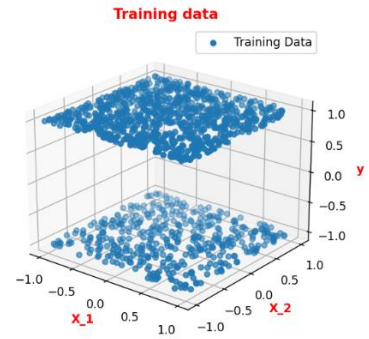
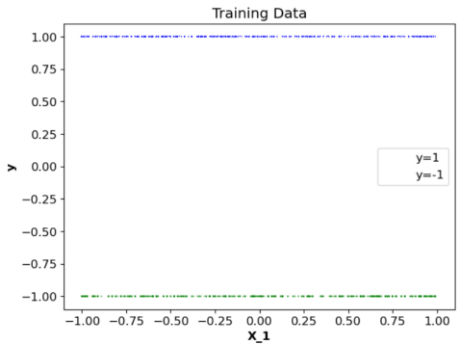
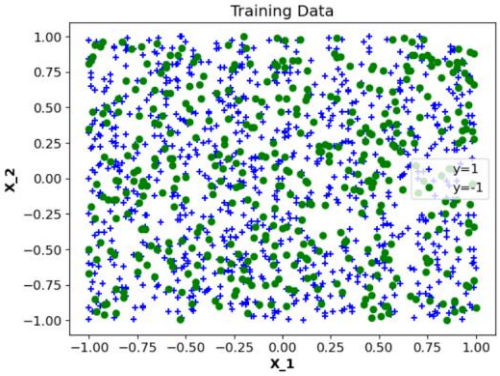
WEEK 4 ASSIGNMENT - REPORT

Input dataset: # id:19--19--19-1

Visualizing input dataset 1:







[(i) Start with the first dataset.

(a) Using sklearn to augment the two features in the dataset with polynomial features and train a Logistic Regression classifier with an L2 penalty added to the cost function. Use cross-validation to select (i) the maximum order of the 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/analyses to justify your choices. Use the ideas/tools you've learned in the module to help you with this. Be sure to include error bars in cross-validation plots. Remember to use a performance measure appropriate to classification (so probably not mean square error)]

The first step is to select the best suitable polynomial feature and weight C for the L2 penalty. To select them, the input dataset has been split into test and training datasets using sklearn's train_test_split(). Then, the model has been trained iteratively with different polynomial and C values. Finally, a final decision has been made based on the following metrics,

1. F1 score for 5-fold cross-validation
2. Area under the ROC curve

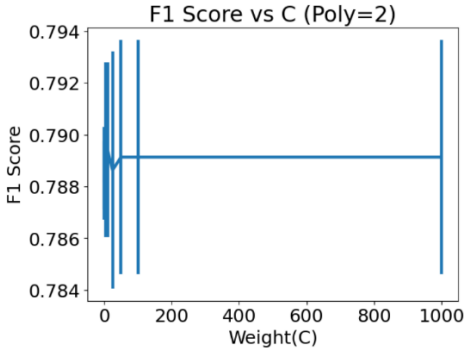
The following polynomial features and weight C have been considered for this experiment,

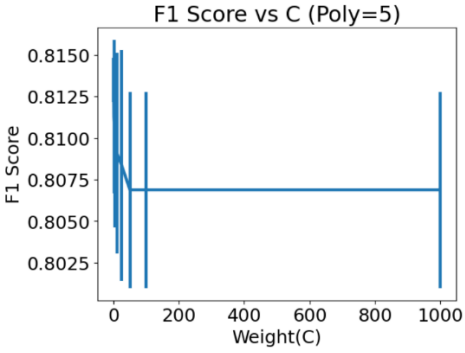
- Polynomial feature, [2, 5, 10]
- Weight C, [0.01, 0.1, 1, 5, 10, 25, 50, 100, 1000]

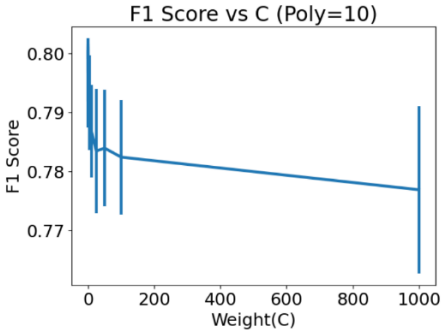
1) F1 score for 5-fold cross-validation:

Generally, the F1 score corresponds to the model's precision and recall (Harmonic mean of these parameters). So, a high F1 score means high precision and high recall. Considering this, the best model would be the one with a high F1 score, typically close to 1. The F1 score is calculated for each polynomial and weights for comparison.

- F1 score against different weights(C) for a polynomial with less than or equal to 2, 5, and 10.







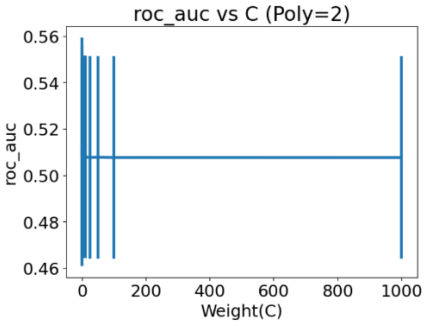
- The maximum F1 score for a polynomial with less than or equal to 2 is 0.795. [See Appendix, Pg: 9]
- The maximum F1 score for a polynomial with less than or equal to 5 is 0.816. [See Appendix, Pg: 9]
- The maximum F1 score for a polynomial with less than or equal to 10 is 0.805. [See Appendix, Pg: 9]

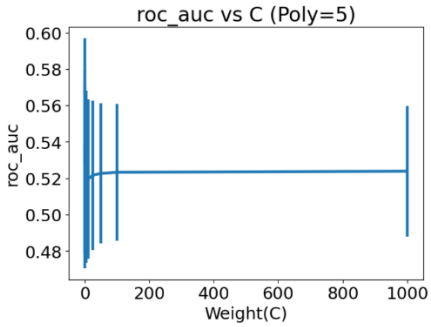
Thus, The F1 score for polynomial features less than or equal to 5 is the maximum when compared with others.

