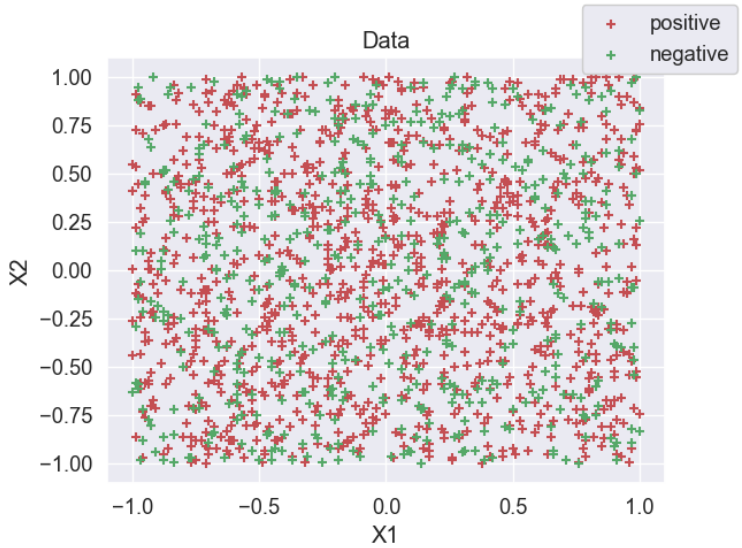
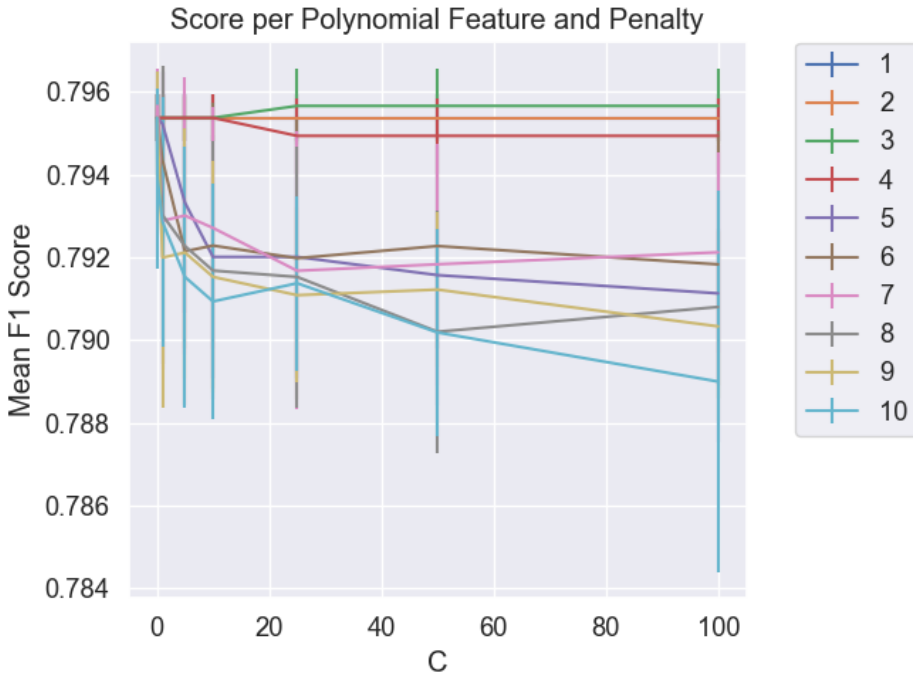
NAME: ARNAV BHATTACHARYA
ID:22307812
COURSE: MSc in Computer Science (Intelligent Systems)
MODULE: Machine Learning
WEEK 4 ASSIGNMENT
DATASET1 ID: 18—18-18-1
DATASET2 ID: 18-18—18-1

THE CODE FOR ALL PARTS IS IN THE APPENDIX

Question i)
Plot of the given Data:

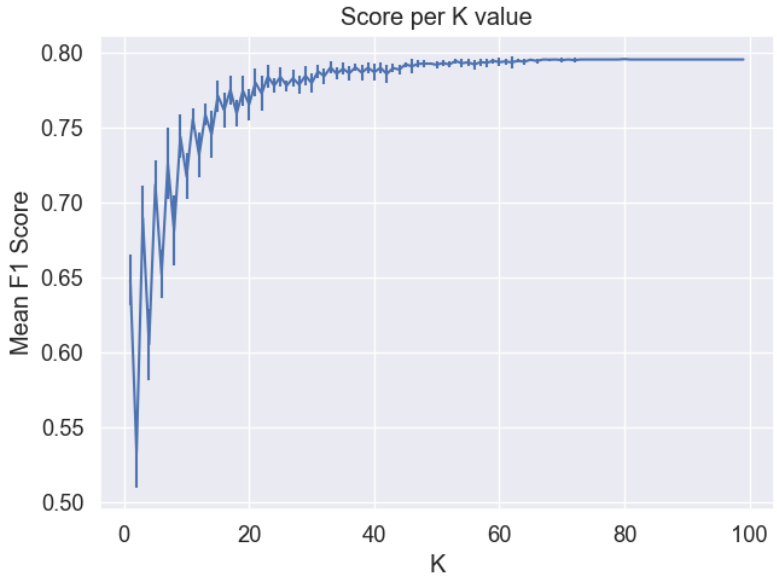


a): Using sklearn augment the two features in the dataset with polynomial features and train a Logistic Regression classifier with L2 penalty added to the cost function. Use cross-validation to select (i) the maximum order of polynomial to use and (ii) the weight C given to the penalty in the cost function. Remember you'll also need to select the range of C values to consider and the range of maximum polynomial orders. Important: It is essential that you present data, including cross-validation plots and plots of model predictions and the training data, and give clear explanations/analysis to justify your choices. Use the ideas/tools you've learned in the module to help you in this. Be sure to include error bars in cross-validation plots. Rememberr to use a performance measure appropriate to classification (so probably not mean square error).
Answer:



As observed from the plot above, when the PolynomialFeature degree is set to 3, the mean F1 score is maximum and consistent at value=0.7956610768978519 when C is above 25.
I tested the Logistic Regression model for the following values of polynomial feature degree and C:
poly_degree=[1,2,3,4,5,6,7,8,9,10]
C_range=[0.01, 0.1, 1, 5, 10, 25, 50, 100]

Question i) b): Now train a kNN classifier on the data. Use cross-validation to select k, again presenting data and explanations/analysis to justify your choice. There is no need to augment the features with polynomial features for a kNN classifier, kNN can already capture nonlinear decision boundaries. Again, it is essential that you present data and give clear explanations/analysis to justify your choices.
Answer:



As observed from the plot above, we can notice that the deviation in the Mean F1 Score keeps on decreasing when k is 40 and above until it becomes negligible. The plot becomes stable when the value is 40 and above. The deviation keeps on decreasing as we increase the k value.

The F1 Score is calculated based on the following formula: **2*((precision*recall)/(precision+recall))**

Question i) c): Calculate the confusion matrices for your trained Logistic Regression and kNN classifier. Also calculate the confusion matrix for one or more baseline classifiers of your choice e.g. one that always predicts the most frequent class in the training data and/or one that makes random predictions.

Answer: The given data is quite scattered in nature and so, the difficulty in predicting accurately is difficult. I split the given data into training and testing data where the test size was 20% of the entire data. I made use of sklearn.model_selection packages' train_test_split function to perform the same.

For kNN Model:

```
The confusion matrix for KNN Model is:

[[  1  99]
 [  4 230]]
The Classification Report for KNN Model is:

              precision    recall  f1-score   support

          -1       0.20      0.01      0.02       100
           1       0.70      0.98      0.82       234

    accuracy                           0.69       334
   macro avg       0.45      0.50      0.42       334
weighted avg       0.55      0.69      0.58       334
```

The accuracy of the model is 69% with the sum of False Positive and False Negative to be 103.

For Logistic Regression Model:

```
The confusion matrix for LR Model is:

[[  0 127]
 [  0 207]]
The Classification Report for LR Model is:

              precision    recall  f1-score   support

          -1       0.00      0.00      0.00       127
           1       0.62      1.00      0.77       207

    accuracy                           0.62       334
   macro avg       0.31      0.50      0.38       334
weighted avg       0.38      0.62      0.47       334
```

The accuracy of the model is 62% with the sum of False Positive and False Negative to be 127.

