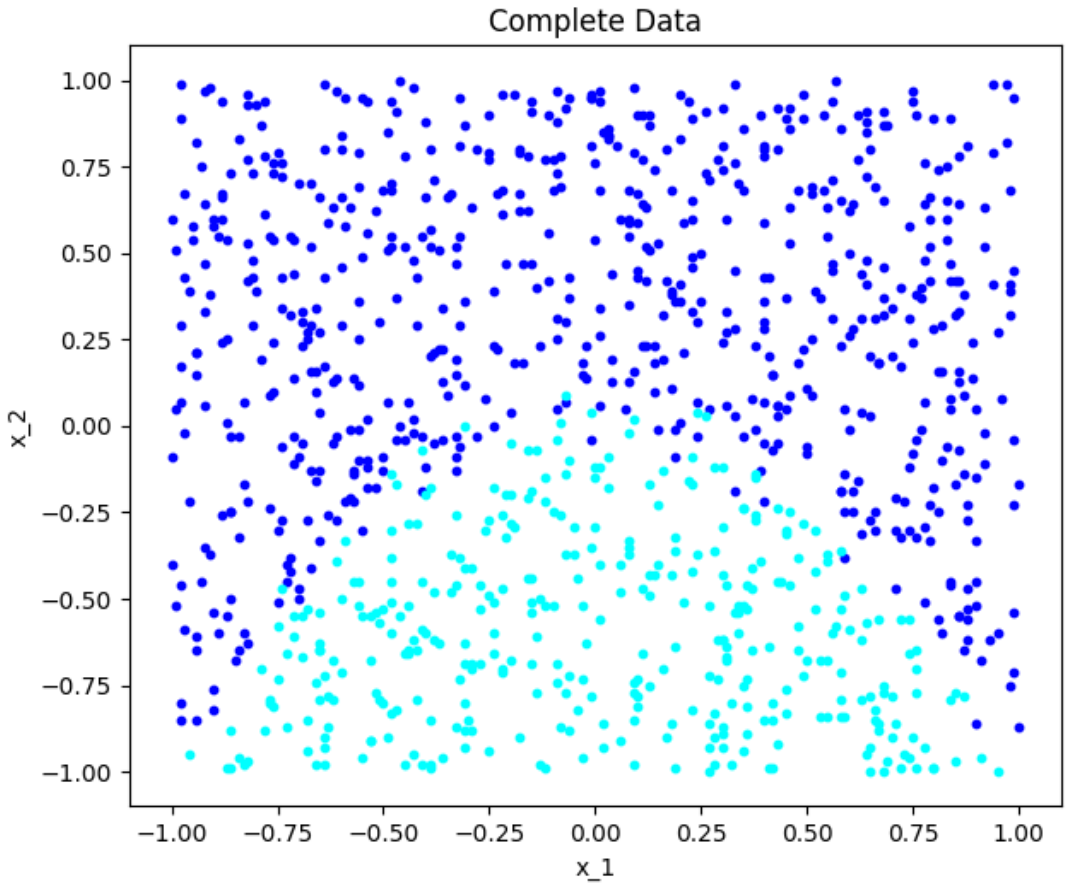
Machine Learning Assignment Week 2

A 1) I had to change the import code snippet to put my needs but otherwise this was an easy question. Instead of using X and O markers I decided to go with colours as it looked less cluttered to me.



In the above plot the +1 values are in blue and the -1 values are in light blue. This was done with matplotlib as were all of the following graphs

A 2)

