TRINITY COLLEGE DUBLIN SCHOOL OF COMPUTER SCIENCE AND STATISTICS

CS7CS4-20212022 MACHINE LEARNING

WEEKLY ASSIGNMENT

**MACHINE LEARNING-ASSIGNMENT WEEK 2**

REPORT

BY

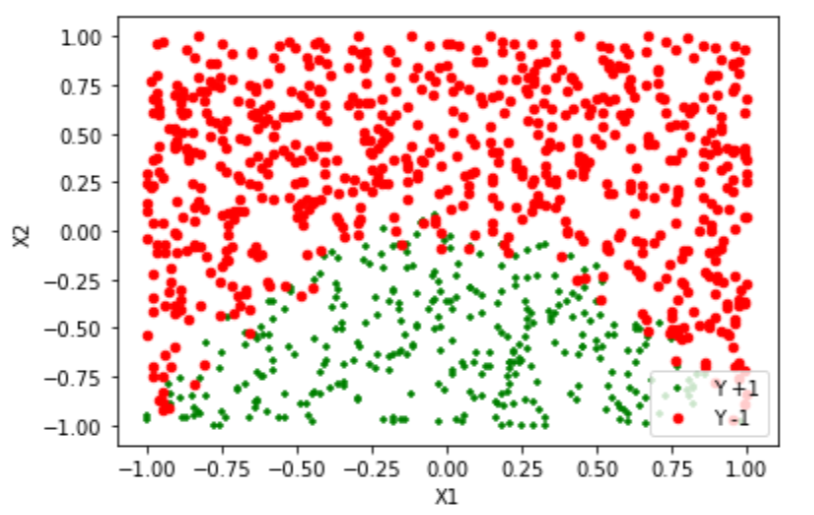
KAAVIYA KARUNANIDHI

21331515

# id:20--20--20

Model Solution:

1. (i) Figure 1 shows the 2D plot with the values of feature 1 (X1) in x-axis and the values of feature 2 (X2) in y-axis. The target(Y) values in the given dataset(week2.csv) is -1 and +1. A ‘+’ marker with green colour is used for values of ‘Y’ with +1 and a ‘o’ marker with red colour is used to mark the values of ‘Y’ with -1.



**Figure 1**

(ii) The data is trained using Logistic regression classifier where the target value is predicted based on statistical analysis of the given feature values (X = X\_train). Implementation of Logistic Regression classifier model is shown in Appendix.

Parameter values of the trained model are given as below:

**Coefficient**: [[-0.20494459 -5.25685516]]- Used to determine the slope of the model. Here the coefficients determine the weights of the feature values. The value of feature X1 impacts more on the target value Y.

**Intercept**: [-1.8091341]- Theta 0 indicates the intercept of the model with equation (theta\_0 + theta\_1 \* x1 + theta\_2 \* x2). In our model y = +1 when -1.8091341-0.20494459 -5.25685516x> 0 and y = −1 when -1.8091341-0.20494459 -5.25685516x< 0

**Score**: 1.0

**REPORT**:

precision recall f1-score support

-1 0.90 0.90 0.90 680

1 0.79 0.79 0.79 319

accuracy 0.86 999

macro avg 0.84 0.84 0.84 999

weighted avg 0.86 0.86 0.86 999

**Confusion Matrix:**

[[612 68]

[ 67 252]]

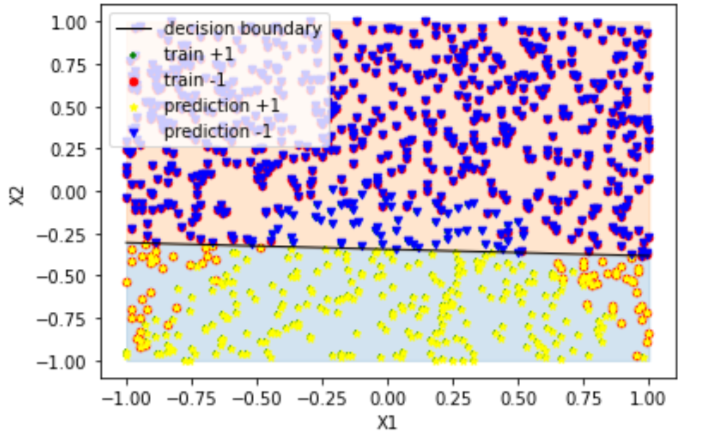
Confusion Matrix represents True positive, True negative, False Positive and False Negative. In our model we have 999 total number of datasets and the model has predicted 612 (True Positive) datasets correctly for y=+1 and 252(True Negative) datasets correctly for y=-1 and the model has predicted 68 (False Negative) datasets wrongly as y=-1 and 67 (False Positive) datasets wrongly as y=+1

