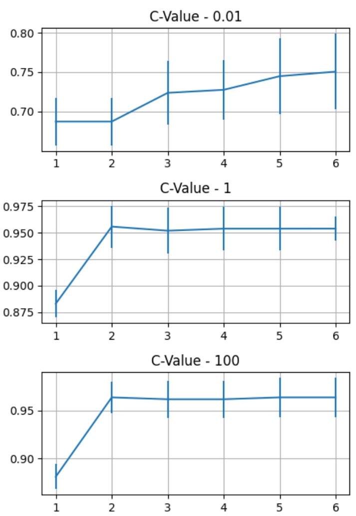
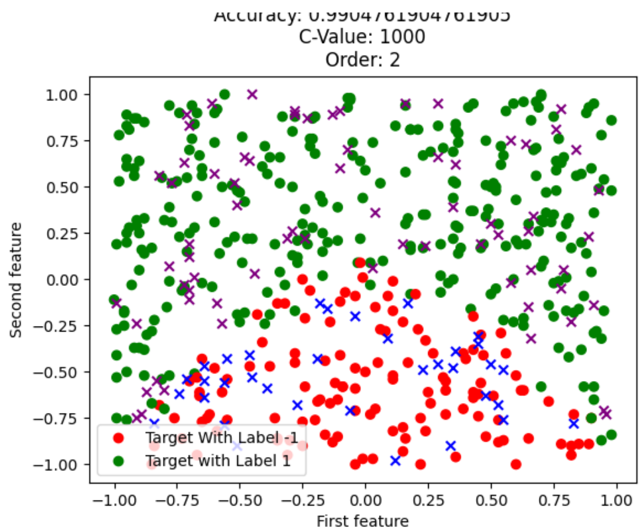
VSEVOLOD SYRTSOV
18323202
DATASET ONE ID: # id:6-6-6-0
DATASET TWO ID: # id:6--6--6-0

Q1 - FIRST DATASET

(a).

To select the best configuration for the logistic regression model, a k-fold cross validation with 5 folds was run to select from a range the best C-value and polynomial order for manipulating the features, for the model. The model was selected based on how accurately it predicted the target label. I refrained from presenting the parameters for every single configuration as it would've been not very conducive to measuring the performance of the

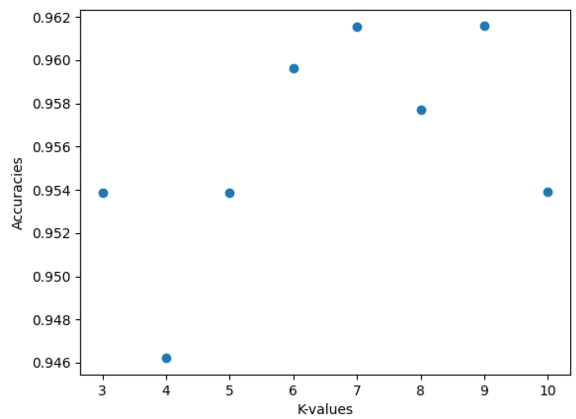




The above two charts show all accuracies for the different configurations, with the chart on the right showcasing the performance of the best performing model configurations, with the polynomial features augmented with a maximum order of 4, and the model penalised with a C-value of 10. Though for C-values above 1, models yield generally the same accuracy across the board I have selected the best performing model by a relatively small margin. So, the best C-value across all choices is,
(i) 10
...and the maximum polynomial order of the augmented features:
(ii) 4

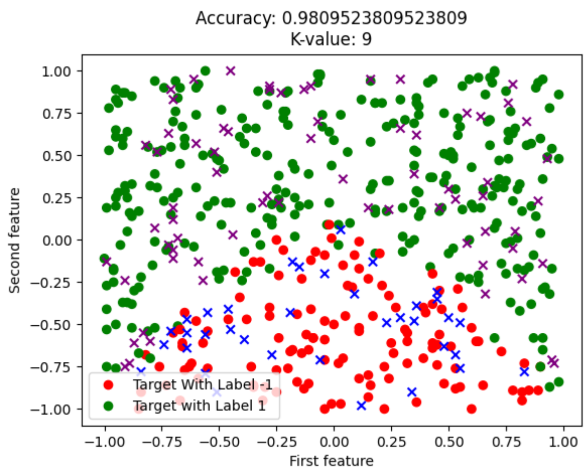
(b).

