Assignment 3 – Week 4

Pallavit Aggarwal – Ms IS
ID: 22333721

a)

For Dataset 1 - # id:6--6--6-1

The dataset 1 consists of targets -1 and 1, but is **imbalanced**.

```
+1 targets = 719 , -1 targets = 326
```

This imbalance is evident from the values getting skewed towards +1 more than -1.
This data visualised on a 2d and 3d scatter is given below in Fig1 and Fig2

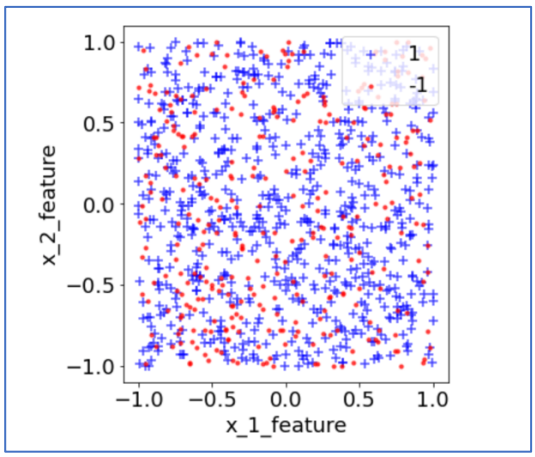


Fig1

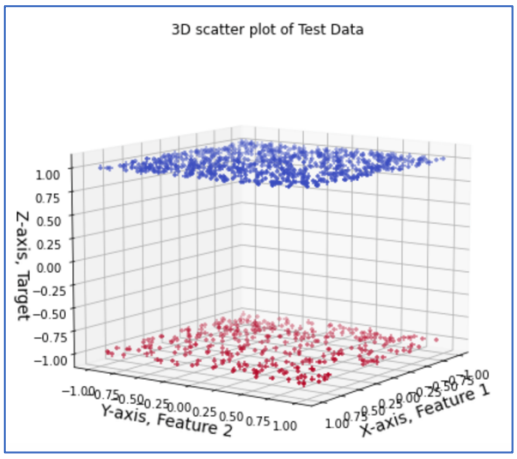
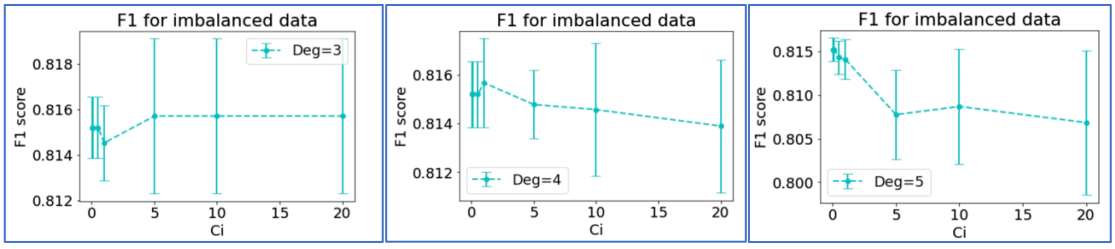
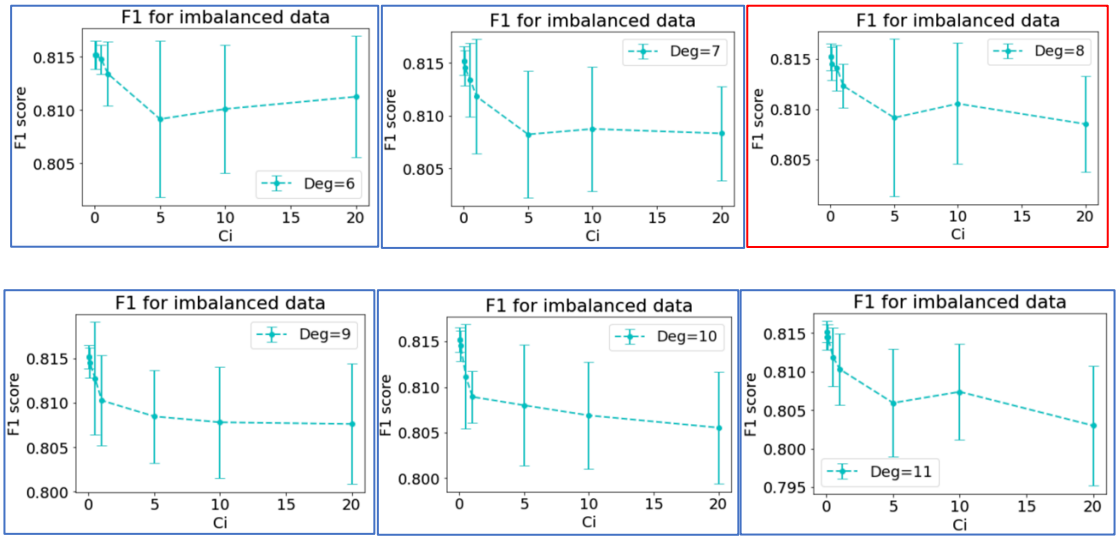


Fig 2

Figure 2 shows the dispersion of +1 and -1 target values along the z axis based on different values of Feature 1 and Feautre 2 plotted on the X-Y plane as two points on the plane.