For Baseline Classifiers, I considered both, the random model and the most frequent model:

For Random Model:

```
The confusion matrix for Random Model is:

[[279 287]
 [533 567]]
The Classification Report for Random Model is:

              precision    recall  f1-score   support

          -1       0.34      0.49      0.40       566
           1       0.66      0.52      0.58      1100

    accuracy                           0.51      1666
   macro avg       0.50      0.50      0.49      1666
weighted avg       0.56      0.51      0.52      1666
```

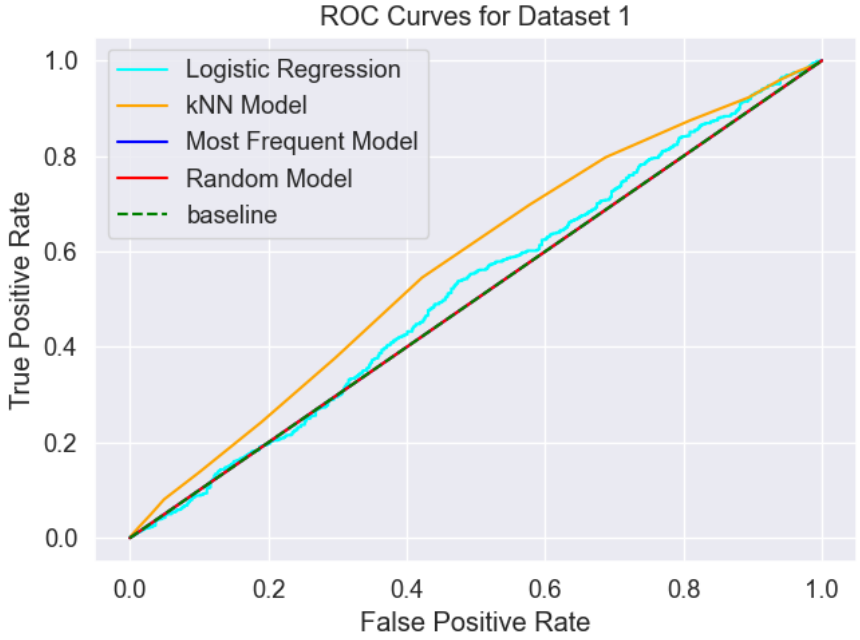
The accuracy of the model is 51% with the sum of False Positive and False Negative to be 820.

For Most Frequent Model:

```
The confusion matrix for Most Frequent Model is:

[[   0  566]
 [   0 1100]]
The Classification Report for Most Frequent Model is:

              precision    recall  f1-score   support

          -1       0.00      0.00      0.00       566
           1       0.66      1.00      0.80      1100

    accuracy                           0.66      1666
   macro avg       0.33      0.50      0.40      1666
weighted avg       0.44      0.66      0.53      1666
```

The accuracy of the model is 66% with the sum of False Positive and False Negative to be 1100.

Question i) d): Plot the ROC curves for your trained Logistic Regression and kNN classifiers. Also plot the point(s) oin the ROC plot corresponding to the baseline classifiers. Be sure to include enough points in the ROC curves to allow the detailed shape to be seen.
Answer:



As observed in the plot above, the ROC curve for Logistic Regression, kNN Classifier, Most Frequent Model, Random Model, and baseline are demarcated by cyan, orange, blue, red, and green respectively. The baseline is marked from (0,0) to (1,1) in the plot. The Logistic Regression Model, kNN Classifier, and the Dummy Classifiers maintain distance from each other and converge at (1,1).

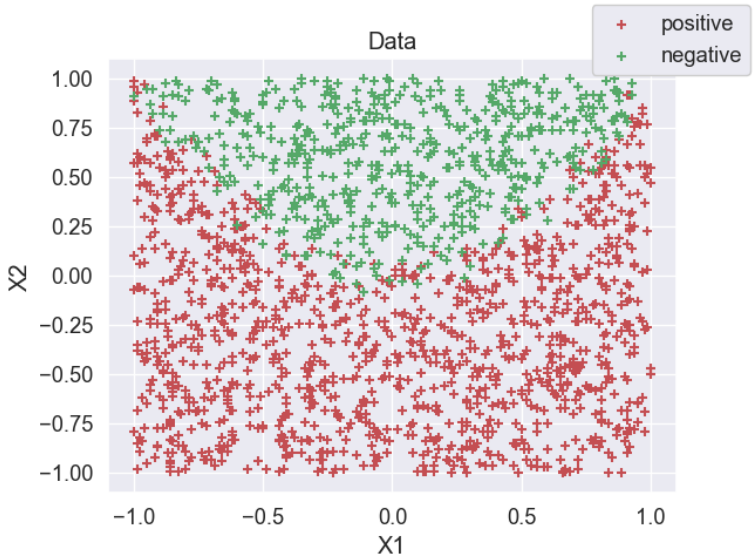
Question i) e): Using the data from (c) and (d) evaluate and compare the performance of the Logistic Regression, kNN and baseline classifiers. Is one classifier significantly better or worse than the other? How do their ROC curves compare with that of a random classifier. Which classifier, if any, would you recommend be used? Explain the reasoning behind you recommendation.
Answer: Since the data is scattered, the accuracy is understandably low all along for all the classifiers. The accuracy for all the classifiers is as follows:

NAME CLASSIFIER	ACCURACY
kNN Classifier	69%
Logistic Regression Model	62%
Random Model	51%
Most Frequent Model	66%

If given the opportunity to select the most suitable classifier for the above dataset, I would most definitely go for kNN Classifier as by definition too, the kNN Classifier marks a point and finds more similar points around it. So, for a scattered dataset, it is most definitely the best choice.