To select the most suited k value for a range of kNN Classifiers, I once again tested the accuracy of each model against each other,



with a k-fold cross validation with 5 folds once again.

The chart on the previous page showcases the accuracies of every k-configured model. As the k-value approaches 9, the tendency of the model to accurately predict the target label increases. After this the accuracy seems to decrease. On the right hand side the outputs of the model trained on a k-value of 9 are shown.



(c).

Each following table was obtained from the confusion_matrix() metric package from sklearn. Each table was obtained from the corresponding selected best performing models.
The columns correspond to the **Actual Label** and the rows to the **Predicted Label.**

kNN Classifier Confusion Matrix

	P	**N**
P	78	3
N	3	47

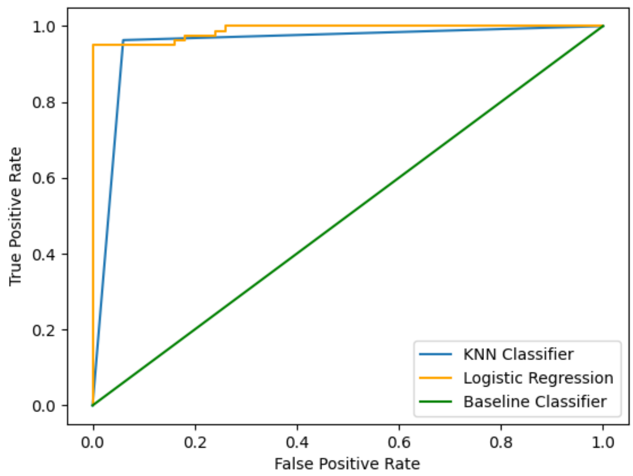
Logistic Regression Confusion Matrix

	P	**N**
P	77	4
N	4	46

Most Frequent Label Classifier Confusion Matrix

	P	**N**
P	81	50
N	0	0

(d).



Here is the ROC curve. This curve was obtained from the roc_curve() method provided by the sklearn package. Clearly the kNN classifier and the Logistic Regression classifier have quite similar ROC curves.

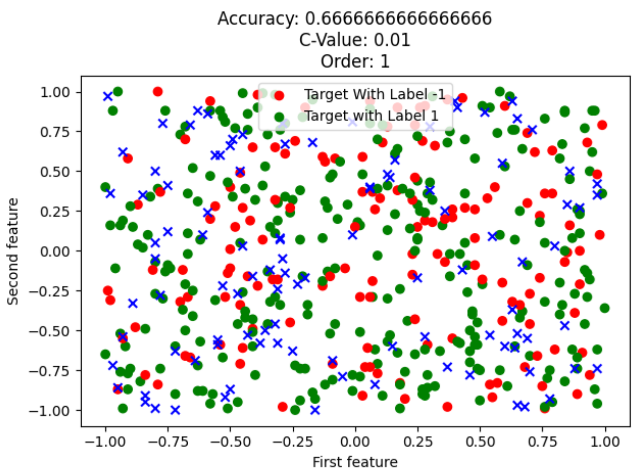
(e).

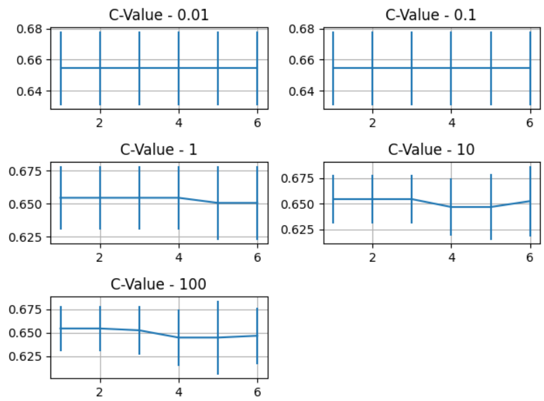
Based on the data obtained from the observations in (c) and (d), the choice to pick a kNN classifier over a Logistic Regression for this scenario is a difficult one as both deliver similar performances, as well as a high accuracy for predicting the target label. Neither is *significantly* better than the other, from looking at both the confusion matrix and ROC curves. The ROC curves and confusion matrices for the baseline classifier show that it is as good as useless, when compared to the other models. The kNN classifier and logistic regression are both as good as each other, as well as accurate, so for this scenario I would recommend either, but definitely not the most common target label classifier.