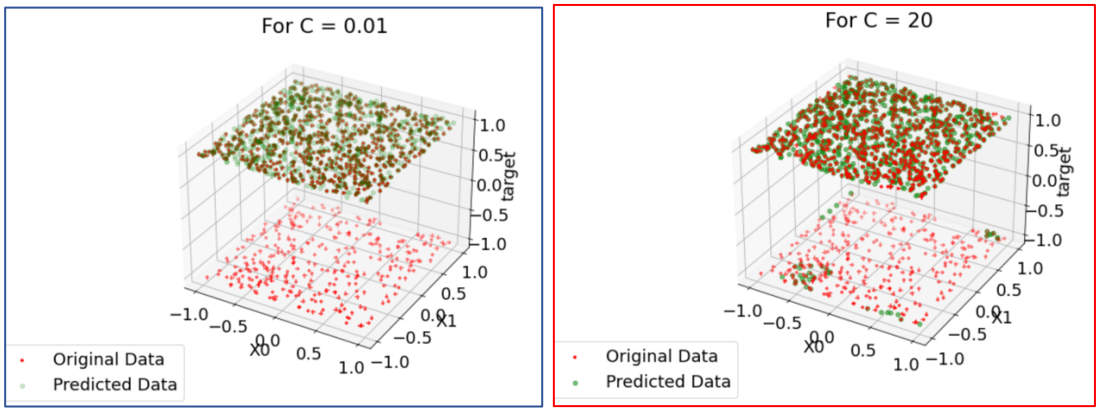
Now using the Logistic Regression with a penalty 2, from the sklearn.linear_model import LogisticRegression, I loop over the following degree values [3,4,5,6,7,8,9,10,11] and for each degree value create a polynomial feature of that value and loop over a set of C values given as [0.01,0.05,0.1,0.5,1,5,10,20]. For each C value inside each degree value, I calculate the cross_val_score using the score as 'F1' since this is an imbalanced data set, and using 'accuracy' or simply using the mean squared error values would not give us the appropriate idea of the performance of the model





As we can infer from the F1 score plots against the C values on the X-axis, the value of F1 score is the maximum for a low value of C and the standard deviation is also less. A low standard deviation initialized as the yerr of the errorbar plot is desirable as the value of F1 score should not fluctuate over a wider range for stabler model.

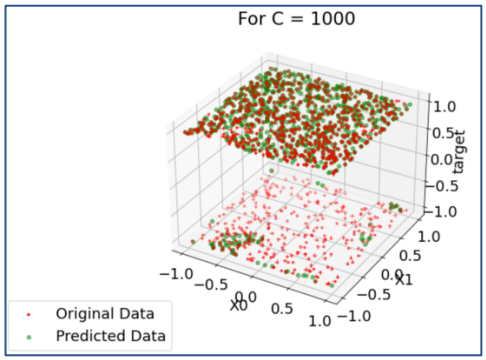
However, for a selected value of degree in this case : Degree = 8 , we plot the following scatters of predictions in 3d for the values of C given.



In the above figures when C=0.01 (Since C is inversely proportional to alpha or the strength of the Regularizer) the weights of the parameters in this case for all the features of degree 8 are penalized heavily i.e. Lower C means higher regularization strength.

On the contrary, a higher value of C = 20, shows some points getting predicted for the case -1. In the above figures, Red is used for Original Data and Green for Predicted Data.

An even higher value of C=1000 shows that there are some predictions that are in the -1 plane. This could be a case of over-fitting as the parameters are too high in case of degree 8, and the model is not getting penalized, which means that model is learning to use the idiosyncrasies of the data points that it has been trained on.

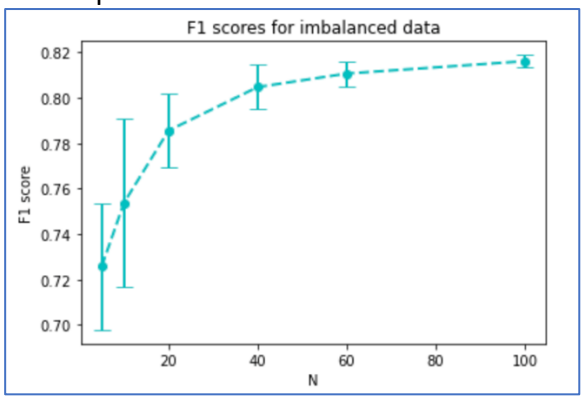


Example of C=1000, regularizer strength is low, and overfitting could be caused.

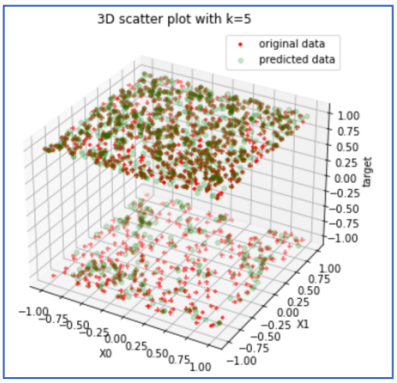
As per the values of the predictions from the above 3D scatter plots in conjunction with the F1 scores and errorbar plots, the value of C=20 appears to give a more realistic fit to the model. C=0.01 though has good accuracy and less std-err but it doesn't give a good Prediction scatter. For real life data, C=0.01 maybe more penalized than usual.