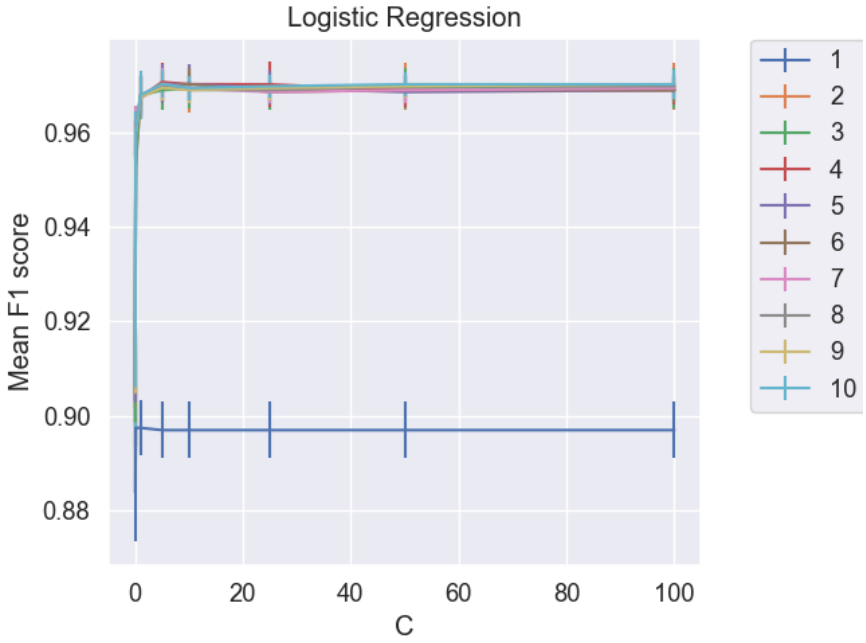
Question ii)

Plot of the given data:



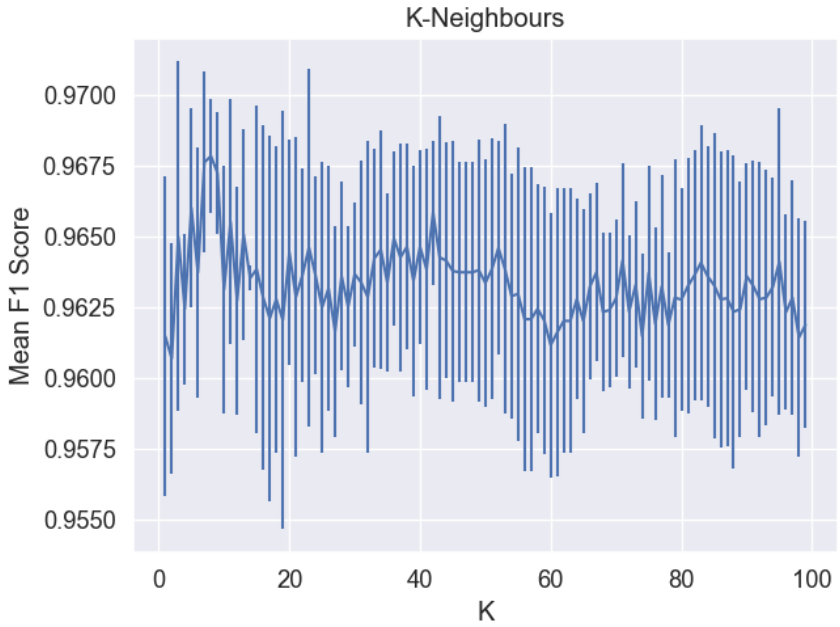
a): Using sklearn augment the two features in the dataset with polynomial features and train a Logistic Regression classifier with L2 penalty added to the cost function. Use cross-validation to select (i) the maximum order of polynomial to use and (ii) the weight C given to the penalty in the cost function. Remember you'll also need to select the range of C values to consider and the range of maximum polynomial orders. Important: It is essential that you present data, including cross-validation plots and plots of model predictions and the training data, and give clear explanations/analysis to justify your choices. Use the ideas/tools you've learned in the module to help you in this. Be sure to include error bars in cross-validation plots. Rememberr to use a performance measure appropriate to classification (so probably not mean square error).