Q1 - SECOND DATASET

(a).

Once again, the same setup was used to select the best performing model, though after observing all the accuracies from the configurations they don’t vary very much at all which makes me believe that the data has no correlation, or that the models selected to observe the data are not fit for the purpose. This is the data referred to in the introduction to the

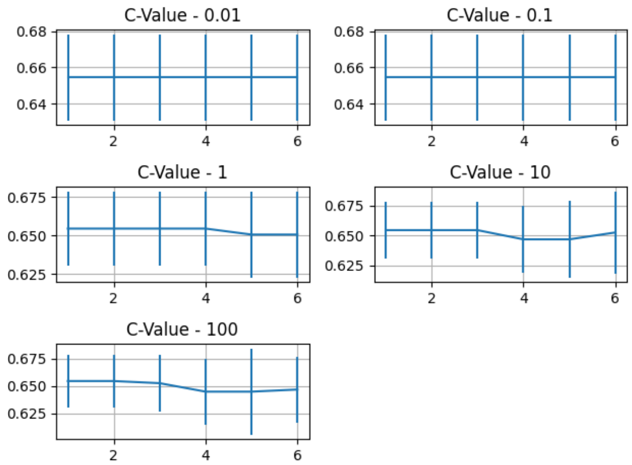




question as the dataset that does not capture the important relationships, or is too noisy. At this rate any model can be selected from the pool and be as good as the other.

(b).

All the charts on the right hand side correspond to the accuracy of configurations obtained. Once again, the values are relatively the same, and no model can be selected for consideration as a significant performer. The model selected as the best performer is the model that has the highest accuracy.



(c).

Each following table was obtained from the confusion_matrix() metric package from sklearn. Each table was obtained from the corresponding selected best performing models.
The columns correspond to the **Actual Label** and the rows to the **Predicted Label.**

kNN Classifier Confusion Matrix

	P	**N**
P	4	39
N	19	69

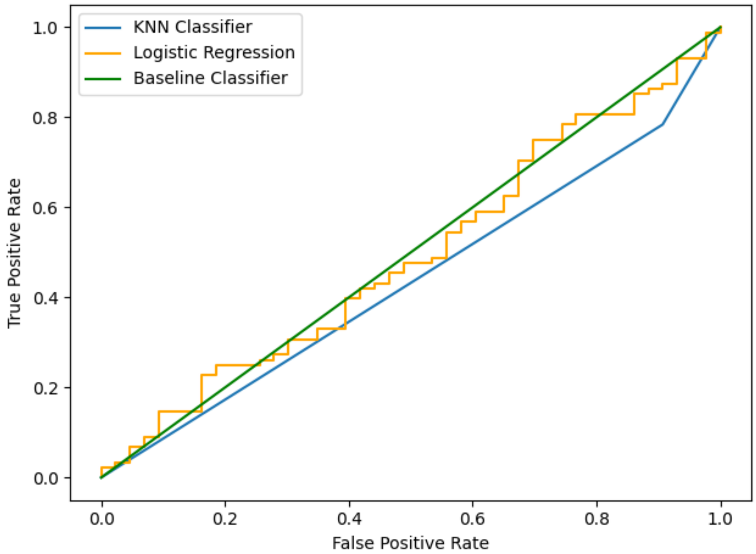
Logistic Regression Confusion Matrix

	P	**N**
P	0	43
N	0	88

Most Frequent Label Classifier Confusion Matrix

	P	**N**
P	0	43
N	0	88

(d).



Above is the ROC curve obtained for the logistic regression, kNN classifier and baseline classifier. There doesn't appear to be any significant between any of the models. I used the same method provided by the sklearn library as used for the first dataset.

(e).

As remarked upon in previous questions, neither the observations from **(c)** or **(d)** lend themselves to make any distinctions in performances between the models. The two models perform as well as the baseline classifier if not slightly worse, which is not a good argument for either of them, using both the confusion matrix and the calculation of the ROC curves. The dataset is without either too noisy, does not represent the relationship between input and output or the models used for the process aren't well suited for the purpose.