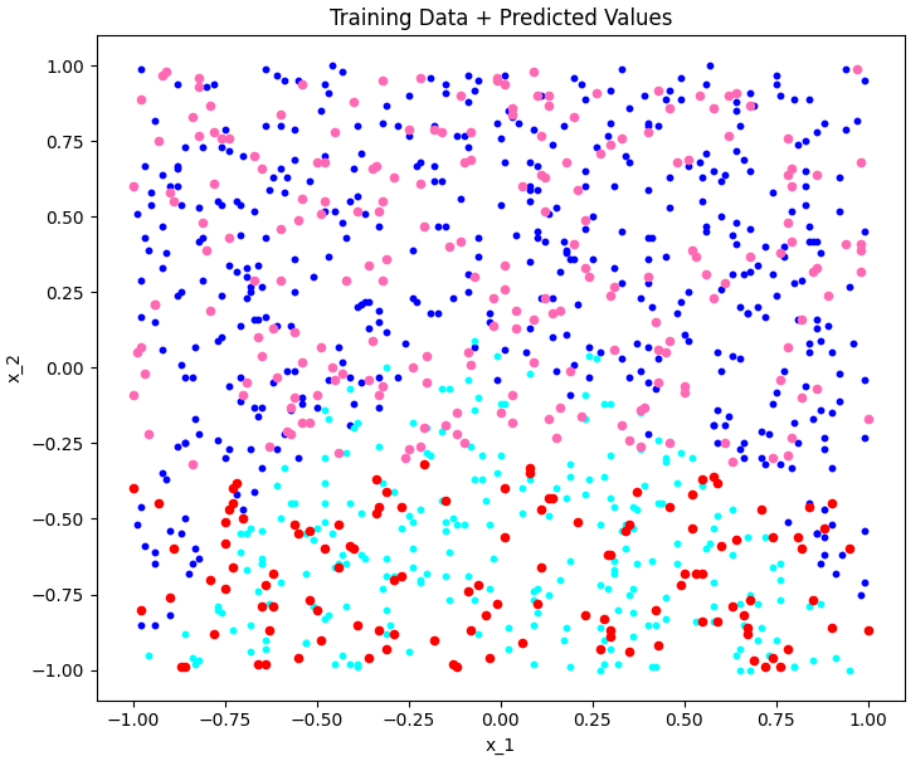
I used the basic LogisticRegression function and used random state = 0. My data was split into train and test with a 2/3 train and 1/3 test ratio using the sklearn.model_selection library.

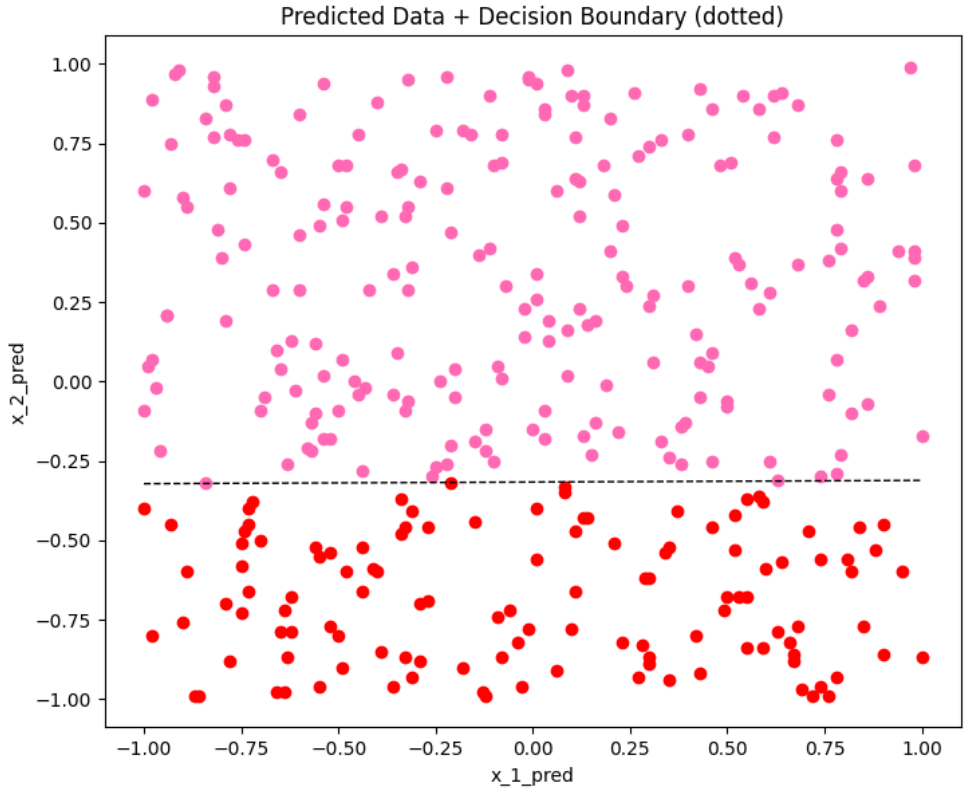
Here are the parameters of my model:

Parameter	Value
C	1.0
class_weight	None
Dual	False
fit_intercept	True
intercept_scaling	1
l1_ratio	None
max_iter	100
multi_class	Auto
n_jobs	None
Penalty	12
random_state	0
Solver	Lbfgs
Tol	0.0001
Verbose	0
warm_start	False

A 3)

This is a plot of the predicted data and the original data.
The original data is in blue and lightblue like in the previous question. The predicted values are in pink and red. +1 predicted values are in pink and the -1 predicted values are in red.
As one can see the model does a fairly good job of predicting values within the major parts of the data distribution however fails as it approaches the boarders





In the above graph we have the predicted values illustrated along with the decision boundary. I decided to remove the training data from this picture to clearly highlight how the decision boundary influences predicted values. It influences them by having classified values fall clearly on either side of the boundary or very close to it. You can clearly see how there are no predicted positive values on the lower side and vice versa. This is the line used by the algorithm as a lookup for each new point to see where it falls and what it is then classified as.

A 4)

The predictions alongside the training data looks to be quite accurate. Clearly there is a quadratic shape to the separation between the initial data so a linear classifier like this one will have a hard time making a perfect classification model. The model does deteriorate at very large or very small X1 values as well as struggle to classify around the “tip” of the -1 values because they jot out into the +1 values. The model struggles with this.

B 1) Each box here represents a different model. The parameters of each model are contained within.

Parameter	Value
C	1
class_weight	None
dual	True
fit_intercept	True
intercept_scaling	1
loss	squared_hinge
max_iter	1000
multi_class	ovr
penalty	l2
random_state	None
tol	0.0001
verbose	0
ACCURACY =	0.8636363636363636

Parameter	Value
C	100
class_weight	None
dual	True
fit_intercept	True
intercept_scaling	1
loss	squared_hinge
max_iter	1000
multi_class	ovr
penalty	l2
random_state	None
tol	0.0001
verbose	0
ACCURACY =	0.8666666666666667

Parameter	Value
C	1000
class_weight	None
dual	True
fit_intercept	True
intercept_scaling	1
loss	squared_hinge
max_iter	1000
multi_class	ovr
penalty	l2
random_state	None
tol	0.0001
verbose	0
ACCURACY =	0.7909090909090909