Answer:



As observed from the plot above, when the PolynomialFeature degree is set to 4, the mean F1 score is maximum when C=5 then takes a dip and after that, remains almost consistent around value=0.970118677498937 when C is 10 and above.
I tested the Logistic Regression model for the following values of polynomial feature degree and C:
poly_degree=[1,2,3,4,5,6,7,8,9,10]
C_range=[0.01, 0.1, 1, 5, 10, 25, 50, 100]

Question ii) b): Now train a kNN classifier on the data. Use cross-validation to select k, again presenting data and explanations/analysis to justify your choice. There is no need to augment the features with polynomial features for a kNN classifier, kNN can already capture nonlinear decision boundaries. Again, it is essential that you present data and give clear explanations/analysis to justify your choices.
Answer:



As per definition, kNN means k-Nearest-Neighbours. As observed from the plot above, the Mean F1 Score increases when k is low, and after reaching its peak when k=8, there is a steady decrease. Although there are a few points before k=8 where the F1 score is maximum, but the mean F1 score is maximum for k=8. The F1 Score is calculated based on the following formula:
2*((precision*recall)/(precision+recall))

Question ii) c): Calculate the confusion matrices for your trained Logistic Regression and kNN classifier. Also calculate the confusion matrix for one or more baseline classifiers of your choice e.g. one that always predicts the most frequent class in the training data and/or one that makes random predictions.
Answer: I split the given data into training and testing data where the test size was 20% of the entire data. I made use of sklearn.model_selection packages' train_test_split function to perform the same.
For kNN Model:

```
The confusion matrix for KNN Model is:

[[109   6]
 [  6 225]]
The Classification Report for KNN Model is:

              precision    recall  f1-score   support

          -1       0.95      0.95      0.95       115
           1       0.97      0.97      0.97       231

    accuracy                           0.97       346
   macro avg       0.96      0.96      0.96       346
weighted avg       0.97      0.97      0.97       346
```

The accuracy of the model is 97% with the sum of False Positive and False Negative to be 12.

For Logistic Regression Model:

```
The confusion matrix for LR Model is:

[[109   8]
 [  6 223]]
The Classification Report for LR Model is:

              precision    recall  f1-score   support

          -1       0.95      0.93      0.94       117
           1       0.97      0.97      0.97       229

    accuracy                           0.96       346
   macro avg       0.96      0.95      0.95       346
weighted avg       0.96      0.96      0.96       346
```

The accuracy of the model is 96% with the sum of False Positive and False Negative to be 14.

For Baseline Classifiers, I considered both, the random model and the most frequent model:

For Random Model:

```
The confusion matrix for Random Model is:

[[283 279]
 [614 554]]
The Classification Report for Random Model is:

              precision    recall  f1-score   support

          -1       0.32      0.50      0.39       562
           1       0.67      0.47      0.55      1168

    accuracy                           0.48      1730
   macro avg       0.49      0.49      0.47      1730
weighted avg       0.55      0.48      0.50      1730
```

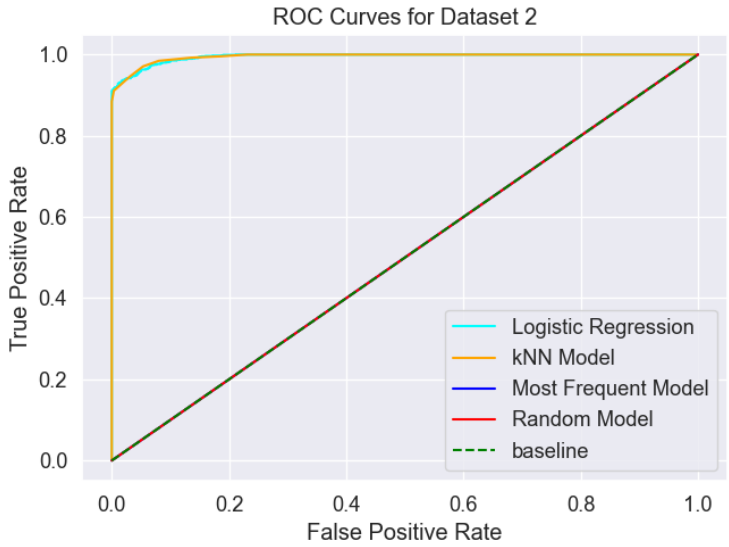
The accuracy of the model is 48% with the sum of False Positive and False Negative to be 893.