(iii) Figure 2 shows the 2D plot generated for the predicted target values (y\_pred) and the original target values (Y) on the original feature/training feature set (X1,X2).

* Marker ‘+’ with green colour indicates the dataset with original target value (Y= +1)
* Marker ‘o’ with red colour indicates the dataset with original target value (Y= -1)
* Marker ‘\*’ with yellow colour indicates the dataset with predicted target value (y\_pred = +1)
* Marker ‘v’ with blue colour indicates the dataset with original target value (y\_pred = -1)

Decision boundary is shown in the figure in a linearly separable way with the line equation yd = m\*xd + c where m is the slope of the line which is obtained from the coefficients of the features (w1,w2) and c is the constant value which is calculated from the intercept of the logistic regression model.

Here the coefficients of the model is [-0.20494459 -5.25685516]. So the feature values X1 and X2 decreases the target value Y. But the feature X2 has the greater negative coefficient and decreases the Y to a great extent.



**Figure 2-Original and Trained data with decision boundary line**

(iv) Using the logistic regression classifier the data trained is of less accuracy when compared to the original data and the False positives and False negatives of the predicted value (y\_pred) can be clearly seen on the Figure 2.

1. (i) LinearSVC model is used on the dataset with wide range of penalty parameter C (C = 0.001,1,100)

When C= 1,

Score: 1.0

Coefficient: [[-0.15792521 -3.31487318]]

cnf\_Matrix: [[606 74]

[ 60 259]]

Accuracy: 0.8658658658658659

When C= 0.01,

Score: 1.0

Coefficient: [[-0.00097 -0.39645]]

cnf\_Matrix: [[677 3]

[274 45]]

Accuracy: 0.7227227227227228

When C= 100,

Score:1.0

Coefficient: [[-0.06644814 -3.48353988]]

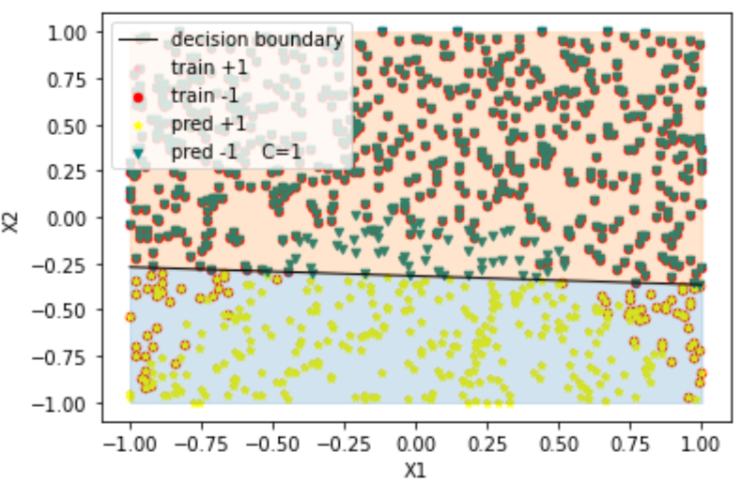
cnf\_matrix: [[605 75]

[ 63 256]]

Accuracy: 0.8618618618618619

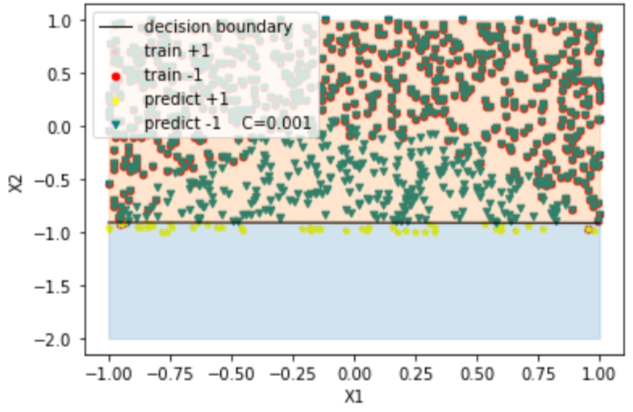
(ii) The scatter plots for the different penalty parameters are shown below:

When C=1:



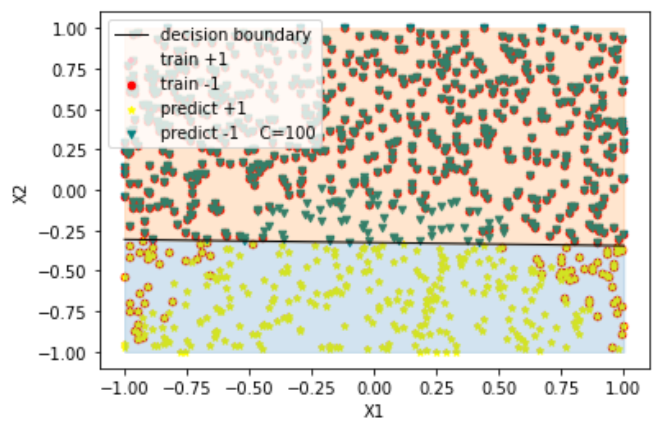
**Figure 3-LinearSVC Model with C=1**

When C=0.001:



**Figure 4-LinearSVC Model with C=0.001**

When C= 100



**Figure 5-LinearSVC Model with C=100**

(iii) Penalty Parameter C is used for SVM optimization which avoids the misclassification of training sample. When C is less, large-margin separating hyperplane will be chosen and the classification accuracy is less here. When C is large, small margin hyperplane is selected and the training points will be classified with more accuracy. But if C is very less, the data will be misclassified even if it is linearly separable. C is basically a regularisation parameter, which wheels the trade-off between realizing a small error on the training data and minimising the model of the weights

(iv) The model tries to finds the “best” margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data, while logistic regression does not, instead it can have different decision boundaries with different weights that are near the optimal point.

(c) (i) Two additional features (X3,X4) are calculated by adding the square of each features that is X3=(X1\*X1) and X4=(X2\*X2). The dataset is trained with logistic regression classifier. The model parameters are:

Score: 1.0

Coefficient: [[ -0.38708149 -12.25185303 -12.68042402 0.533444 ]]

Intercept: [0.04667256]

REPORT:: precision recall f1-score support

-1 0.97 0.97 0.97 680

1 0.93 0.93 0.93 319

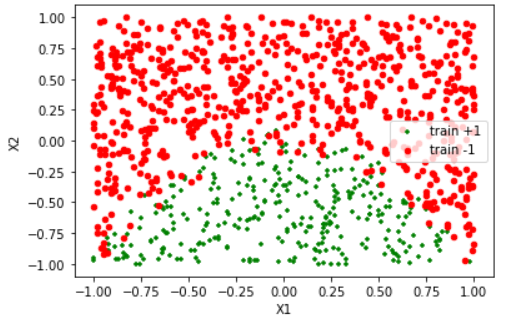
accuracy 0.96 999

macro avg 0.95 0.95 0.95 999

weighted avg 0.96 0.96 0.96 999

cnf\_matrix: [[657 23]