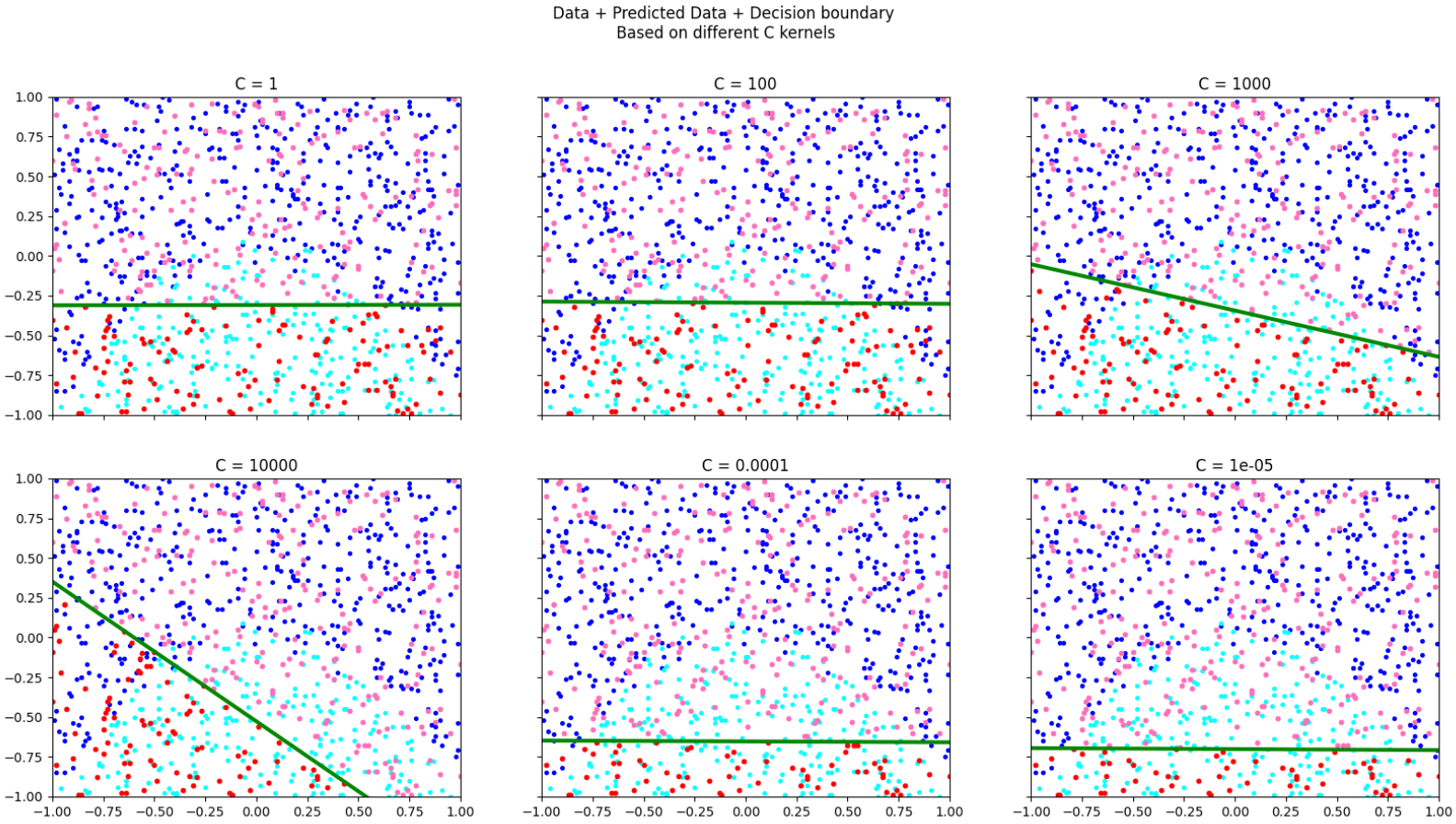
Parameter	Value
C	10000
class_weight	None
dual	True
fit_intercept	True
intercept_scaling	1
loss	squared_hinge
max_iter	1000
multi_class	ovr
penalty	l2
random_state	None
tol	0.0001
verbose	0
ACCURACY =	0.8636363636363636

Parameter	Value
C	0.0001
class_weight	None
dual	True
fit_intercept	True
intercept_scaling	1
loss	squared_hinge
max_iter	1000
multi_class	ovr
penalty	l2
random_state	None
tol	0.0001
verbose	0
ACCURACY =	0.806060606060606

Parameter	Value
C	0.00001
class_weight	None
dual	True
fit_intercept	True
intercept_scaling	1
loss	squared_hinge
max_iter	1000
multi_class	ovr
penalty	l2
random_state	None
tol	0.0001
verbose	0
ACCURACY =	0.7818181818181819

B 2) Below you can see the the totality of all data plotted:

- Each Subplot is for a different C value.
- Blue values are real data. Blue is +1, baby blue is -1.
- Red values are predicted data. Pink is +1, red is -1.
- The green line is the decision boundary of the different SVM models. All other parameters are kept the same as seen above.