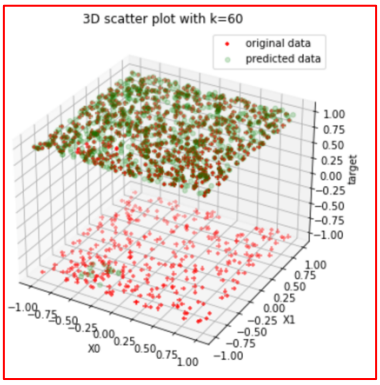
b)

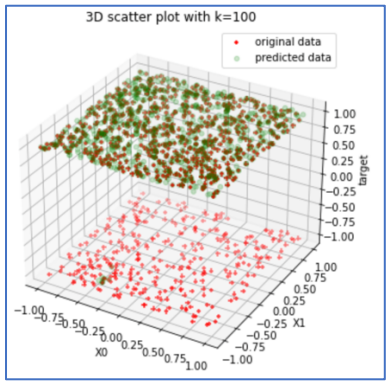
Now, Training a KNN classifier on the above imbalanced dataset, using sklearn.neighbors import KNeighborsClassifier for different values of N in [5,10,20,40,60,100].
The below image shows the plot of F1 score over the changing N values done over a KFold of 5, cross_val_score with params cv=5, scoring='f1'. The KNN weights param is set to default value 'uniform'. metric default is "minkowski" which when p=2, is equivalent to Euclidian distance and Manhattan when p=1.



N>40 shows a plateauing F1 score with minimal standard deviation in the score. N=60 and N=100 seem to be the most ideal values of N for the given dataset.







As we can see, the scatter plot for K=60 shows that the model is not too overfitted as in the case of k=100. There are relatively higher predictions in the -1 z-axis plane for K=60 than K=100.

A middle value of k=60, seems more suitable for the model for unknown data predictions.

c)

To investigate this deeper, we look at confusion matrices which is a tabular summary of the number of correct and incorrect predictions made by a classifier.

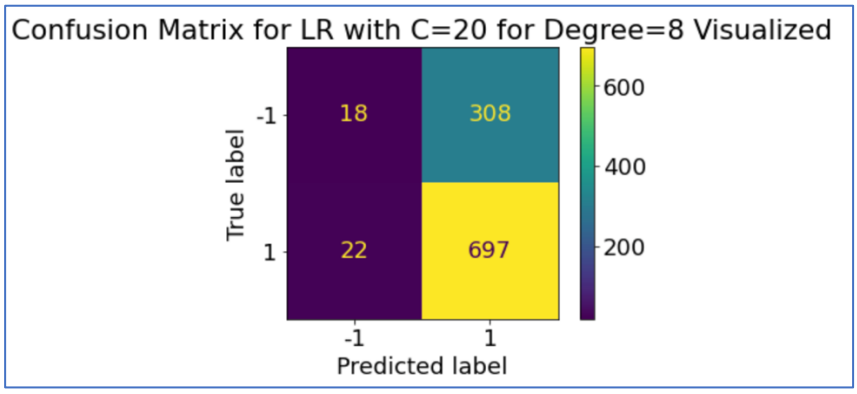
```
Confusion Matrix for LR with Penalty L2 and C=20 for Degree=8
[[697  22]
 [308  18]]


Confusion Matrix for Dummy Classifier with strategy=Stratified for Degree=8
[[107 219]
 [223 496]]


Confusion Matrix for Dummy Classifier with strategy=most_frequent for Degree=8
[[  0 326]
 [  0 719]]
```

Confusion matrices for LR, Dummy(stratified) and Dummy(most frequent)

```
Confusion Matrix for KNN with k=60 with No Polynomial Features
[[  5 321]
 [  2 717]]


Confusion Matrix for Dummy Classifier with strategy=most_frequent for Degree=0
[[  0 326]
 [  0 719]]


Confusion Matrix for Dummy Classifier with strategy=Stratified for Degree=0
[[108 218]
 [221 498]]
```

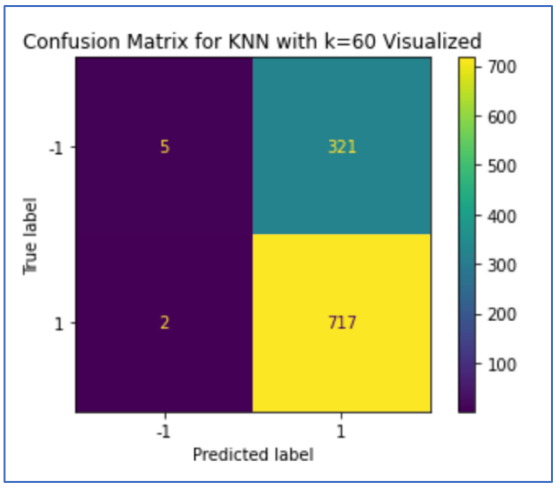
Confusion matrices for KNN, Dummy(most frequent) and Dummy(stratified)