For Most Frequent Model:

```
The confusion matrix for Most Frequent Model is:

[[   0  562]
 [   0 1168]]
The Classification Report for Most Frequent Model is:

              precision    recall  f1-score   support

          -1       0.00      0.00      0.00       562
           1       0.68      1.00      0.81      1168
...
    accuracy                           0.68      1730
   macro avg       0.34      0.50      0.40      1730
weighted avg       0.46      0.68      0.54      1730
```

The accuracy of the model is 68% with the sum of False Positive and False Negative to be 1168.

Question ii) d): Plot the ROC curves for your trained Logistic Regression and kNN classifiers. Also plot the point(s) oin the ROC plot corresponding to the baseline classifiers. Be sure to include enough points in the ROC curves to allow the detailed shape to be seen.
Answer:



As observed in the plot above, the ROC curve for Logistic Regression, kNN Classifier, Most Frequent Model, Random Model, and baseline are demarcated by cyan, orange, blue, red, and green respectively. The baseline is marked from (0,0) to (1,1) in the plot. The Logistic Regression Model has a slightly better True Positive Rate in the beginning but the kNN Classifier catches up quickly. We can also observe that the kNN Classifier and the Logistic Regression Model plot overlap most of the time. We can also note that the logistic regression model has a slightly more False Positive Rate than the kNN Classifier.

Question ii) e): Using the data from (c) and (d) evaluate and compare the performance of the Logistic Regression, kNN and baseline classifiers. Is one classifier significantly better or worse than the other? How do their ROC curves compare with that of a random classifier. Which classifier, if any, would you recommend be used? Explain the reasoning behind you recommendation.
Answer: Both the classifiers, Logistic Regression, and kNN Classifier perform well for this dataset due to non-linearity. The baseline classifiers, Random Model and Most Frequent Model perform quite poorly. The accuracy for all the classifiers is as follows:

NAME CLASSIFIER	ACCURACY

kNN Classifier	97%
Logistic Regression Model	96%
Random Model	48%
Most Frequent Model	68%

If given the opportunity to select the most suitable classifier for the above dataset, I would most definitely go for kNN Classifier as by definition too, the kNN Classifier marks a point and finds more similar points around it. Even though, the Logistic Regression Model gives tight competition and the difference in accuracy is only of 1%, but still, kNN performs way better than Logistic Regression.

APPENDIX:

Question i:

```
plt.scatter(X1[y==1], X2[y==1], marker = '+', c = 'r',label="positive")
plt.scatter(X1[y==-1], X2[y==-1], marker = '+', c = 'g',label='negative')
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Data')
plt.legend(bbox_to_anchor=(1.15,1.15), loc='upper right',fancybox=True, framealpha=1)
# plt.show()
plt.savefig('ds1_0.png')

def check_for_C(x,y,poly_degrees, C_range):
  for degree in poly_degrees:
    mean_error=[]
    std_error=[]
    x_poly=PolynomialFeatures(degree).fit_transform(x)
    for c in C_range:
      LRModel=LogisticRegression(penalty='l2',C=c)
      scores=cross_val_score(LRModel,x_poly,y,cv=5,scoring='f1')
      mean_error.append(np.array(scores).mean())
      std_error.append(np.array(scores).std())
      # if(degree==3):
      #   print(f'{c}:{degree}:{np.array(scores).mean()}')

    plt.errorbar(C_range, mean_error, yerr=std_error, label=f'{degree}')
    plt.rc("font", size=15); plt.rcParams['figure.constrained_layout.use'] = True
    plt.xlabel('C')
    plt.ylabel('Mean F1 Score')
    plt.legend(bbox_to_anchor=(1.3, 1.02))
    # plt.show()

poly_degree=[1,2,3,4,5,6,7,8,9,10]
C_range=[0.01, 0.1, 1, 5, 10, 25, 50, 100]
check_for_C(X,y,poly_degree,C_range)
plt.title('Score per Polynomial Feature and Penalty')
# plt.show()
plt.savefig('ds1_a.png')

def check_for_k(k_range,x,y,step):
  mean_error=[]
  std_error=[]
```

```
  for k in k_range:
    KNNModel=KNeighborsClassifier(k)
    score=cross_val_score(KNNModel,x,y,cv=5, scoring='f1')
    mean_error.append(np.array(score).mean())
    std_error.append(np.array(score).std())
  plt.rc('font', size=18)
  plt.rcParams['figure.constrained_layout.use'] = True
  plt.errorbar(list(range(1,100,step)),mean_error,yerr=std_error)
  plt.xlabel('K')
  plt.ylabel('Mean F1 Score')
  plt.title('Score per K value')
step=1
k_range=np.arange(start=1,stop=100,step=step)
check_for_k(k_range,X,y,step)
plt.savefig('ds1_b.png')

KNN_x_train,KNN_x_test,KNN_y_train,KNN_y_test=train_test_split(X,y,test_size=0.2)
KNNModel=KNeighborsClassifier(40).fit(KNN_x_train,KNN_y_train)
KNN_predictions=KNNModel.predict(KNN_x_test)
print(f'The confusion matrix for KNN Model
is:\n\n{confusion_matrix(KNN_y_test,KNN_predictions)}')
print(f'The Classification Report for KNN Model
is:\n\n{classification_report(KNN_y_test,KNN_predictions)}')

LR_x_poly=PolynomialFeatures(degree=3).fit_transform(X)
LR_x_train,LR_x_test,LR_y_train,LR_y_test=train_test_split(LR_x_poly,y,test_size=0.2)
LRModel=LogisticRegression(C=25).fit(LR_x_train,LR_y_train)
LR_predictions=LRModel.predict(LR_x_test)
print(f'The confusion matrix for LR Model is:\n\n{confusion_matrix(LR_y_test,LR_predictions)}')
print(f'The Classification Report for LR Model
is:\n\n{classification_report(LR_y_test,LR_predictions)}')

random_Model=DummyClassifier(strategy="uniform").fit(X,y)
random_predictions=random_Model.predict(X)
print(f'The confusion matrix for Random Model is:\n\n{confusion_matrix(y,random_predictions)}')
print(f'The Classification Report for Random Model
is:\n\n{classification_report(y,random_predictions)}')
most_frequent_Model=DummyClassifier(strategy="most_frequent").fit(X,y)
most_frequent_predictions=most_frequent_Model.predict(X)
print(f'The confusion matrix for Most Frequent Model
is:\n\n{confusion_matrix(y,most_frequent_predictions)}')
print(f'The Classification Report for Most Frequent Model
is:\n\n{classification_report(y,most_frequent_predictions)}')

KNN_probability=KNNModel.predict_proba(X)
most_frequent_probability=most_frequent_Model.predict_proba(X)
random_probability=random_Model.predict_proba(X)
fpr, tpr, _ = roc_curve(y, LRModel.decision_function(LR_x_poly))
knn_fpr, knn_tpr, thresh = roc_curve(y, KNN_probability[:, 1])
most_freq_fpr, most_freq_tpr, thresh = roc_curve(y, most_frequent_probability[:, 1])
rand_fpr, rand_tpr, thresh = roc_curve(y, random_probability[:, 1])
```

```
plt.rc('font', size=20)
plt.plot(fpr, tpr, color='cyan',label='Logistic Regression')
plt.plot(knn_fpr, knn_tpr, color='orange',label='kNN Model')
plt.plot(most_freq_fpr, most_freq_tpr, color='blue',label='Most Frequent Model')
plt.plot(rand_fpr, rand_tpr, color='red', label='Random Model')
plt.plot([0, 1], [0, 1], color='green', linestyle='--',label='baseline')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title(
   "ROC Curves for Dataset 1")
plt.legend()
# plt.show()
plt.savefig('ds1_d.png')

Question ii:
plt.scatter(D2_X1[D2_y==1], D2_X2[D2_y==1], marker = '+', c = 'r',label="positive")
plt.scatter(D2_X1[D2_y==-1], D2_X2[D2_y==-1], marker = '+', c = 'g',label='negative')
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Data')
plt.legend(bbox_to_anchor=(1.15,1.15), loc='upper right',fancybox=True, framealpha=1)
# plt.show()
plt.savefig('ds2_0.png')

def check_for_C(x,y,poly_degrees, C_range):
  for degree in poly_degrees:
    mean_error=[]
    std_error=[]
    x_poly=PolynomialFeatures(degree).fit_transform(x)
    for c in C_range:
      LRModel=LogisticRegression(penalty='l2',C=c)
      scores=cross_val_score(LRModel,x_poly,y,cv=5,scoring='f1')
      mean_error.append(np.array(scores).mean())
      std_error.append(np.array(scores).std())
      # if(degree==4):
      #   print(f'{c}:{degree}:{scores.mean()}')
    plt.errorbar(C_range, mean_error, yerr=std_error, label=f'{degree}')
    plt.rc("font", size=18); plt.rcParams['figure.constrained_layout.use'] = True
    plt.xlabel('C')
    plt.ylabel('Mean F1 score')
    plt.legend(bbox_to_anchor=(1.3, 1.02))
    # plt.show()

poly_degree=[1,2,3,4,5,6,7,8,9,10]
C_range=[0.01, 0.1, 1, 5, 10, 25, 50, 100]
check_for_C(D2_X,D2_y,poly_degree,C_range)
plt.title('Logistic Regression')
# plt.show()
plt.savefig('ds2_a.png')
```

```
def check_for_k(k_range,x,y,step):
  mean_error=[]
  std_error=[]
  for k in k_range:
    KNNModel=KNeighborsClassifier(k)
    score=cross_val_score(KNNModel,x,y,cv=5, scoring='f1')
    mean_error.append(np.array(score).mean())
    std_error.append(np.array(score).std())
    # print(f'k: {k}; score:{score.mean()}')
  plt.rc('font', size=18)
  plt.rcParams['figure.constrained_layout.use'] = True
  plt.errorbar(list(range(1,100,step)),mean_error,yerr=std_error)
  plt.xlabel('K')
  plt.ylabel('Mean F1 Score')
  plt.title('K-Neighbours')
step=1
k_range=np.arange(start=1,stop=100,step=step)
check_for_k(k_range,D2_X,D2_y,step)
plt.savefig('ds2_b.png')

KNN_x_train,KNN_x_test,KNN_y_train,KNN_y_test=train_test_split(D2_X,D2_y,test_size=0.2)
KNNModel=KNeighborsClassifier(8).fit(KNN_x_train,KNN_y_train)
KNN_predictions=KNNModel.predict(KNN_x_test)
print(f'The confusion matrix for KNN Model
is:\n\n{confusion_matrix(KNN_y_test,KNN_predictions)}')
print(f'The Classification Report for KNN Model
is:\n\n{classification_report(KNN_y_test,KNN_predictions)}')

LR_x_poly=PolynomialFeatures(degree=4).fit_transform(D2_X)
LR_x_train,LR_x_test,LR_y_train,LR_y_test=train_test_split(LR_x_poly,D2_y,test_size=0.2)
LRModel=LogisticRegression(C=10).fit(LR_x_train,LR_y_train)
LR_predictions=LRModel.predict(LR_x_test)
print(f'The confusion matrix for LR Model is:\n\n{confusion_matrix(LR_y_test,LR_predictions)}')
print(f'The Classification Report for LR Model
is:\n\n{classification_report(LR_y_test,LR_predictions)}')

random_Model=DummyClassifier(strategy="uniform").fit(D2_X,D2_y)
random_predictions=random_Model.predict(D2_X)
print(f'The confusion matrix for Random Model
is:\n\n{confusion_matrix(D2_y,random_predictions)}')
print(f'The Classification Report for Random Model
is:\n\n{classification_report(D2_y,random_predictions)}')
most_frequent_Model=DummyClassifier(strategy="most_frequent").fit(D2_X,D2_y)
most_frequent_predictions=most_frequent_Model.predict(D2_X)
print(f'The confusion matrix for Most Frequent Model
is:\n\n{confusion_matrix(D2_y,most_frequent_predictions)}')
print(f'The Classification Report for Most Frequent Model
is:\n\n{classification_report(D2_y,most_frequent_predictions)}')

KNN_probability=KNNModel.predict_proba(D2_X)
most_frequent_probability=most_frequent_Model.predict_proba(D2_X)
```

```
random_probability=random_Model.predict_proba(D2_X)
fpr, tpr, _ = roc_curve(D2_y, LRModel.decision_function(LR_x_poly))
knn_fpr, knn_tpr, thresh = roc_curve(D2_y, KNN_probability[:, 1])
most_freq_fpr, most_freq_tpr, thresh = roc_curve(D2_y, most_frequent_probability[:, 1])
rand_fpr, rand_tpr, thresh = roc_curve(D2_y, random_probability[:, 1])

plt.rc('font', size=20)
plt.plot(fpr, tpr, color='cyan',label='Logistic Regression')
plt.plot(knn_fpr, knn_tpr, color='orange',label='kNN Model')
plt.plot(most_freq_fpr, most_freq_tpr, color='blue',label='Most Frequent Model')
plt.plot(rand_fpr, rand_tpr, color='red', label='Random Model')
plt.plot([0, 1], [0, 1], color='green', linestyle='--',label='baseline')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title(
   "ROC Curves for Dataset 2")
plt.legend()
# plt.show()
plt.savefig('ds2_d.png')
```