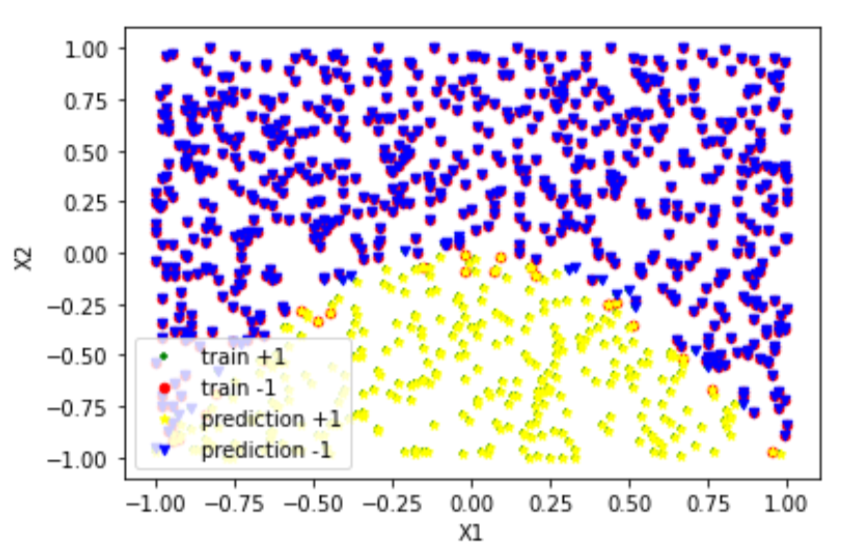
[ 21 298]]



**Figure** **6 - Original Data**

(ii) Figure 7 shows the scatter plot of actual(Y) and predicted (y\_test) target values for the feature X(X1,X2,X3,X4)

* Marker ‘+’ with green colour indicates the dataset with original target value (Y= +1)
* Marker ‘o’ with red colour indicates the dataset with original target value (Y= -1)
* Marker ‘\*’ with yellow colour indicates the dataset with predicted target value (y\_pred = +1)
* Marker ‘v’ with blue colour indicates the dataset with original target value (y\_pred = -1)



**Figure 7 – Original and Predicted Data**

Compared to Figure 2 in (a) the above Figure indicates that as we increase the number of feature values the accuracy and precision of the model gets increased and the decision boundary is non-linear whereas it is linear in the previous models.

(iii) A Dummy classifier is used as a baseline predictor to compare the performance of the model where the accuracy is 0.6786786786786787

APPENDIX:

(a)

import pandas as pd

import numpy as np

import sklearn

# %matplotlib inline

import matplotlib.pyplot as plt

df = pd.read\_csv("week2.csv",header=None)

df.head(1)

df.columns = ['X1','X2','Y']

df

df1=df[df['Y']==1]

df2=df[df['Y']==-1]

df1.head()

df['X'] = df.apply(lambda x: list([x['X1'],x['X2']]),axis=1)

ax = plt.gca()

df1.plot(kind='scatter',x='X1',y='X2',color='green',marker='+',ax=ax,label = 'Y +1')

df2.plot(kind='scatter',x='X1',y='X2',color='red',marker='o',ax=ax,label = 'Y -1')

plt.show()

from sklearn.datasets import load\_iris

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

from sklearn.metrics import classification\_report, confusion\_matrix

data\_frame = df.copy()

data\_frame.head()

cols = ['X1', 'X2']

X\_train = data\_frame[cols] # Features

y\_train = data\_frame.Y # Target variable

logreg = LogisticRegression(solver='liblinear',C=10.0,random\_state=0)

#X, y = load\_iris(return\_X\_y=True)

logreg.fit(X\_train,y\_train)

y\_pred=logreg.predict(X\_train)

report = classification\_report(y\_train, y\_pred)

#print(y\_pred)

print('Score:',logreg.score(X\_train,y\_pred))

print('Coefficient:',logreg.coef\_)

print('Intercept:',logreg.intercept\_)

print('REPORT:: ', report)

cnf\_matrix = metrics.confusion\_matrix(y\_train, y\_pred)

print('cnf\_matrix:',cnf\_matrix)

df\_y = pd.DataFrame(y\_pred)

df\_y.columns =['y\_pred']

df\_y

data\_frame = data\_frame.join(df\_y)

data\_frame

comparison\_column = np.where(data\_frame['Y'] == data\_frame["y\_pred"], True, False)

print(comparison\_column)

df3=data\_frame[data\_frame['y\_pred']==1]

df4=data\_frame[data\_frame['y\_pred']==-1]

import sklearn.linear\_model

b = logreg.intercept\_[0]

w1,w2 = logreg.coef\_.T

# Calculate the intercept and gradient of the decision boundary.

c= -b/w2

m= -w1/w2

xmin, xmax= -1, 1

ymin, ymax = -1, 1

xd = np.array([xmin, xmax])

yd = m\*xd + c

plt.plot(xd, yd, 'k', lw=1, ls='-', label='decision boundary')