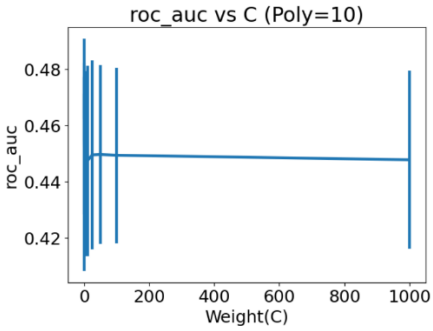
2) Area under the ROC curve:

Another useful metric to compare the models would be AUC, plotted against True positive rate and False positive rate. The higher the area under the curve the better the model. The AUC is calculated for each polynomial and weights for comparison.

- AUC for each weight (C) for a polynomial with less than or equal to 2, 5, and 10







- The maximum AUC score for a polynomial with less than or equal to 2 is 0.537. [See Appendix, Pg: 9]
- The maximum AUC score for a polynomial with less than or equal to 5 is 0.499. [See Appendix, Pg: 9]
- The maximum AUC score for a polynomial with less than or equal to 10 is 0.449. [See Appendix, Pg: 9]

Considering the above metrics, the F1 score is highest when using 5 polynomial features and 0.01 as a weight for the penalty. However, AUC is maximum when using 2 polynomial features and 0.01 as weight. As F1 and AUC use input data and predicted data as input respectively, And also considering the input data being highly imbalanced (will see under Logistic regression classifier). So I would go for F1 score. So polynomial 5 and weight 0.01 is selected for logistic regression.

Logistic regression classifier,

The logistic regression has been trained with degree 5 polynomial input features and L2 penalty weight 0.01. The model parameters, confusion and classification matrix are as follows,

Polynomial feature	5					
L2 Penalty, C	0.01					
Model Intercept	0.67050674					
Model Coefficient	1.90332E-06	-0.029151381	-0.045008102	0.005353637	-0.050209666	0.026781118
	-0.028165439	-0.030850677	-0.015390624	-0.04647373	-0.00350154	-0.018512798
	-0.008375647	-0.015656983	0.017376151	-0.035509653	-0.022093312	-0.011271963
	-0.020200587	-0.010913559	-0.036504468			
Confusion matrix	**TP =**	0	**FN =**	95		
	FP =	0	**TN =**	228		
Classification matrix		**precision**	**recall**	**f1-score**	**support**	
	-1	0	0	0	95	
	1	0.71	1	0.83	228	
	accuracy			0.71	323	
	macro avg	0.35	0.5	0.41	323	
	weighted avg	0.5	0.71	0.58	323	

Note: y=1 and y=-1 are heavily imbalanced. Examining the input data shows that out of 1290 records, 860 records are for y=1. This explains the imbalance.

[(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/analyses to justify your choices.]

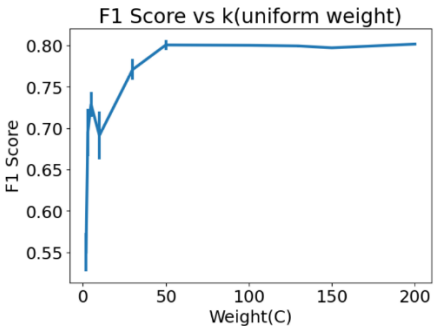
Here the first step would be to find the suitable k value for the kNN classifier, the model has been trained iteratively with different k values for both uniform and gaussian weights. Finally, a final decision has been made based on the following metrics,

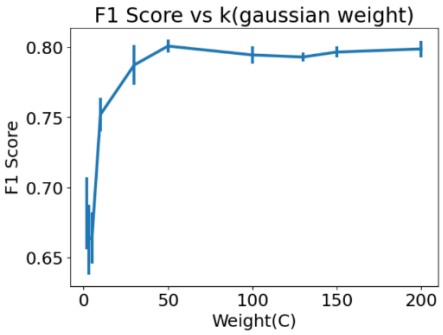
1. F1 score for 5-fold cross-validation
2. Area under the ROC curve

The following k values have been considered for this experiment,

- k Values, [2, 3, 5, 10, 30, 50, 100, 130, 150, 200]
- Weight C, [0.01, 0.1, 1, 5, 10, 25, 50, 100, 1000]

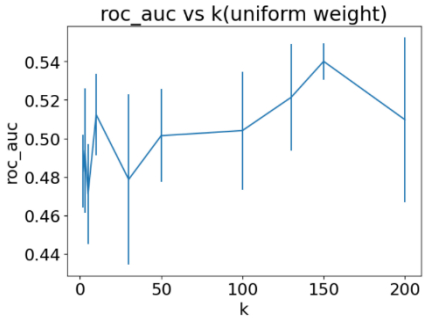
1) F1 score for 5-fold cross-validation:

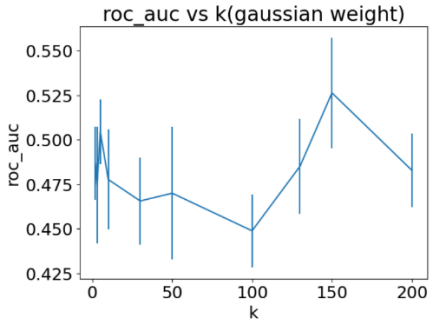




- The F1 score is maximum when k = 50, However, uniform and gaussian weights don't make any notable difference. Max(F1 score) is 0.800. [See Appendix, Pg: 9]

2) Area under the ROC curve:





- The AUC is a maximum of 0.54 and 0.526 for uniform and gaussian weights respectively when k = 150. [See Appendix, Pg: 9]

Thus, for this scenario, F1 value is considered important and k=50 is selected. To make keep the model simple, uniform weight has been selected as there are no significant differences have been seen.

kNN Classifier,

kNN classifier has been trained with k being 50 and uniform weight has been used. Confusion and classification matrix are,

K	50				
Weight	Uniform				
Confusion matrix	**TP =**	0	**FN =**	117	
	FP =	0	**TN =**	206	
Classification matrix		**precision**	**recall**	**f1-score**	**support**
	-1	0	0	0	117
	1	0.64	1	0.78	206
	accuracy			0.64	323
	macro avg	0.33	0.5	0.39	323
	weighted avg	0.44	0.67	0.5	323

[(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.]