Now,



TP=1,1(yellow,697) ; TN=-1,-1(purple,18) ; FP=1,-1(blue,308) ; FN=-1,1(purple,22)

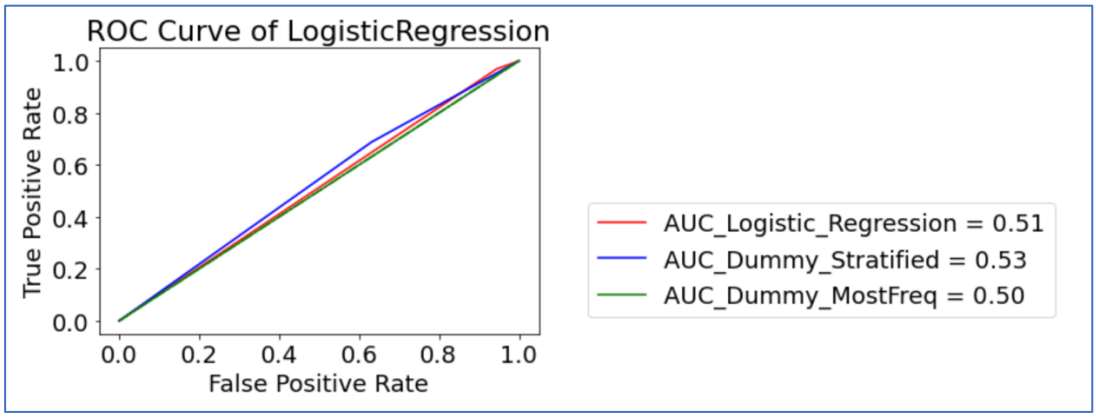
Accuracy = 0.68 ; Precision = 0.69 ; Recall = 0.94 ; F1-Score = 0.79 ; Specificity = 0.05

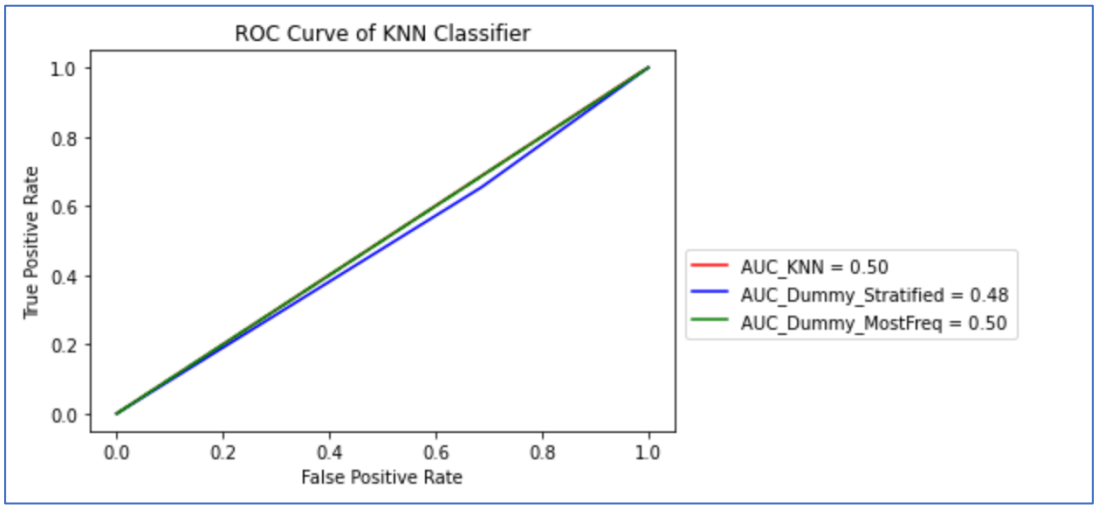


TP=1,1(yellow,717) ; TN=-1,-1(purple,5) ; FP=1,-1(blue,321) ; FN=-1,1(purple,2)

Accuracy = 0.68 ; Precision = 0.69 ; Recall = 0.99 ; F1-Score = 0.81 ; Specificity = 0.01

d)





e)

The results for LR show that for the defined dataset, the ROC curve is the same as the default line of True Positive vs False Positive i.e a random classifier. The ROC curves tells us that no threshold value is good enough for the predictions.
The AUC of logistic regression is almost equal to AUC of KNN for this dataset and the values I considered. Dummy Stratified gives 2 different values of AUC when run differently, and therefore suggests randomness and instability in desired output (hit-miss).

All the classifiers are as good as a coin-toss.