APPENDIX

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold, train_test_split
from sklearn.dummy import DummyClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt

# -- FORMAT DATASET --
data = open("data","r").read().split("\n")
dataset_id_one = data[0]
data = data[1:]
data = data[:-1]

index = 0
for row in data:
   if row[0] == "#":
       break
   index += 1

data_one = data[:index]
data_two = data[index:]
dataset_id_two = data_two[0]
data_two = data_two[1:]

print("DATASET ONE ID: {}".format(dataset_id_one))
print("DATASET TWO ID: {}".format(dataset_id_two))


data_one = [x.split(",") for x in data_one]
data_two = [x.split(",") for x in data_two]

f1_one = []
f2_one = []
f1_two = []
f2_two = []
features_one = []
features_two = []
```

```python
label_one = []
label_two = []

for (row_one, row_two) in zip(data_one,data_two):
   f1_one.append(float(row_one[0]))
   f2_one.append(float(row_one[1]))
   features_one.append([float(row_one[0]), float(row_one[1])])
   label_one.append(float(row_one[2]))

   f1_two.append(float(row_two[0]))
   f2_two.append(float(row_two[1]))
   features_two.append([float(row_two[0]), float(row_two[1])])
   label_two.append(float(row_two[2]))

# -- END OF FORMATTING

# --- FIRST DATASET ---

# -- QUESTION ONE --
# a
# Acquire the accuracy and predictions of each Logistic Regression
model configured with C and polynomial orders.


C_values = [0.01, 0.1, 1, 10, 100, 1000]
poly_orders = [1,2,3,4,5,6]
dataframe_values = []
all_accuracy_stds = []
all_accuracies_for_plots = []


colormap = ListedColormap(["r", "g"])
pred_colormap = ListedColormap(["b","purple"])

classes = ["Target With Label -1", "Target with Label 1"]
selected_logreg_model = None
selected_logreg_model_accuracy = 0
selected_lr_poly_order = 1


for c in C_values:
   accuracy_stds = []
   accuracy_for_plot = []
```

```
    for p in poly_orders:
        split_vals = []
        predictions = []
        plotted = False
        for train, test in KFold(n_splits=5).split(features_one):
            x_poly =
PolynomialFeatures(p).fit_transform(np.array(features_one)[train])
            x_poly_test =
PolynomialFeatures(p).fit_transform(np.array(features_one)[test])
            model = LogisticRegression(penalty='l2',C=c,
max_iter=1000).fit(x_poly,np.array(label_one)[train])
            prediction = model.predict(x_poly_test)
            predictions.append(prediction)
            accuracy = accuracy_score(np.array(label_one)[test],
prediction, normalize=True)
            split_vals.append([c,p,accuracy])
            if (c==1 or c == 10 or c == 100 or c==1000) and p == 2 and
not plotted:
                # Plot each prediction
                categories = [x if x != -1 else 0 for x in
np.array(label_one)[train]]
                plot =  plt.scatter(np.array(f1_one)[train],
np.array(f2_one)[train], c=categories, cmap=colormap)
                plt.title("Accuracy: {}\nC-Value: {}\nOrder:
{}".format(accuracy,c,p))
                plt.xlabel("First feature")
                plt.ylabel("Second feature")
                plt.legend(handles=plot.legend_elements()[0],
labels=classes)
                pred_categories = [x if x != -1 else 0 for x in
prediction]
                pred_plot =
plt.scatter(np.array(f1_one)[test],np.array(f2_one)[test],
c=pred_categories, cmap=pred_colormap, marker="x")
                # plt.show()
                plotted=True
        accuracy_stds.append(np.array([x[2] for x in split_vals]).std())
        split_vals = np.array(split_vals).mean(axis=0)
        accuracy_for_plot.append(split_vals[2])
        dataframe_values.append(split_vals)

        # Update model to be the one with the greatest accuracy
```

```
        if selected_logreg_model == None or split_vals[2] >
selected_logreg_model_accuracy:
            selected_logreg_model = model
            selected_logreg_model_accuracy = split_vals[2]
            selected_lr_poly_order = p
    all_accuracy_stds.append(accuracy_stds)
    all_accuracies_for_plots.append(accuracy_for_plot)