Comparing the confusion matrix of the logistic classifier, kNN classifier, and Dummy classifier:

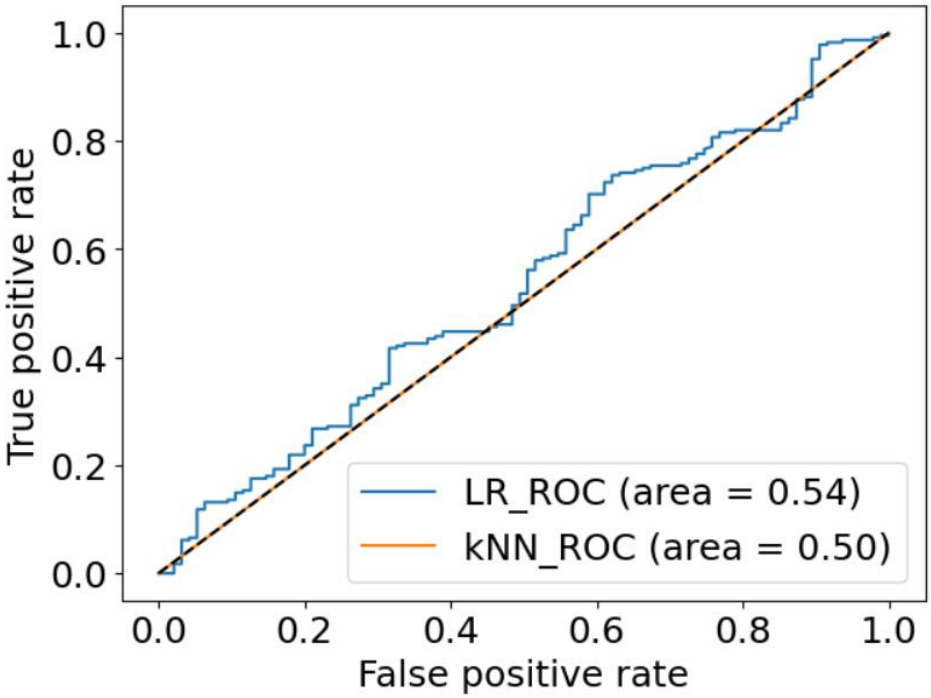
Logistic classifier, Confusion matrix	**TP =**	0	**FN =**	95
	FP =	0	**TN =**	228

kNN classifier, Confusion matrix	**TP =**	0	**FN =**	117
	FP =	0	**TN =**	206

Dummy classifier, Confusion matrix	**TP =**	0	**FN =**	95
	FP =	0	**TN =**	228

[(d) Plot the ROC curves for your trained Logistic Regression and kNN classifiers. Also plot the point(s) in 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.]

The ROC curve has been plotted below for the trained Logistics classifier and kNN classifier. A random classifier has been added to compare the performance of the classifiers,



[(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 your recommendation.]

Comparing the performance of the classifiers:

All 3 models performed well in predicting true negatives, this may also be due to the imbalance in the dataset. Also, none of the models predicted true positives and false positives. This may be due to the decision threshold being skewed, it also explains the high number of false negative predictions.

Is one classifier significantly better or worse than the other?

Both classifiers performed poorly, as is clear when comparing them against the dummy classifier. However, the logistic classifier is slightly better than kNN, as it predicts fewer false negatives than kNN classifier.

How do their ROC curves compare with that of a random classifier?

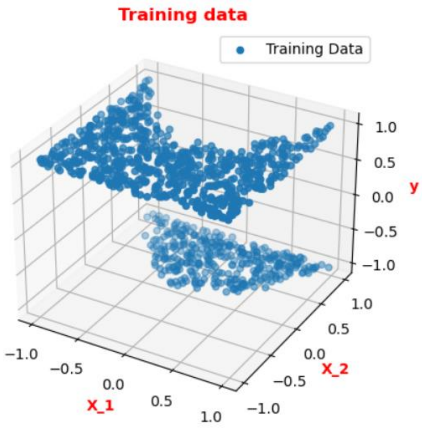
Both the classifier perform no better than the random classifier, While the ROC curve for kNN almost coincides with the random classifier. When considering the AUC, both classifiers have a minimum value (~0.5).

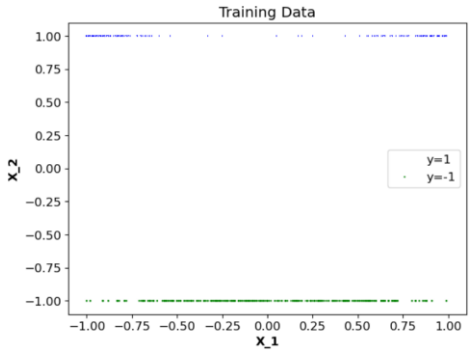
Which classifier, if any, would you recommend be used? Explain the reasoning behind your recommendation

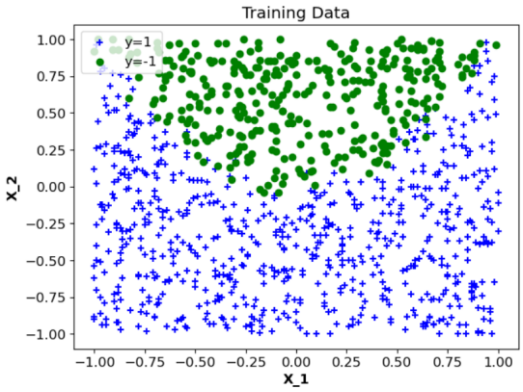
It's hard to recommend a classifier with the above data. Both the classifiers performed poorly even after cross-validation showing the inability of the algorithm to detect any useful pattern. This may be due to poor-quality input data.