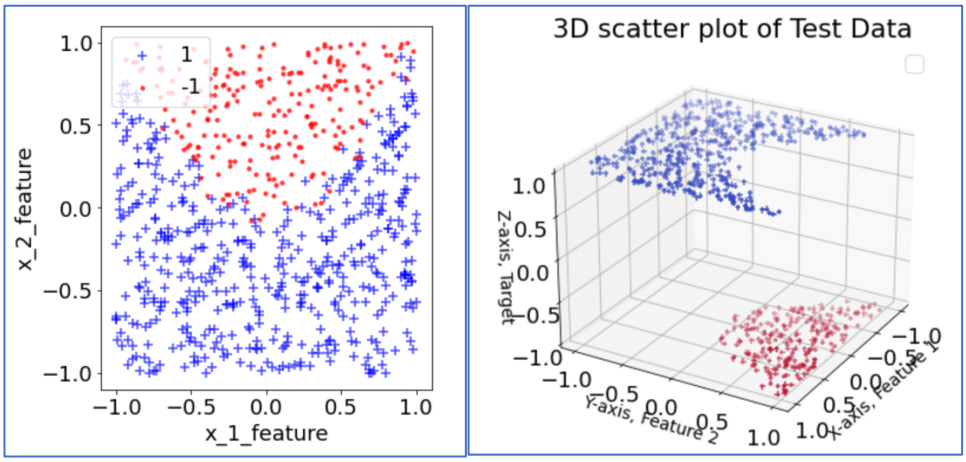
---

a)

For dataset: # id:6-6--6-1 w4 -d2

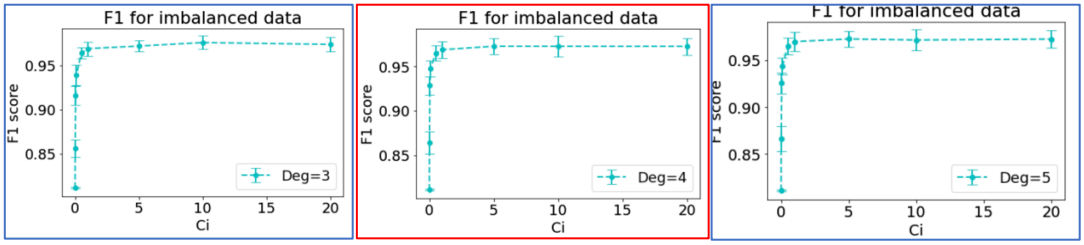
```
+1 targets = 493 , -1 targets = 230
```

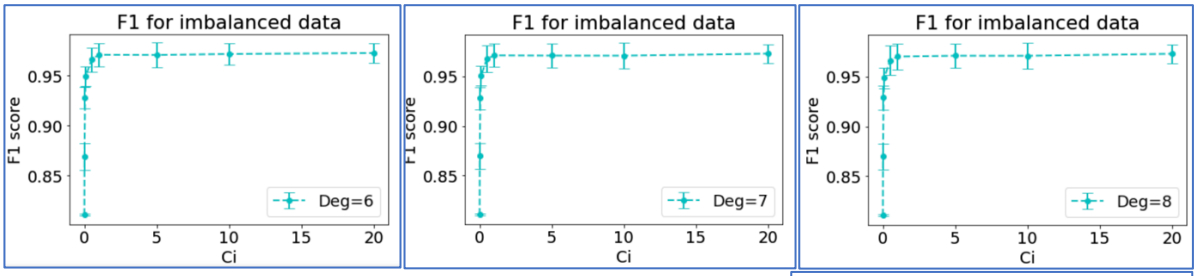
Dataset 2 is also skewed with +1 targets exceeding -1 by over 100%.

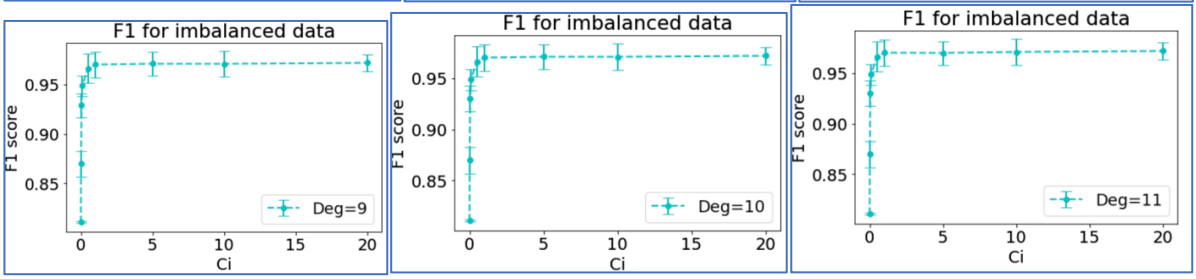


The scatters of the two plots shows us the same.