# -- Plot the accuracies of each config and their standard deviations
for Logistic Regression --

subplot_index = 1
c_index = 0


for accuracy,std in zip(all_accuracies_for_plots,all_accuracy_stds):
    plt.subplot(4,2,subplot_index)
    plt.grid()
    plt.title("C-Value - {}".format(C_values[c_index]))
    plt.errorbar(poly_orders,accuracy,std)
    subplot_index += 1
    c_index += 1

plt.show()
print("Accuracy of selected Logistic Regression Model:
{}".format(selected_logreg_model_accuracy))
print(selected_logreg_model)
print(selected_lr_poly_order)
# -- End of plotting accuracies

# b
# kNN
selected_knn_model = None
selected_knn_model_accuracy = 0
k_values = [3,4,5,6,7,8,9,10,15,20,25,30]
subplot_index = 1
knn_accuracies = []
for k in k_values:
    accuracies = []
    plotted = False
    for train, test in KFold(n_splits=5).split(features_one):
```

```
        model =
KNeighborsClassifier(n_neighbors=k,weights="uniform").fit(np.array(feat
ures_one)[train], np.array(label_one)[train])
        prediction = model.predict(np.array(features_one)[test])
        accuracy = accuracy_score(np.array(label_one)[test], prediction,
normalize=True)
        accuracies.append(accuracy)
        if k==9 and not plotted:
            categories = [x if x != -1 else 0 for x in
np.array(label_one)[train]]
            plot =  plt.scatter(np.array(f1_one)[train],
np.array(f2_one)[train], c=categories, cmap=colormap)
            plt.title("Accuracy: {}\nK-value: {}".format(accuracy,k))
            plt.xlabel("First feature")
            plt.ylabel("Second feature")
            plt.legend(handles=plot.legend_elements()[0],
labels=classes)
            pred_categories = [x if x != -1 else 0 for x in prediction]
            pred_plot =
plt.scatter(np.array(f1_one)[test],np.array(f2_one)[test],
c=pred_categories, cmap=pred_colormap, marker="x")
            # plt.show()
        plotted = True
    accuracy = np.array(accuracies).mean()
    knn_accuracies.append(accuracy)
    if selected_knn_model == None or accuracy >
selected_knn_model_accuracy:
            selected_knn_model = model
            selected_knn_model_accuracy = accuracy
    print("K-values: {} Accuracy:
{}".format(k,np.array(accuracies).mean()))

plt.clf()
plt.cla()

plt.scatter(k_values,knn_accuracies)
plt.xlabel("K-values")
plt.ylabel("Accuracies")
plt.show()

print("Accuracy of knn model: {}".format(selected_knn_model_accuracy))
print(selected_knn_model)
```

```
# c
# -- Calculate confusion matrix for both the kNN classifier and
regression classifier

train_features,test_features,train_labels,test_labels =
train_test_split(features_one,label_one,test_size=0.25,random_state=0)
knn_preds = selected_knn_model.predict(test_features)
lr_preds =
selected_logreg_model.predict(PolynomialFeatures(selected_lr_poly_order
).fit_transform(test_features))
dummy_model =
DummyClassifier(strategy="most_frequent").fit(train_features,train_labe
ls)
dummy_preds = dummy_model.predict(test_features)
knn_tn, knn_fp, knn_fn, knn_tp = confusion_matrix(test_labels,
knn_preds).ravel()
lr_tn, lr_fp, lr_fn, lr_tp = confusion_matrix(test_labels,
lr_preds).ravel()
bc_tn, bc_fp, bc_fn, bc_tp = confusion_matrix(test_labels,
dummy_preds).ravel()


print("\n\nkNN Classifier Confusion Matrix\n\nTrue Negative: {}\nFalse
Postive: {}\nFalse Negative: {}\nTrue Positive: {}".format(knn_tn,
knn_fp, knn_fn, knn_tp))
print(" \n-----\nLogistic Regression Confusion Matrix\n\nTrue Negative:
{}\nFalse Postive: {}\nFalse Negative: {}\nTrue Positive:
{}".format(lr_tn, lr_fp, lr_fn, lr_tp))
print(" \n-----\nMost Frequent Label Classifier Confusion
Matrix\n\nTrue Negative: {}\nFalse Postive: {}\nFalse Negative:
{}\nTrue Positive: {}".format(bc_tn, bc_fp, bc_fn, bc_tp))