[(ii) Repeat (i) for the second dataset]

Visualizing input dataset 2:







(ii)(a)

The first step is to select the best suitable polynomial feature and weight C for the L2 penalty. To select them, the input dataset has been split into test and training datasets using sklearn's train_test_split(). Then, the model has been trained iteratively with different polynomial and C values. Finally, a final decision has been made based on the following metrics,

1. F1 score for 5-fold cross-validation
2. Area under the ROC curve

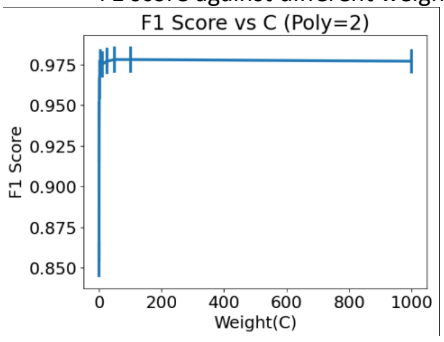
The following polynomial features and weight C have been considered for this experiment,

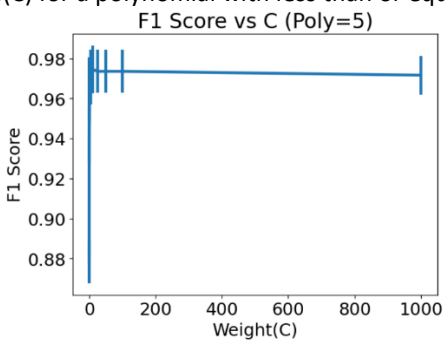
- Polynomial feature, [2, 5, 10]
- Weight C, [0.01, 0.1, 1, 5, 10, 25, 50, 100, 1000]

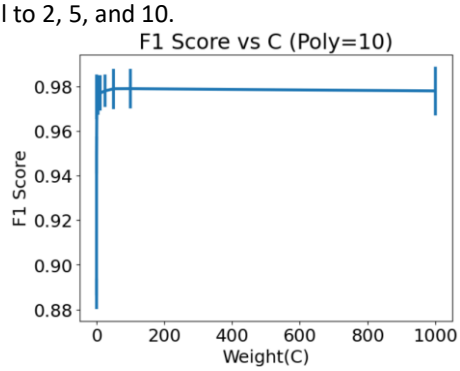
1) F1 score for 5-fold cross-validation:

Generally, the F1 score corresponds to the model's precision and recall (Harmonic mean of these parameters). So, a high F1 score means high precision and high recall. Considering this, the best model would be the one with a high F1 score, typically close to 1. The F1 score is calculated for each polynomial and weights for comparison.

- F1 score against different weights(C) for a polynomial with less than or equal to 2, 5, and 10.







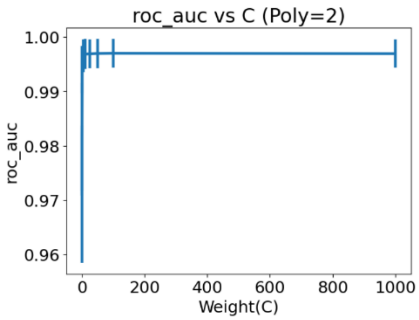
- The maximum F1 score for a polynomial with less than or equal to 2 is 0.978. [See Appendix, Pg: 9]
- The maximum F1 score for a polynomial with less than or equal to 5 is 0.974. [See Appendix, Pg: 9]
- The maximum F1 score for a polynomial with less than or equal to 10 is 0.972. [See Appendix, Pg: 9]

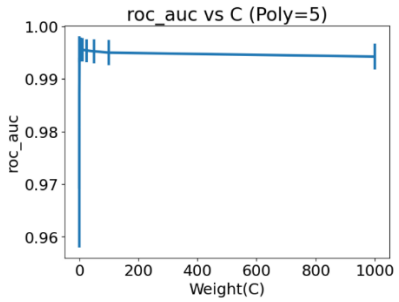
Thus, The F1 score for polynomial features less than or equal to 2 is the maximum when compared with others.

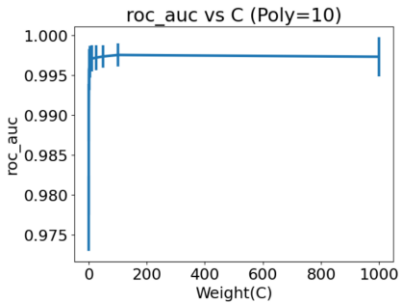
2) Area under the ROC curve:

Another useful metric to compare the models would be AUC, plotted against the True positive rate and False positive rate. The higher the area under the curve the better the model. The AUC is calculated for each polynomial and weights for comparison.

- AUC for each weight (C) for a polynomial with less than or equal to 2, 5, and 10







- The maximum AUC score for a polynomial with less than or equal to 2 is 0.997. [See Appendix, Pg: 9]
- The maximum AUC score for a polynomial with less than or equal to 5 is 0.995. [See Appendix, Pg: 9]
- The maximum AUC score for a polynomial with less than or equal to 10 is 0.995. [See Appendix, Pg: 9]

Considering the above metrics, both the F1 score and AUC are maximum for polynomial degree 2 and penalty weight 100.

Logistic regression classifier,

The logistic regression has been trained with degree 2 polynomial input features and L2 penalty weight 100. The model parameters, confusion and classification matrix are as follows,

Polynomial feature	2				
L2 Penalty, C	100				
Model Intercept	-0.48484664				
Model Coefficient	-2.29E-07	1.050195594	-19.07612462	22.04691572	-0.617146453
	-2.909375137				
Confusion matrix	**TP =**	77	**FN =**	3	
	FP =	6	**TN =**	173	
Classification matrix		**precision**	**recall**	**f1-score**	**support**
	-1	0.93	0.96	0.94	80
	1	0.98	0.97	0.97	179
	accuracy			0.97	259
	macro avg	0.96	0.96	0.96	259
	weighted avg	0.97	0.97	0.97	259

Note: y=1 and y=-1 are imbalanced. Examining the input data shows that out of 1036 records, 693 records are for y=1. This explains the imbalance.