For Logistic Regression:
Now looping over degs = [3,4,5,6,7,8,9,10,11] for each of C_values = [0.001,0.01,0.05,0.1,0.5,1,5,10,20] we get the following plots

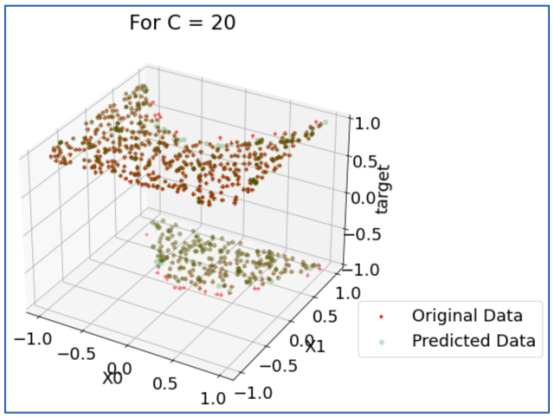




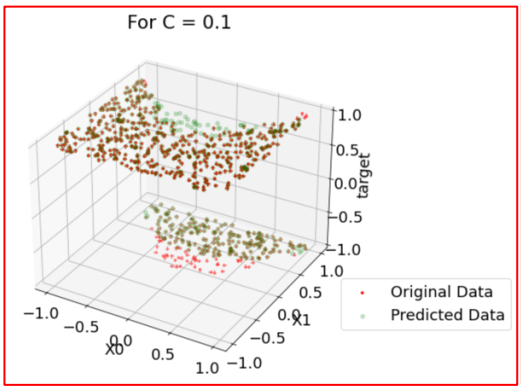


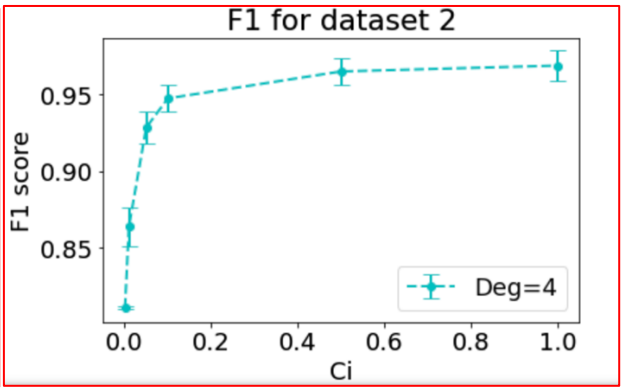
As all plots looks same, and they plateau for higher C value, it would be ideal to keep the model parameters few with degree = 4, and C=20

The Scatter plot for degree 4 and c=2- is given below

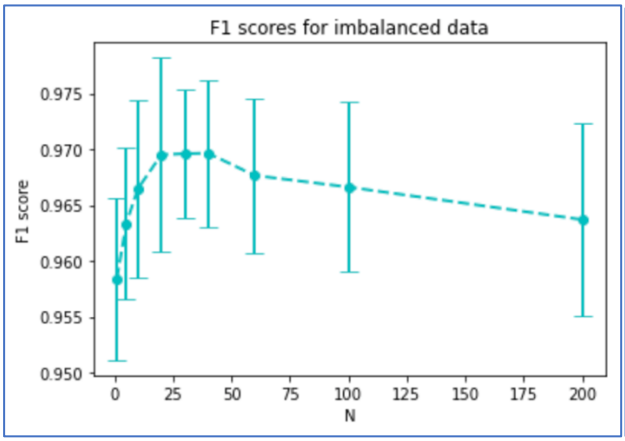


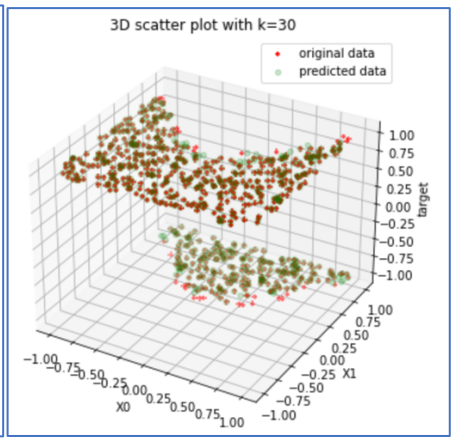
The scatter shows a good overlap of predicted datapoints and original datapoints as compared to dataset 1. However, at a lower value of C, meaning when the regularizer strength is higher, the f1 score is less as compared to C>1. We may be overfitting our model with the training data, to consider a more realistic model, below is a plot of C=0.1. C=0.1 lies at the beginning of the plateau.





Chosen C=0.1 ; degree =4
For KNN Classifier:
looping over k values [1,5,10,20,30,40,60,100,200] (keeping k<=578 samples)

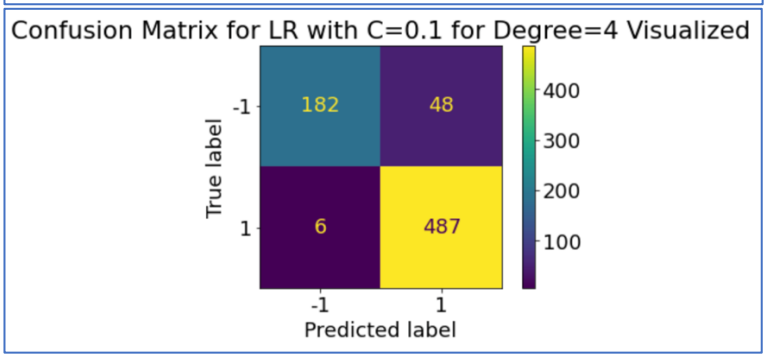




The most ideal f1 score value is at k=30 with the scatter plot of the datapoints of original and predicted values given next to the f1 scores image.