plt.fill\_between( xd, yd, ymin, color='tab:blue', alpha=0.2)

plt.fill\_between( xd, yd, ymax, color='tab:orange', alpha=0.2)

data\_frame['X'] = data\_frame.apply(lambda x: list([x['X1'],x['X2']]),axis=1)

ax = plt.gca()

df1.plot(kind='scatter',x='X1',y='X2',color='green',marker='+',ax=ax, label='train +1')

df2.plot(kind='scatter',x='X1',y='X2',color='red',marker='o',ax=ax,label = 'train -1')

df3.plot(kind='scatter',x='X1',y='X2',color='yellow',marker='\*',ax=ax, label='prediction +1')

df4.plot(kind='scatter',x='X1',y='X2',color='blue',marker='v',ax=ax, label='prediction -1')

ax.legend(loc='upper left')

plt.show()

(b)

import numpy as np

from matplotlib import pyplot as plt

import pandas as pd

import io

# %matplotlib inline

import sklearn

df = pd.read\_csv("week2.csv",header=None)

print(df)

df.head(1)

df.columns = ['X1','X2','Y']

col = ['X1','X2']

from sklearn.svm import LinearSVC

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.datasets import make\_classification

from sklearn import metrics

from sklearn.metrics import classification\_report, confusion\_matrix

X = df[col]

y= df.Y

# Step 2: Get data

data\_fr= df.copy()

columns = ['X1','X2']

x\_training = data\_fr[columns]

y\_training = data\_fr.Y

# Step 3: Create a model and train it

clf = LinearSVC(C=1, loss="hinge", random\_state=42).fit(x\_training, y\_training)

y\_pred = clf.predict(x\_training)

score\_ = clf.score(x\_training, y\_pred)

#report = classification\_report(y\_training, y\_pred)

print(y\_pred)

print(score\_)

print(clf.coef\_)

cnf\_matrix = metrics.confusion\_matrix(y\_training, y\_pred)

print('cnf\_matrix:',cnf\_matrix)

print("Accuracy:",metrics.accuracy\_score(y\_training, y\_pred))

#linsvc = LinearSVC(C=1)

#linsvc.fit(X,y)

df\_ypred = pd.DataFrame(y\_pred)

df\_ypred.columns = ['y\_pred']

data\_fr = data\_fr.join(df\_ypred)

print(data\_fr)

df1=df[df['Y']==1]

df2=df[df['Y']==-1]

df3=data\_fr[data\_fr['y\_pred']==1]

df4=data\_fr[data\_fr['y\_pred']==-1]

data\_fr['X'] = data\_fr.apply(lambda x: list([x['X1'],x['X2']]),axis=1)

ax = plt.gca()

df1.plot(kind='scatter', x='X1', y = 'X2', color='pink',marker='+',ax=ax, label='train +1')

df2.plot(kind='scatter', x='X1', y = 'X2', color='red',marker='o',ax=ax, label='train -1')

df3.plot(kind='scatter', x='X1', y = 'X2', color='yellow',marker='\*',ax=ax, label='pred +1')

df4.plot(kind='scatter', x='X1', y = 'X2', color='teal',marker='v',ax=ax, label='pred -1 C=1')

ax.legend(loc='upper left');

#Retrieve the model parameters.

b = clf.intercept\_[0]

w1,w2 = clf.coef\_.T

# Calculate the intercept and gradient of the decision boundary.

c= -b/w2

m= -w1/w2

#Plot the data and the classification with the decision boundary.

xmin, xmax= -1, 1

ymin, ymax = -1, 1

xd = np.array([xmin, xmax])

yd = m\*xd + c

plt.plot(xd, yd, 'k', lw=1, ls='-', label='decision boundary')

plt.fill\_between( xd, yd, ymin, color='tab:blue', alpha=0.2)

plt.fill\_between( xd, yd, ymax, color='tab:orange', alpha=0.2)

ax.legend(loc='upper left');

plt.show()

from sklearn.svm import LinearSVC

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.datasets import make\_classification