(ii)(b)

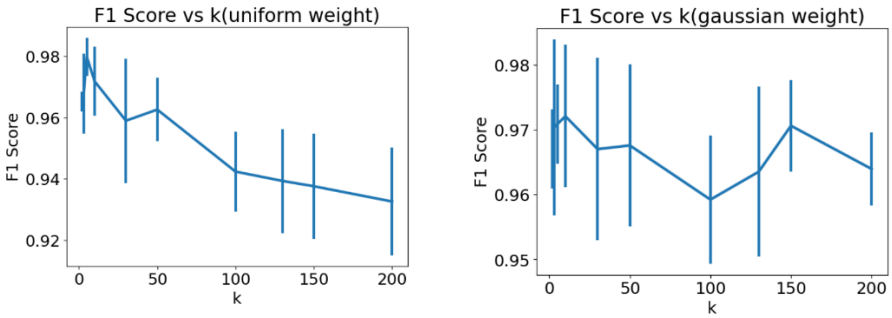
Here the first step would be to find the suitable k value for the kNN classifier, the model has been trained iteratively with different k values for both uniform and gaussian weights. Finally, a final decision has been made based on the following metrics,

1. F1 score for 5-fold cross-validation
2. Area under the ROC curve

The following k values have been considered for this experiment,

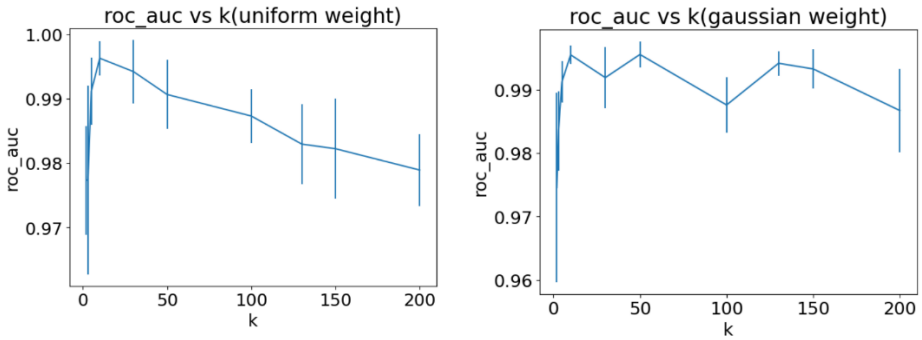
- k Values, [2, 3, 5, 10, 30, 50, 100, 130, 150, 200]
- Weight C, [0.01, 0.1, 1, 5, 10, 25, 50, 100, 1000]

1) F1 score for 5-fold cross-validation:



- The F1 score is maximum in both uniform and gaussian weights when k = 10. However, uniform and gaussian weights don't make any notable difference. The maximum F1 score is 0.972. [See Appendix, Pg: 9]

3) Area under the ROC curve:



- The AUC is a maximum of 0.996 and 0.995 for uniform and gaussian weights respectively when k = 10. [See Appendix, Pg: 9]

Thus, k=10 is selected for kNN classification.

kNN Classifier,

kNN classifier has been trained with k being 10 and gaussian weight has been used. Confusion and classification matrix are,

K	50				
Weight	Gaussian				
Confusion matrix	**TP =**	84	**FN =**	5	
	FP =	6	**TN =**	164	
Classification matrix		**precision**	**recall**	**f1-score**	**support**
	-1	0.93	0.94	0.94	89
	1	0.97	0.96	0.97	170
	accuracy			0.96	259
	macro avg	0.95	0.95	0.95	259
	weighted avg	0.44	0.67	0.5	323

Gaussian weight is chosen after comparing the confusion matrix of both the weight. [see Appendix Pg: 10].

(ii)(c)

Comparing the confusion matrix of the logistic classifier, kNN classifier, and Dummy classifier:

Logistic classifier, Confusion matrix	**TP =**	77	**FN =**	3
	FP =	6	**TN =**	173

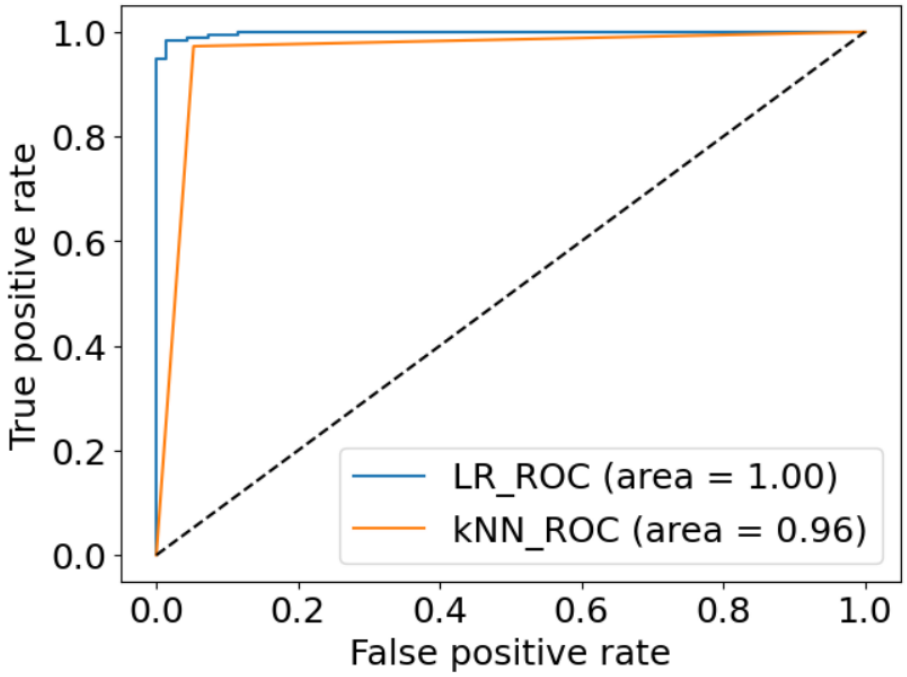
kNN classifier, Confusion matrix	**TP =**	84	**FN =**	5
	FP =	6	**TN =**	164