# d
# Calculate and plot roc curves for knn classifier, lr model and dummy
classifier

plt.cla()
plt.clf()

fpr, tpr, _ = roc_curve(test_labels,knn_preds)

plt.plot(fpr,tpr)
```

```
fpr, tpr, _ = roc_curve(test_labels,
selected_logreg_model.decision_function(PolynomialFeatures(selected_lr_
poly_order).fit_transform(test_features)))
plt.plot(fpr,tpr,color="orange")

fpr, tpr, _ = roc_curve(test_labels,dummy_preds)

plt.plot(fpr,tpr, color="green")
plt.legend(["KNN Classifier", "Logistic Regression", "Baseline
Classifier"])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()

# --- SECOND DATASET ---

# -- QUESTION ONE --
# a
# Acquire the accuracy and predictions of each Logistic Regression
model configured with C and polynomial orders.
plt.cla()
plt.clf()

C_values = [0.01, 0.1, 1, 10, 100]
poly_orders = [1,2,3,4,5,6]
dataframe_values = []
all_accuracy_stds = []
all_accuracies_for_plots = []


colormap = ListedColormap(["r", "g"])
pred_colormap = ListedColormap(["b","purple"])

classes = ["Target With Label -1", "Target with Label 1"]
selected_logreg_model = None
selected_logreg_model_accuracy = 0
selected_lr_poly_order = 1


for c in C_values:
   accuracy_stds = []
   accuracy_for_plot = []
   for p in poly_orders:
```

```
        split_vals = []
        predictions = []
        plotted = False
        for train, test in KFold(n_splits=5).split(features_two):
            x_poly =
PolynomialFeatures(p).fit_transform(np.array(features_two)[train])
            x_poly_test =
PolynomialFeatures(p).fit_transform(np.array(features_two)[test])
            model = LogisticRegression(penalty='l2',C=c,
max_iter=500).fit(x_poly,np.array(label_two)[train])
            prediction = model.predict(x_poly_test)
            predictions.append(prediction)
            accuracy = accuracy_score(np.array(label_two)[test],
prediction, normalize=True)
            split_vals.append([c,p,accuracy])
            if (c==0.01 or c==1 or c == 100) and p==1 and not plotted:
                # Plot each prediction
                categories = [x if x != -1 else 0 for x in
np.array(label_two)[train]]
                plot =  plt.scatter(np.array(f1_two)[train],
np.array(f2_two)[train], c=categories, cmap=colormap)
                plt.title("Accuracy: {}\nC-Value: {}\nOrder:
{}".format(accuracy,c,p))
                plt.xlabel("First feature")
                plt.ylabel("Second feature")
                plt.legend(handles=plot.legend_elements()[0],
labels=classes)
                pred_categories = [x if x != -1 else 0 for x in
prediction]
                pred_plot =
plt.scatter(np.array(f1_two)[test],np.array(f2_two)[test],
c=pred_categories, cmap=pred_colormap, marker="x")
                plt.show()
            plotted=True
        accuracy_stds.append(np.array([x[2] for x in split_vals]).std())
        split_vals = np.array(split_vals).mean(axis=0)
        accuracy_for_plot.append(split_vals[2])
        dataframe_values.append(split_vals)

        # Update model to be the one with the greatest accuracy
        if selected_logreg_model == None or split_vals[2] >
selected_logreg_model_accuracy:
            selected_logreg_model = model
```

```
            selected_logreg_model_accuracy = split_vals[2]
            selected_lr_poly_order = p
    all_accuracy_stds.append(accuracy_stds)
    all_accuracies_for_plots.append(accuracy_for_plot)


# -- Plot the accuracies of each config and their standard deviations
for Logistic Regression --

subplot_index = 1
c_index = 0
print(len(all_accuracy_stds))
print(len(all_accuracies_for_plots))
for accuracy,std in zip(all_accuracies_for_plots,all_accuracy_stds):
    plt.subplot(3,2,subplot_index)
    plt.grid()
    plt.title("C-Value - {}".format(C_values[c_index]))
    plt.errorbar(poly_orders,accuracy,std)
    subplot_index += 1
    c_index += 1

plt.show()
print("Accuracy of selected Logistic Regression Model:
{}".format(selected_logreg_model_accuracy))
print(selected_logreg_model)
print(selected_lr_poly_order)
# -- End of plotting accuracies