Confusion matrices for LR

```
Confusion Matrix for LR with Penalty L2 and C=0.1 for Degree=4
[[487   6]
 [ 48 182]]


Confusion Matrix for Dummy Classifier with strategy=Stratified for Degree=4
[[ 68 162]
 [141 352]]


Confusion Matrix for Dummy Classifier with strategy=most_frequent for Degree=4
[[  0 230]
 [  0 493]]
```

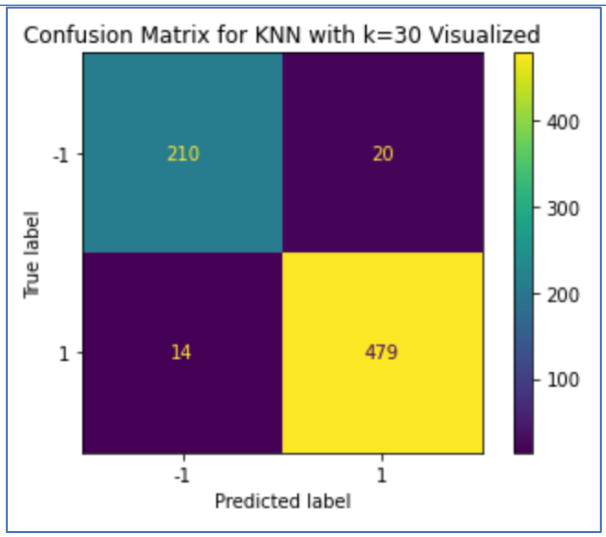


TP=1,1(yellow,487) ; TN=-1,-1(blue,182) ; FP=1,-1(purple,48) ; FN=-1,1(purple,6)

Accuracy = 0.64 ; Precision = 0.91 ; Recall = 0.97 ; F1-Score = 0.93 ; Specificity = 0.79