Dummy classifier, Confusion matrix	**TP =**	0	**FN =**	80

	FP =	0	**TN =**	179

(ii)(d)

The ROC curve has been plotted below for the trained Logistics classifier and kNN classifier. A random classifier has been added to compare the performance of the classifiers,



(ii)(e)

Comparing the performance of the classifiers:

Both the logistic classifier and kNN classifier performed exceptionally well. Both models predicted more true negatives than TP, this may also be due to the presence of more negative input data than positive ones. Logistic classifier defected fewer false negatives than kNN.

Is one classifier significantly better or worse than the other?

Even though both models performed well compared to dummy classifiers, depending on the context logistic classification would be better than kNN. Because the logistic classifier predicted fewer false negatives. However, both models predicted equal number of false positives.

How do their ROC curves compare with that of a random classifier?

AUC for the logistic classifier ROC curve is ideal, while kNN's AUC falls behind slightly.

Which classifier, if any, would you recommend be used? Explain the reasoning behind your recommendation

While considering both the confusion matrix and ROC curve, both models can be used for this dataset (or application). Depending on the context of the classification, we can decide between logistics and kNN. For example, If it is anything related to epidemic management, then a logistic classifier would be recommended as it has fewer FN.

APPENDIX:

F1 score and AUC comparison for dataset 1:

Polynomial feature	2								
Penalty weight	0.01	0.1	1	5	10	25	50	100	1000
Mean F1 in 5-fold CV	0.78997	0.78997	0.78997	0.79495	0.79495	0.79495	0.79495	0.79495	0.79495
AUC	0.53699	0.53752	0.53432	0.53344	0.53347	0.53337	0.53339	0.53339	0.53337

Polynomial feature	5								
Penalty weight	0.01	0.1	1	5	10	25	50	100	1000
Mean F1 in 5-fold CV	0.81595	0.81595	0.81595	0.81595	0.81595	0.81595	0.81346	0.81346	0.81346
AUC	0.49944	0.47424	0.46851	0.47674	0.47778	0.47870	0.47932	0.48016	0.48095

Polynomial feature	10								
Penalty weight	0.01	0.1	1	5	10	25	50	100	1000
Mean F1 in 5 fold CV	0.80124	0.80503	0.80255	0.80124	0.79751	0.80000	0.80000	0.80000	0.80000
AUC	0.47666	0.45524	0.43964	0.44631	0.44740	0.44948	0.44960	0.44936	0.44774

F1 score vs k values comparison for dataset1:

Uniform weight:

k values	2	3	5	10	30	50	100	130	150	200
F1 value	0.55020	0.69499	0.72877	0.69129	0.77087	0.80070	0.80025	0.79950	0.79726	0.80173
AUC	0.48298	0.49379	0.47123	0.51237	0.47874	0.50145	0.50409	0.52136	0.54005	0.50976

Gaussian weight:

k values	2	3	5	10	30	50	100	130	150	200
F1 value	0.68152	0.66286	0.66401	0.75191	0.78726	0.80074	0.79444	0.79298	0.79650	0.79873
AUC	0.48659	0.47449	0.50455	0.47763	0.46559	0.46995	0.44885	0.48508	0.52626	0.48279

F1 score and AUC comparison for dataset 2:

Polynomial feature	2								
Penalty weight	0.01	0.1	1	5	10	25	50	100	1000
Mean F1 in 5-fold CV	0.85494	0.93368	0.96580	0.97617	0.97519	0.97710	0.97805	0.97805	0.97707
AUC	0.97110	0.98168	0.99399	0.99656	0.99686	0.99690	0.99698	0.99701	0.99694

Polynomial feature	5								
Penalty weight	0.01	0.1	1	5	10	25	50	100	1000
Mean F1 in 5-fold CV	0.87852	0.94162	0.96425	0.97068	0.97449	0.97355	0.97355	0.97355	0.97161
AUC	0.96940	0.98745	0.99568	0.99553	0.99553	0.99539	0.99524	0.99498	0.99424

Polynomial feature	10								
Penalty weight	0.01	0.1	1	5	10	25	50	100	1000
Mean F1 in 5 fold CV	0.87777	0.94073	0.96632	0.96905	0.96910	0.96905	0.97105	0.97105	0.97213
AUC	0.97844	0.99060	0.99477	0.99561	0.99546	0.99538	0.99534	0.99515	0.99413

F1 score vs k values comparison for dataset2:

Uniform weight:

k values	2	3	5	10	30	50	100	130	150	200
F1 value	0.95918	0.96526	0.96949	0.97394	0.96223	0.95759	0.94475	0.94051	0.94258	0.93331
AUC	0.97735	0.97740	0.99116	0.99627	0.99421	0.99065	0.98731	0.98296	0.98224	0.97893

Gaussian weight:

k values	2	3	5	10	30	50	100	130	150	200
F1 value	0.96704	0.97038	0.97087	0.97210	0.96702	0.96759	0.95923	0.96354	0.97063	0.96396
AUC	0.97459	0.98350	0.99128	0.99552	0.99192	0.99557	0.98761	0.99418	0.99329	0.98673