B 3)

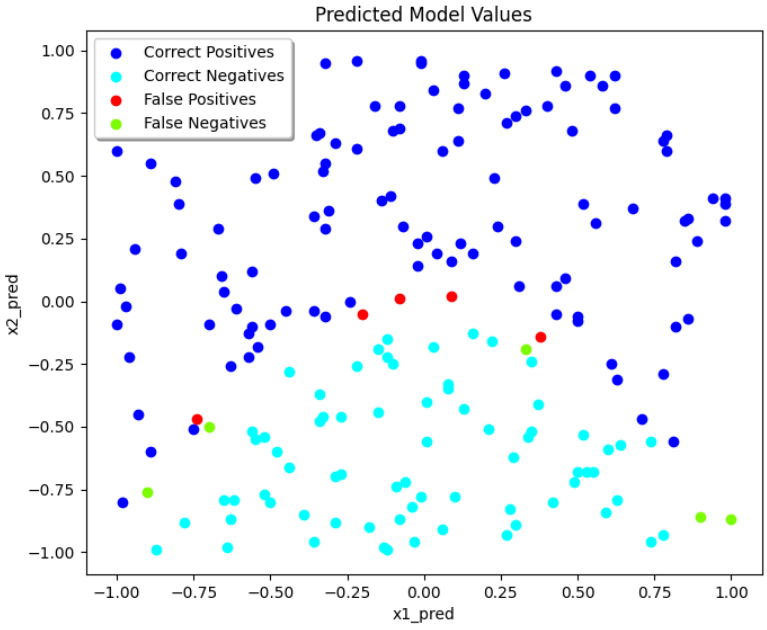
“The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.”

https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

C 1) Parameters of the logistic regression classifier:

Parameter	Value
C	1
class_weight	None
dual	False
fit_intercept	True
intercept_scaling	1
l1_ratio	None
max_iter	100
multi_class	auto
n_jobs	None
Penalty	L2
random_state	0
Solver	lbfgs
Tol	0.0002
Verbose	0
warm_start	False
ACCURACY =	0.9666666666666667

C 2)



Above we see the plot containing data plotted by the model based on test values which it had not seen before. As we can see it performed quite well by misclassifying only 10 values around the outer edges of both distributions. I purposefully did not plot training data as not to crowd actual results of the model. I added the squared features in the excel sheet directly so I did not have to do it programmatically. The excel sheet is attached.

C 3) For the baseline predictor I used a dummy classifier that predicts the most common class. It had an accuracy of 0.625. Comparing this with the accuracy of logistic regression classifier which had an accuracy of ~0.95 or 95% the log-reg-classifier is much more accurate because it correct much more often and does not restrict itself to a single class.

FIRST WE TRAIN OUR CLASSIFIERS

```
clf = LogisticRegression(random_state=0).fit(X_train, y_train)
predictions = clf.predict(X_test)

dummy_predictions = dummy_clf.predict(X_test)
dummy_clf = DummyClassifier(strategy="most_frequent").fit(X_train, y_train)


true_neg, false_pos, false_neg, true_pos = confusion_matrix(y_list_ok, predictions).ravel()
conf_matrix = confusion_matrix(y_list_ok, predictions)
accuracy = accuracy_score(y_list_ok, predictions)
```

PRINT THE RESULTS

```
print(true_neg, false_pos, false_neg, true_pos)
print("conf_matrix: \n", conf_matrix)
print("accuracy score: ", accuracy)


true_neg, false_pos, false_neg, true_pos = confusion_matrix(y_list_ok,
dummy_predictions).ravel()
conf_matrix = confusion_matrix(y_list_ok, dummy_predictions)
accuracy = accuracy_score(y_list_ok, dummy_predictions)
```

PRINT THE RESULTS

```
print(true_neg, false_pos, false_neg, true_pos)
print("conf_matrix: \n", conf_matrix)
print("accuracy score: ", accuracy)
```

We then compare by looking at the different performances of the models.