X = df[col]

y= df.Y

# Step 2: Get data

data\_fr= df.copy()

columns = ['X1','X2']

x\_training = data\_fr[columns]

y\_training = data\_fr.Y

# Step 3: Create a model and train it

#X, y = make\_classification(n\_features=4, random\_state=0)

#clf = make\_pipeline(StandardScaler(),

#LinearSVC(C=0.001))

#clf.fit(x\_training, y\_training)

clf = LinearSVC(C=0.001, loss="hinge", random\_state=42).fit(x\_training, y\_training)

y\_pred = clf.predict(x\_training)

score\_ = clf.score(x\_training, y\_pred)

print(y\_pred)

print(score\_)

print(clf.coef\_)

cnf\_matrix = metrics.confusion\_matrix(y\_training, y\_pred)

print('cnf\_matrix:',cnf\_matrix)

print("Accuracy:",metrics.accuracy\_score(y\_training, y\_pred))

df\_ypred = pd.DataFrame(y\_pred)

df\_ypred.columns = ['y\_pred']

data\_fr = data\_fr.join(df\_ypred)

df1=df[df['Y']==1]

df2=df[df['Y']==-1]

df3=data\_fr[data\_fr['y\_pred']==1]

df4=data\_fr[data\_fr['y\_pred']==-1]

data\_fr['X'] = data\_fr.apply(lambda x: list([x['X1'],x['X2']]),axis=1)

ax = plt.gca()

df1.plot(kind='scatter', x='X1', y = 'X2', color='pink',marker='+',ax=ax, label='train +1')

df2.plot(kind='scatter', x='X1', y = 'X2', color='red',marker='o',ax=ax, label='train -1')

df3.plot(kind='scatter', x='X1', y = 'X2', color='yellow',marker='\*',ax=ax, label='predict +1')

df4.plot(kind='scatter', x='X1', y = 'X2', color='teal',marker='v',ax=ax, label='predict -1 C=0.001')

ax.legend(loc='upper left');

#Retrieve the model parameters.

b = clf.intercept\_[0]

w1,w2 = clf.coef\_.T

# Calculate the intercept and gradient of the decision boundary.

c= -b/w2

m= -w1/w2

#Plot the data and the classification with the decision boundary.

xmin, xmax= -1, 1

ymin, ymax = -2, 1

xd = np.array([xmin, xmax])

yd = m\*xd + c

plt.plot(xd, yd, 'k', lw=1, ls='-', label='decision boundary')

plt.fill\_between( xd, yd, ymin, color='tab:blue', alpha=0.2)

plt.fill\_between( xd, yd, ymax, color='tab:orange', alpha=0.2)

ax.legend(loc='upper left');

plt.show()

from sklearn.svm import LinearSVC

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.datasets import make\_classification

X = df[col]

y= df.Y

# Step 2: Get data

data\_fr= df.copy()

columns = ['X1','X2']

x\_training = data\_fr[columns]

y\_training = data\_fr.Y

# Step 3: Create a model and train it

#X, y = make\_classification(n\_features=4, random\_state=0)

clf = LinearSVC(C=100, loss="hinge", random\_state=42).fit(x\_training, y\_training)

y\_pred = clf.predict(x\_training)

score\_ = clf.score(x\_training, y\_pred)

print(y\_pred)

print(score\_)

print(clf.coef\_)

cnf\_matrix = metrics.confusion\_matrix(y\_training, y\_pred)

print('cnf\_matrix:',cnf\_matrix)

print("Accuracy:",metrics.accuracy\_score(y\_training, y\_pred))

df\_ypred = pd.DataFrame(y\_pred)

df\_ypred.columns = ['y\_pred']

data\_fr = data\_fr.join(df\_ypred)

df1=df[df['Y']==1]

df2=df[df['Y']==-1]

df3=data\_fr[data\_fr['y\_pred']==1]

df4=data\_fr[data\_fr['y\_pred']==-1]

data\_fr['X'] = data\_fr.apply(lambda x: list([x['X1'],x['X2']]),axis=1)

ax = plt.gca()

df1.plot(kind='scatter', x='X1', y = 'X2', color='pink',marker='+',ax=ax, label='train +1')

df2.plot(kind='scatter', x='X1', y = 'X2', color='red',marker='o',ax=ax, label='train -1')

df3.plot(kind='scatter', x='X1', y = 'X2', color='yellow',marker='\*',ax=ax, label='predict +1')

df4.plot(kind='scatter', x='X1', y = 'X2', color='teal',marker='v',ax=ax, label='predict -1 C=100')

ax.legend(loc='upper left');