Python Code:

```
#Importing input
import numpy as np
import pandas as pd
df=pd.read_csv("Week4 Dataset1.csv", skiprows=[0], names=["X1","X2","y"])
print(df.head())
X1=df.iloc[:,0]
X2=df.iloc[:,1]
X=np.column_stack((X1,X2))
y=df.iloc[:,2]

#Knowing the input data
print(np.shape(df))
print(min(X1), max(X1))
print(min(X2), max(X2))
print(min(y), max(y))
#Visualizing the input datsets
# %matplotlib widget
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X1,X2,y, label = "Training Data")
ax.set_xlabel("X_1", fontweight ='bold').set_color('red')
ax.set_ylabel("X_2", fontweight ='bold').set_color('red')
ax.set_zlabel("y", fontweight ='bold').set_color('red')
ax.legend()
# ax.view_init(0, 90)
ax.set_title("Training data", fontweight ='bold').set_color('red')
import matplotlib.pyplot as plt
plt.figure()
plt.rc('font', size =12)
plt.rcParams['figure.constrained_layout.use'] = True
plt.scatter(X1[y==1],X2[y==1], c='b',marker='+', label='y=1')
plt.scatter(X1[y==-1], X2[y==-1], c='g',marker='o', label='y=-1')
# plt.scatter(X1[y==-1],X2[y==-1],c='b',marker='o', label=r'$y_t=-1$')
plt.ylabel('X_2', fontweight ='bold')
plt.xlabel('X_1', fontweight ='bold')
plt.legend()
plt.title("Training Data")
plt.show()
def splitNaddPoly(X, y, n=5, gridRange=2):
    ####Splitting the model into train, and test
    from sklearn.model_selection import train_test_split
    Xtrain, Xtest, ytrain, ytest = train_test_split(X,y) # --> Return ytrain, ytest
```

```
    ####Adding extra polynomial features equal to all combinations of powers of the two
features up to power n
    from sklearn.preprocessing import PolynomialFeatures
    Xtrain_poly = PolynomialFeatures(n).fit_transform(Xtrain) # --> Return Xtrain_poly
    Xtest_poly = PolynomialFeatures(n).fit_transform(Xtest) # --> Return Xtest_poly
    X_poly = PolynomialFeatures(n).fit_transform(X)

    ####Grid of feature values, to use for predictions
    # Xt=[]
    # grid=np.linspace(-gridRange,gridRange)
    # for i in grid:
    #     for j in grid:
    #         Xt.append([i,j])
    # Xt = np.array(Xt) #### Return Xt
    # Xtest = PolynomialFeatures(n).fit_transform(Xt) #### Return Xtest

    return Xtrain_poly, Xtest_poly, ytrain, ytest
def logisticRegression(Xtrain_poly, Xtest_poly, ytrain, ytest, c):
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression(penalty='l2', C= c,max_iter=10000).fit(Xtrain_poly, ytrain)
    prediction = model.predict(Xtest_poly)
    return model, prediction
def modelResults(testY,model, prediction,c):
    print("C = ", c)
    print(f"Model Intercept = {model.intercept_}")
    print(f"Model Coefficient = {model.coef_.tolist()}")
    from sklearn.metrics import mean_squared_error, confusion_matrix,
classification_report

    from sklearn.model_selection import cross_val_score
    scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')
    # print(scores)
    print("Accuracy (for 5 fold cross validation): %0.2f (+/- %0.2f)" % (scores.mean(),
scores.std()))

    print("Confusion matrix:")
    print(confusion_matrix(testY,prediction))
    print("Classification report:")
    print(classification_report(testY,prediction))
def dummyRegressor(Xtrain_poly, Xtest_poly, ytrain, ytest):
    from sklearn.dummy import DummyRegressor
    dummy = DummyRegressor(strategy='mean').fit(Xtrain_poly, ytrain)
    ydummy = dummy.predict(Xtest_poly)
    from sklearn.metrics import mean_squared_error
    print('Mean square error (Dummy) = %f'%(mean_squared_error(ytest,ydummy)))
def dummyClassifier(Xtrain_poly, Xtest_poly, ytrain, ytest):
   from sklearn.dummy import DummyClassifier
   dummy = DummyClassifier(strategy='most_frequent').fit(Xtrain_poly, ytrain)
   ydummy = dummy.predict(Xtest_poly)
```

```
   from sklearn.metrics import confusion_matrix, classification_report
   print("Confusion matrix:")
   print(confusion_matrix(ytest, ydummy))
   print("Classification report:")
   print(classification_report(ytest, ydummy))
def errorBar(Xtrain_poly, Xtest_poly, ytrain, ytest):
    mean_error=[]
    std_error=[]
    f1 = []
    Ci_range = [0.01, 0.1, 1, 5, 10, 25, 50, 100, 1000]
    for Ci in Ci_range:
        model, prediction = logisticRegression(Xtrain_poly, Xtest_poly, ytrain, ytest, Ci)
        from sklearn.model_selection import cross_val_score
        scores = cross_val_score(model, Xtrain_poly, ytrain, cv=5, scoring='f1')
        # f1.append(max(scores))
        mean_error.append(np.array(scores).mean())
        std_error.append(np.array(scores).std())

    print(Ci_range)
    print(mean_error)

    import matplotlib.pyplot as plt
    plt.rc('font', size=18); plt.rcParams['figure.constrained_layout.use'] = True
    plt.errorbar(Ci_range,mean_error,yerr=std_error,linewidth=3)

    plt.xlabel('Weight(C)'); plt.ylabel('F1 Score')
    # plt.xlim((0,1))
    plt.title("F1 Score vs C (Poly=2)")
    plt.show()
    # return f1
def kNN_CV(X,y):
    mean_error=[]
    std_error=[]
    f1 = []
    K_range = [2,3,5,10,30,50,100, 130, 150, 200]
    for k in K_range:
        from sklearn.model_selection import train_test_split