# b
# kNN
selected_knn_model = None
selected_knn_model_accuracy = 0
k_values = [3,4,5,6,7,8,9,10]
subplot_index = 1
for k in k_values:
    accuracies = []
    plotted = False
    for train, test in KFold(n_splits=5).split(features_two):
        model =
KNeighborsClassifier(n_neighbors=k,weights="uniform").fit(np.array(feat
ures_two)[train], np.array(label_one)[train])
        prediction = model.predict(np.array(features_two)[test])
```

```
        accuracy = accuracy_score(np.array(label_two)[test], prediction,
normalize=True)
        accuracies.append(accuracy)
        if k==9 and not plotted:
            categories = [x if x != -1 else 0 for x in
np.array(label_two)[train]]
            plot =  plt.scatter(np.array(f1_two)[train],
np.array(f2_two)[train], c=categories, cmap=colormap)
            plt.title("Accuracy: {}\nK-value: {}".format(accuracy,k))
            plt.xlabel("First feature")
            plt.ylabel("Second feature")
            plt.legend(handles=plot.legend_elements()[0],
labels=classes)
            pred_categories = [x if x != -1 else 0 for x in prediction]
            pred_plot =
plt.scatter(np.array(f1_two)[test],np.array(f2_two)[test],
c=pred_categories, cmap=pred_colormap, marker="x")
            plt.show()
        plotted = True
    accuracy = np.array(accuracies).mean()
    if selected_knn_model == None or accuracy >
selected_knn_model_accuracy:
            selected_knn_model = model
            selected_knn_model_accuracy = accuracy
    print("K-values: {} Accuracy:
{}".format(k,np.array(accuracies).mean()))

print("Accuracy of knn model: {}".format(selected_knn_model_accuracy))
print(selected_knn_model)
# c
# -- Calculate confusion matrix for both the kNN classifier and
regression classifier

train_features,test_features,train_labels,test_labels =
train_test_split(features_two,label_two,test_size=0.25,random_state=0)
knn_preds = selected_knn_model.predict(test_features)
lr_preds =
selected_logreg_model.predict(PolynomialFeatures(selected_lr_poly_order
).fit_transform(test_features))
dummy_model =
DummyClassifier(strategy="most_frequent").fit(train_features,train_labe
ls)
dummy_preds = dummy_model.predict(test_features)
```

```
knn_tn, knn_fp, knn_fn, knn_tp = confusion_matrix(test_labels,
knn_preds).ravel()
lr_tn, lr_fp, lr_fn, lr_tp = confusion_matrix(test_labels,
lr_preds).ravel()
bc_tn, bc_fp, bc_fn, bc_tp = confusion_matrix(test_labels,
dummy_preds).ravel()


print("\n\nkNN Classifier Confusion Matrix\n\nTrue Negative: {}\nFalse
Postive: {}\nFalse Negative: {}\nTrue Positive: {}".format(knn_tn,
knn_fp, knn_fn, knn_tp))
print(" \n-----\nLogistic Regression Confusion Matrix\n\nTrue Negative:
{}\nFalse Postive: {}\nFalse Negative: {}\nTrue Positive:
{}".format(lr_tn, lr_fp, lr_fn, lr_tp))
print(" \n-----\nMost Frequent Label Classifier Confusion
Matrix\n\nTrue Negative: {}\nFalse Postive: {}\nFalse Negative:
{}\nTrue Positive: {}".format(bc_tn, bc_fp, bc_fn, bc_tp))

# d
# Calculate and plot roc curves for knn classifier, lr model and dummy
classifier

plt.cla()
plt.clf()

fpr, tpr, _ = roc_curve(test_labels,knn_preds)

plt.plot(fpr,tpr)

fpr, tpr, _ = roc_curve(test_labels,
selected_logreg_model.decision_function(PolynomialFeatures(selected_lr_
poly_order).fit_transform(test_features)))
plt.plot(fpr,tpr,color="orange")

fpr, tpr, _ = roc_curve(test_labels,dummy_preds)

plt.plot(fpr,tpr, color="green")
plt.legend(["KNN Classifier", "Logistic Regression", "Baseline
Classifier"])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```