#Retrieve the model parameters.

b = clf.intercept\_[0]

w1,w2 = clf.coef\_.T

# Calculate the intercept and gradient of the decision boundary.

c= -b/w2

m= -w1/w2

#Plot the data and the classification with the decision boundary.

xmin, xmax= -1, 1

ymin, ymax = -1, 1

xd = np.array([xmin, xmax])

yd = m\*xd + c

plt.plot(xd, yd, 'k', lw=1, ls='-', label='decision boundary')

plt.fill\_between( xd, yd, ymin, color='tab:blue', alpha=0.2)

plt.fill\_between( xd, yd, ymax, color='tab:orange', alpha=0.2)

ax.legend(loc='upper left');

plt.show()

(c)

import pandas as pd

import numpy as np

import sklearn

# %matplotlib inline

import matplotlib.pyplot as plt

df = pd.read\_csv("week2.csv",header=None)

df.columns = ['X1','X2','Y']

df['X3'] = (df.X1\*df.X1)

df['X4'] = (df.X2\*df.X2)

df = df[["X1", "X2", "X3","X4","Y"]]

from sklearn.datasets import load\_iris

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report, confusion\_matrix

data\_frame = df.copy()

data\_frame.head()

cols = ['X1', 'X2','X3','X4']

X\_train = data\_frame[cols] # Features

y\_train = data\_frame.Y # Target variable

logreg = LogisticRegression(solver='liblinear',C=10.0,random\_state=0)

#X, y = load\_iris(return\_X\_y=True)

logreg.fit(X\_train,y\_train)

y\_pred=logreg.predict(X\_train)

report = classification\_report(y\_train, y\_pred)

print(y\_pred)

print('Score:',logreg.score(X\_train,y\_pred))

print('Coefficient:',logreg.coef\_)

print('Intercept:',logreg.intercept\_)

print('REPORT:: ', report)

cnf\_matrix = metrics.confusion\_matrix(y\_train, y\_pred)

print('cnf\_matrix:',cnf\_matrix)

df\_y = pd.DataFrame(y\_pred)

df\_y.columns =['y\_pred']

df\_y

data\_frame = data\_frame.join(df\_y)

data\_frame

df1=df[df['Y']==1]

df2=df[df['Y']==-1]

ax = plt.gca()

df1.plot(kind='scatter',x='X1',y='X2',color='green',marker='+',ax=ax,label = 'train +1')

df2.plot(kind='scatter',x='X1',y='X2',color='red',marker='o',ax=ax,label = 'train -1')

df3=data\_frame[data\_frame['y\_pred']==1]

df4=data\_frame[data\_frame['y\_pred']==-1]

ax = plt.gca()

df1.plot(kind='scatter',x='X1',y='X2',color='green',marker='+',ax=ax,label = 'train +1')

df2.plot(kind='scatter',x='X1',y='X2',color='red',marker='o',ax=ax,label = 'train -1')

df3.plot(kind='scatter',x='X1',y='X2',color='yellow',marker='\*',ax=ax,label = 'prediction +1')

df4.plot(kind='scatter',x='X1',y='X2',color='blue',marker='v',ax=ax,label = 'prediction -1')

plt.show()

#Baseline Predictor

import statistics

statistics.mode(y\_train)

from sklearn.metrics import accuracy\_score

from sklearn.metrics import accuracy\_score

guess = statistics.mode(y\_train)

y\_pred\_Base = [guess] \* len(y\_pred)

score = accuracy\_score(y\_train,y\_pred\_Base)

print (f'Baseline accuracy score {round(score\*100,0)}%')

print (f'Baseline prediction {guess}')

from sklearn.dummy import DummyClassifier

dummy\_clfr = DummyClassifier(strategy="most\_frequent")

dummy\_clfr.fit(X\_train, y\_pred)

DummyClassifier()

dummy\_clfr.predict(X\_train)

dummy\_clfr.score(X\_train, y\_pred)