CODE FOR PART A)

```
# id:19-19-19
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

df = pd.read_csv("data.csv",header=None, comment="#",
names=["X1","X2","X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]
X_train, X_test, y_train, y_test = train_test_split(X_all, X_labels,
test_size=0.33, random_state=42)

clf = LogisticRegression(random_state=0).fit(X_train, y_train)
params = clf.get_params()
print(params)
b = clf.intercept_[0]
w1, w2 = clf.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

#Seperating 1 and -1 values into Xpositive and Xnegative
Xp = df.loc[df["X3"]==1]
Xm = df.loc[df["X3"]==-1]
Xp_labels = Xp["X3"]
Xm_labels = Xm["X3"]
```

```
predictions = clf.predict(X_test)

#splitting the data
X_predictions = X_test
X_predictions["X3"] = predictions

#splitting the data
Xp_predictions = X_predictions.loc[X_predictions["X3"]==1]
Xm_predictions = X_predictions.loc[X_predictions["X3"]==-1]
Xp_pred_labels = X_predictions["X3"]
Xm_pred_labels = X_predictions["X3"]

#plotting the data
fig, (ax1, ax2, ax3) = plt.subplots(3)
fig.tight_layout(pad=1.0)
ax1.title("Complete Data")
ax1.ylabel("x_2")
ax1.xlabel("x_1")
ax1.scatter(Xp["X1"],Xp["X2"], color='b', s = 10)
ax1.scatter(Xm["X1"],Xm["X2"], color='aqua', s = 10)
ax2.set_title("Training Data + Predicted Values")
ax2.set_ylabel("x_2")
ax2.set_xlabel("x_1")
ax2.scatter(Xp["X1"],Xp["X2"], color='b', s = 10)
ax2.scatter(Xm["X1"],Xm["X2"], color='r', s = 10)
ax2.scatter(Xp_predictions["X1"], Xp_predictions["X2"], color='c', s = 20)
ax2.scatter(Xm_predictions["X1"], Xm_predictions["X2"], color='m', s = 20)
ax3.set_title("Predicted Data + Decision Boundary (dotted)")
ax3.set_ylabel("x_2_pred")
ax3.set_xlabel("x_1_pred")
ax3.scatter(Xp_predictions["X1"], Xp_predictions["X2"], color='c')
ax3.scatter(Xm_predictions["X1"], Xm_predictions["X2"], color='m')
ax3.plot(xd, yd, 'k', lw=1, ls='--')

plt.show()
```

CODE FOR PART B

```
# id:19-19-19
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
df = pd.read_csv("data.csv",header=None, comment="#",
names=["X1","X2","X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]

X_train, X_test, y_train, y_test = train_test_split(X_all, X_labels,
test_size=0.33, random_state=42)

X_predictions = X_test

C_array = [1, 100, 1000, 10000, 0.0001, 0.00001]

fig, axs = plt.subplots(2,3,sharex=True, sharey=True)

fig.suptitle("Data + Predicted Data + Decision boundary\n Based on different
C kernels")

axs = axs.flatten()

#splitting the data
#Seperating 1 and -1 values into Xpositive and Xnegative
Xp = df.loc[df["X3"]==1]
Xm = df.loc[df["X3"]==-1]
Xp_labels = Xp["X3"]
Xm_labels = Xm["X3"]
```

```
print(type(axs))

#training the models
def svm_fit_test_measure(c_val,i):
    #training the models
    clf = LinearSVC(C=c_val)
    clf.fit(X_train, y_train)
    X_predictions = X_test
    predicted_y = clf.predict(X_test[["X1","X2"]])

    X_predictions["X3"] = predicted_y

    Xp_predictions = X_predictions.loc[X_predictions["X3"]==1]
    Xm_predictions = X_predictions.loc[X_predictions["X3"]==-1]

    y_list_ok = y_test.tolist()

    #performance evaluation
    true_neg, false_pos, false_neg, true_pos = confusion_matrix(y_list_ok,
predicted_y).ravel()
    conf_matrix = confusion_matrix(y_list_ok, predicted_y)
    accuracy = accuracy_score(y_list_ok, predicted_y)

    b = clf.intercept_[0]
    w1, w2 = clf.coef_.T

    # Calculate the intercept and gradient of the decision boundary.
    c = -b/w2
    m = -w1/w2
    xmin, xmax = -1, 1
    xd = np.array([xmin, xmax])
    yd = m*xd + c


    #plotting the data
    axs[i].plot(xd, yd, 'k', lw=3, color='g')
```

```
    axs[i].set_xlim([-1,1])
    axs[i].set_ylim([-1,1])
    axs[i].set_title("C = " + str(c_val))
    axs[i].scatter(Xp["X1"],Xp["X2"], color='b', s = 7)
    axs[i].scatter(Xm["X1"],Xm["X2"], color='aqua', s = 7)
    axs[i].scatter(Xp_predictions["X1"], Xp_predictions["X2"],
color='hotpink', s = 10)
    axs[i].scatter(Xm_predictions["X1"], Xm_predictions["X2"], color='r', s =
10)

    print(true_neg, false_pos, false_neg, true_pos)
    print("conf_matrix: \n", conf_matrix)
    print("accuracy score: ", accuracy)

    return [c_val, true_neg, false_pos, false_neg, true_pos, accuracy,
clf.get_params()]

svm_array = []

for i in range(len(C_array)):
    svm_array.append(svm_fit_test_measure(C_array[i],i))

plt.show()
print(svm_array)
```