        Xtrain, Xtest, ytrain, ytest = train_test_split(X,y)
        from sklearn.neighbors import KNeighborsClassifier
        model = KNeighborsClassifier(n_neighbors=k,weights='distance').fit(Xtrain, ytrain)
        from sklearn.model_selection import cross_val_score
        scores = cross_val_score(model, Xtrain, ytrain, cv=5, scoring='f1')
        f1.append(max(scores))
        mean_error.append(np.array(scores).mean())
        std_error.append(np.array(scores).std())
    print(K_range)
    print(mean_error)
    import matplotlib.pyplot as plt
    plt.rc('font', size=18); plt.rcParams['figure.constrained_layout.use'] = True
```

```
    plt.errorbar(K_range,mean_error,yerr=std_error,linewidth=3)
    plt.xlabel('k'); plt.ylabel('F1 Score')
    # plt.xlim((0,1))
    plt.title("F1 Score vs k(uniform weight)")
    plt.show()

    # return f1
def kNN(X, y, k):
    from sklearn.model_selection import train_test_split
    Xtrain, Xtest, ytrain, ytest = train_test_split(X,y)
    Xtrain, Xtest, ytrain, ytest
    from sklearn.neighbors import KNeighborsClassifier
    model = KNeighborsClassifier(n_neighbors=k,weights='uniform').fit(Xtrain, ytrain)
    prediction = model.predict(Xtest)
    return model, prediction, Xtest, ytest
def kNN_Metrics(ytest,prediction):
    from sklearn.metrics import confusion_matrix, classification_report
    print("Confusion matrix (kNN):")
    print(confusion_matrix(ytest,prediction))
    print("Classification report (kNN):")
    print(classification_report(ytest,prediction))
def rocCurve(model,Xtest_poly, ytest):
    # model, prediction = logisticRegression(Xtrain_poly, Xtest_poly, ytrain, ytest, 10)
    from sklearn.metrics import roc_curve,auc
    fpr, tpr, _ = roc_curve(ytest,model.decision_function(Xtest_poly))
    import matplotlib.pyplot as plt
    roc_auc = auc(fpr,tpr)
    plt.plot(fpr,tpr,label='ROC curve (area = %0.2f)' % roc_auc)
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.legend(loc="lower right")
    plt.show()
    return fpr, tpr, roc_auc
Xtrain_poly, Xtest_poly, ytrain, ytest = splitNaddPoly(X, y, 5)
mean_error=[]
std_error=[]
f1 = []
Ci_range = [0.01, 0.1, 1, 5, 10, 25, 50, 100, 1000]
for Ci in Ci_range:
    model, prediction = logisticRegression(Xtrain_poly, Xtest_poly, ytrain, ytest, Ci)
    from sklearn.model_selection import cross_val_score
    scores = cross_val_score(model, Xtrain_poly, ytrain, cv=5, scoring='roc_auc')
    # f1.append(max(scores))
    mean_error.append(np.array(scores).mean())
    std_error.append(np.array(scores).std())

print(Ci_range)
print(mean_error)
```

```
import matplotlib.pyplot as plt
plt.rc('font', size=18); plt.rcParams['figure.constrained_layout.use'] = True
plt.errorbar(Ci_range,mean_error,yerr=std_error,linewidth=3)

plt.xlabel('Weight(C)'); plt.ylabel('roc_auc')
# plt.xlim((0,1))
plt.title("roc_auc vs C (Poly=2)")
plt.show()
# to get polynomial and penalty weight value
f1_values = errorBar(Xtrain_poly, Xtest_poly, ytrain, ytest)
print(f1_values)
model1, prediction1 = logisticRegression(Xtrain_poly, Xtest_poly, ytrain, ytest, 0.01)
modelResults(ytest, model1, prediction1, 0.01)
dummyClassifier(Xtrain_poly, Xtest_poly, ytrain, ytest)
Lfpr, Ltpr, Lroc_auc = rocCurve(model1,Xtest_poly, ytest)
kNN_CV(X,y)
model, prediction, Xtest, ytest = kNN(X,y,50)
kNN_Metrics(ytest,prediction)
def rocCurve(model,ytest,prediction):
    from sklearn.metrics import roc_curve,auc
    fpr, tpr, _ = roc_curve(ytest,prediction,)
    roc_auc = auc(fpr,tpr)
    import matplotlib.pyplot as plt
    plt.plot(fpr,tpr,label='ROC curve_ (area = %0.2f)' % roc_auc)
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.legend(loc="lower right")
    plt.show()
    return fpr, tpr, roc_auc

kfpr, ktpr, kauc = rocCurve(model,ytest,prediction)
def combinedROC(Lfpr, Ltpr, Lroc_auc, kfpr, ktpr, kauc):
    import matplotlib.pyplot as plt
    plt.plot(Lfpr, Ltpr,label='LR_ROC (area = %0.2f)' % Lroc_auc)
    plt.plot(kfpr, ktpr,label='kNN_ROC (area = %0.2f)' % kauc)
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.legend(loc="lower right")
    plt.show()


combinedROC(Lfpr, Ltpr, Lroc_auc, kfpr, ktpr, kauc)
mean_error=[]
std_error=[]
f1 = []
K_range = [2,3,5,10,30,50,100, 130, 150, 200]
for k in K_range:
    from sklearn.model_selection import train_test_split
```

```
    Xtrain, Xtest, ytrain, ytest = train_test_split(X,y)
    from sklearn.neighbors import KNeighborsClassifier
    model = KNeighborsClassifier(n_neighbors=k,weights='uniform').fit(Xtrain, ytrain)
    from sklearn.model_selection import cross_val_score
    scores = cross_val_score(model, Xtrain, ytrain, cv=5, scoring='roc_auc')
    f1.append(max(scores))
    mean_error.append(np.array(scores).mean())
    std_error.append(np.array(scores).std())
print(K_range)
print(mean_error)
import matplotlib.pyplot as plt
plt.rc('font', size=18); plt.rcParams['figure.constrained_layout.use'] = True
plt.errorbar(K_range,mean_error,yerr=std_error)
plt.xlabel('k'); plt.ylabel('roc_auc')
# plt.xlim((0,1))
plt.title("roc_auc vs k(uniform weight)")
plt.show()
```