1]
x=np.column_stack((x1,x2))
y=df.iloc[:,2]

x_with_positive_y = x[y == 1]
x_with_negative_y = x[y == -1]

# (a)
# (i) Plot the points
plt.scatter(x_with_positive_y[:, 0], x_with_positive_y[:, 1],
color='#00ff00', marker='+', label='+1 Data')
plt.scatter(x_with_negative_y[:, 0], x_with_negative_y[:, 1],
color='#0000fd', marker='o', label='-1 Data')
plt.rc('font', size=18)
plt.xlabel('X_1'); plt.ylabel('X_2')
plt.legend(loc='upper right')
plt.show()

# (ii) Train a logistic regression model and report the coefficients
```

```
and the intercept
from sklearn.model_selection import train_test_split
# split data into 80% train and 20% test data sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 1)
train_positive = x_train[y_train == 1]
train_negative = x_train[y_train == -1]

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
print("logisticRegression:")
print("coefficients: ", model.coef_[0])
print("intercept: ", model.intercept_)

# (iii)
# plot original data
plt.scatter(x_with_positive_y[:, 0], x_with_positive_y[:, 1],
color='#00ff00', marker='+', label='+1 Data')
plt.scatter(x_with_negative_y[:, 0], x_with_negative_y[:, 1],
color='#0000fd', marker='o', label='-1 Data')

# predict by logistic regression model
y_pred = model.predict(x_test)

# Plot the predictions of the logistic regression model
pltPredictions(x_test, y_pred)

# Plot the decision boundary of the logistic regression model
pltLinearBoundary(model)

plt.xlabel('X_1'); plt.ylabel('X_2')
plt.legend(loc='upper right', fontsize = 10)
plt.show()

# (iv) Report the accuracy
from sklearn.metrics import accuracy_score
print("Accuracy: ", accuracy_score(y_test, y_pred))


# (b)
# (i) Train LinearSVC model with different penalties
from sklearn.svm import LinearSVC
penalties = [0.001, 1, 100]
for p in penalties:
    # train LinearSVC with each penalty value on train data
    model = LinearSVC(C=p, dual='auto').fit(x_train, y_train)
```

```
    # report model parameters
    # predict on test data
    y_pred = model.predict(x_test)
    print("Penalty: ", p, " Coefficients: ", model.coef_, " Intercept:
", model.intercept_)
    print("Accuracy: ", accuracy_score(y_test, y_pred))

# (ii) plot predictions and training data together
    # plot original data
    plt.scatter(x_with_positive_y[:, 0], x_with_positive_y[:, 1],
color='#00ff00', marker='+', label='+1 Data')
    plt.scatter(x_with_negative_y[:, 0], x_with_negative_y[:, 1],
color='#0000fd', marker='o', label='-1 Data')

    # plot predictions
    pltPredictions(x_test, y_pred)

    # plot boundary
    pltLinearBoundary(model)
    plt.xlabel('X_1'); plt.ylabel('X_2')
    plt.legend(loc='upper right', fontsize = 10)
    plt.show()

# (c)
# (i)
# insert x1^2 x2^2 to train dataset
x1_train = x_train[:,0]
x2_train = x_train[:,1]
x_train_4_features = x_train
x_train_4_features = np.insert(x_train_4_features, 2, values =
x1_train*x1_train, axis = 1)
x_train_4_features = np.insert(x_train_4_features, 3, values =
x2_train*x2_train, axis = 1)

# train logistic regression model with 4 features
model = LogisticRegression()
model.fit(x_train_4_features, y_train)
print("LogisticRegression with 4 features:\nCoefficients: ",
model.coef_, " Intercept: ", model.intercept_)


# (ii)
# insert x1^2 x2^2 to test dataset
x1_test = x_test[:, 0]
x2_test = x_test[:, 1]
x_test_4_features = x_test
x_test_4_features = np.insert(x_test_4_features, 2, values =
```

```
x1_test*x1_test, axis = 1)
x_test_4_features = np.insert(x_test_4_features, 3, values =
x2_test*x2_test, axis = 1)

# plot original data
plt.scatter(x_with_positive_y[:, 0], x_with_positive_y[:, 1],
color='#00ff00', marker='+', label='+1 Data')
plt.scatter(x_with_negative_y[:, 0], x_with_negative_y[:, 1],
color='#0000fd', marker='o', label='-1 Data')

# predict on test data
y_pred = model.predict(x_test_4_features)
pltPredictions(x_test_4_features, y_pred)

plt.xlabel('X_1'); plt.ylabel('X_2')
plt.legend(loc='upper right', fontsize = 10)
plt.show()

# (iii) train dummy classifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
print("Logistic Regression on 4 Feature Accuracy: ",
accuracy_score(y_test, y_pred))
from sklearn.dummy import DummyClassifier
dummy =
DummyClassifier(strategy='most_frequent').fit(x_train_4_features,
y_train)
base_pred = dummy.predict(x_test_4_features)
print("Dummy Classifier Accuracy: ", accuracy_score(y_test, base_pred))

# (iv) draw the decision boundary
# plot original data
plt.scatter(x_with_positive_y[:, 0], x_with_positive_y[:, 1],
color='#00ff00', marker='+', label='+1 Data')
plt.scatter(x_with_negative_y[:, 0], x_with_negative_y[:, 1],
color='#0000fd', marker='o', label='-1 Data')

# plot curve decision boundary
coef = model.coef_[0]
intercept = model.intercept_[0]
x1_values = np.linspace(-1, 1, 10000)

# x2 equals to the solution of the quadratic equation with respect to
x1
discriminant = coef[1]**2 - 4 * coef[3] * (intercept + coef[0] *
x1_values + coef[2]  * x1_values**2)
x2_values = (-coef[1] + np.sqrt(discriminant)) / (2 * coef[3])
```

```
plt.plot(x1_values, x2_values, label='Decision Boundary',
color='black')
plt.xlabel('X_1'); plt.ylabel('X_2')
plt.legend(loc='upper right', fontsize = 10)
plt.show()
```