CODE FOR PART C

```
# id:19-19-19
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.dummy import DummyClassifier

df = pd.read_csv("data_etd.csv",header=None, comment="#",
names=["X1","X2", "X12", "X22", "X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]
X_train, X_test, y_train, y_test = train_test_split(X_all, X_labels,
test_size=0.20, random_state=42)

#training the models
clf = LogisticRegression(random_state=0).fit(X_train, y_train)
dummy_clf = DummyClassifier(strategy="most_frequent").fit(X_train,
y_train)

params = clf.get_params()
print(params)

predictions = clf.predict(X_test)
dummy_predictions = dummy_clf.predict(X_test)

#splitting the data
X_predictions = X_test
X_predictions["X3"] = predictions

y_list_ok = y_test.tolist()
```

```
X_test = X_test.drop(columns=["X12"])
X_test = X_test.drop(columns=["X22"])
X_test = X_test.drop(columns=["X3"])

X_test_x1 = X_test["X1"].tolist()
X_test_x2 = X_test["X2"].tolist()

pred_tp_x1 = []
pred_tp_x2 = []

pred_tn_x1 = []
pred_tn_x2 = []

pred_fp_x1 = []
pred_fp_x2 = []

pred_fn_x1 = []
pred_fn_x2 = []

for i in range(len(y_list_ok)):
    if predictions[i] == y_list_ok[i]:
        if predictions[i] == 1:
            pred_tp_x1.append(X_test_x1[i])
            pred_tp_x2.append(X_test_x2[i])
        if predictions[i] == -1:
            pred_tn_x1.append(X_test_x1[i])
            pred_tn_x2.append(X_test_x2[i])
    if predictions[i] != y_list_ok[i]:
        if predictions[i] == 1:
            pred_fp_x1.append(X_test_x1[i])
            pred_fp_x2.append(X_test_x2[i])
        if predictions[i] == -1:
            pred_fn_x1.append(X_test_x1[i])
            pred_fn_x2.append(X_test_x2[i])

#end of data splitting
```

```
#performance evaluation
true_neg, false_pos, false_neg, true_pos = confusion_matrix(y_list_ok,
predictions).ravel()
conf_matrix = confusion_matrix(y_list_ok, predictions)
accuracy = accuracy_score(y_list_ok, predictions)

#plotting the data
print(true_neg, false_pos, false_neg, true_pos)
print("conf_matrix: \n", conf_matrix)
print("accuracy score: ", accuracy)

#performance evaluation
true_neg, false_pos, false_neg, true_pos = confusion_matrix(y_list_ok,
dummy_predictions).ravel()
conf_matrix = confusion_matrix(y_list_ok, dummy_predictions)
accuracy = accuracy_score(y_list_ok, dummy_predictions)

#plotting the data
print(true_neg, false_pos, false_neg, true_pos)
print("conf_matrix: \n", conf_matrix)
print("accuracy score: ", accuracy)

#plotting the data
plt.scatter(pred_tp_x1,pred_tp_x2, color='blue', label='Correct Positives')
plt.scatter(pred_tn_x1,pred_tn_x2, color='aqua', label='Correct Negatives')
plt.scatter(pred_fp_x1,pred_fp_x2, color='red', label='False Positives')
plt.scatter(pred_fn_x1,pred_fn_x2, color='lawngreen', label='False
Negatives')
plt.legend(loc='upper left', shadow=True)
plt.title("Predicted Model Values")
plt.xlabel("x1_pred")
plt.ylabel("x2_pred")

plt.show()
```