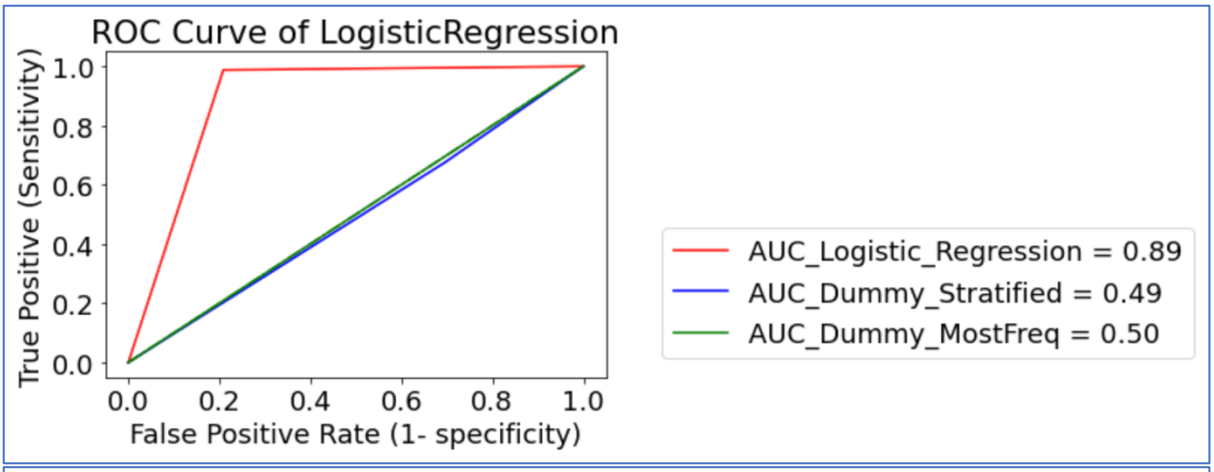
Confusion matrices for KNN Classifier

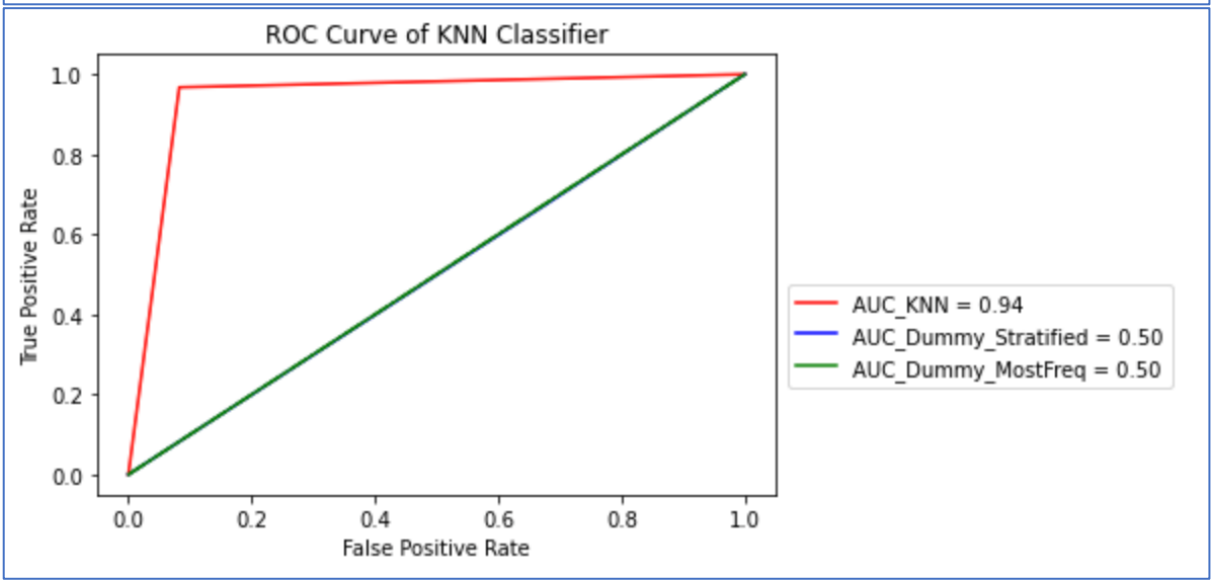
```
Confusion Matrix for KNN with k=30 with No Polynomial Features
[[210  20]
 [ 14 479]]


Confusion Matrix for Dummy Classifier with strategy=most_frequent for Degree=0
[[  0 230]
 [  0 493]]


Confusion Matrix for Dummy Classifier with strategy=Stratified for Degree=0
[[ 73 157]
 [171 322]]
```



TP=1,1(yellow,479) ; TN=-1,-1(blue,210) ; FP=1,-1(purple,20) ; FN=-1,1(purple,14)

Accuracy = 0.65 ; Precision = 0.95 ; Recall = 0.97 ; F1-Score = 0.95 ; Specificity = 0.95





The ROC curve for dataset 2 shows that, the most ideal threshold for LR is somewhere around 0.2. Between KNN and LR for dataset 2, KNN is performing better as the AUC for KNN is greater than AUC for LR. I would go with KNN for this dataset.

Appendix

For LR:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, PolynomialFeatures

df = pd.read_csv('week4d2.csv')

X1=df.iloc[:,0]
X2=df.iloc[:,1]
y=df.iloc[:,2]
X = np.column_stack((X1,X2))
print('+1 targets =',len(X2[y==1]),', -1 targets =',len(X2[y==-1]))
```

```
plt.subplots(figsize=(5, 5))
plt.xlabel('x_1_feature')
plt.ylabel('x_2_feature')
plt.scatter(X1[y==1], X2[y==1], c='blue', marker= '+', s=50, alpha=0.7, label='1')
plt.scatter(X1[y==-1], X2[y==-1],c='red', marker='o',s=10, alpha=0.7, label='-1')
plt.legend()
```

```
fig = plt.figure(figsize = (5, 5))
ax = fig.add_subplot(111,projection='3d')

ax.view_init(25, 30)
ax.scatter3D(X[:,0],X[:,1],y,marker='+',c = y<0,s=20,cmap = 'coolwarm')
ax.set_title('3D scatter plot of Test Data')
ax.set_xlabel('X-axis, Feature 1',fontsize=14)
ax.set_ylabel('Y-axis, Feature 2',fontsize=14)
ax.set_zlabel('Z-axis, Target',fontsize=14)
ax.set_zlim(-1,1)
ax.legend()
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

model_params = []
C_values = [0.001,0.01,0.05,0.1,0.5,1]
degs = [3,4,5,6,7,8,9,10,11]

for d in degs:
    trans5 = PolynomialFeatures(degree=d,include_bias=False)
    X1_poly5 = trans5.fit_transform(X)
    mean_error=[]
    std_error=[]
    i=0
    for c_value in C_values:
        model = LogisticRegression(solver='lbfgs', penalty='l2', C=c_value, max_iter=10000)
        scores = cross_val_score(model, X1_poly5, y, cv=5, scoring='f1')
        mean_error.append(np.array(scores).mean())
        std_error.append(np.array(scores).std())

    import matplotlib.pyplot as plt
    print("For degree :",d)
    plt.errorbar(C_values,mean_error,yerr=std_error,linewidth=2, capsize=6, fmt='co--', label="Deg={}".format(d))
    plt.title('F1 for dataset 2')
    plt.xlabel('Ci');
    plt.ylabel('F1 score')
    plt.legend()
    plt.show()
```

```
poly = PolynomialFeatures(4)
X2_poly = poly.fit_transform(X)


from sklearn.model_selection import cross_val_predict

model = LogisticRegression(solver='lbfgs', penalty='l2', C=0.1, max_iter=50000)
ypred = cross_val_predict(model, X2_poly, y, cv=5)

fig = plt.figure()
fig.set_size_inches(8, 8)
ax.view_init(25, 30)
ax = fig.add_subplot(111, projection='3d', xlabel='X0', ylabel='X1', zlabel='target')
ax.scatter3D(X[:,0],X[:,1],y, color='red', marker='+',label='Original Data')
ax.scatter3D(X[:,0],X[:,1],ypred, color='green',marker='o',label='Predicted Data',alpha=0.2)
ax.set_zlim(-1,1)
ax.legend(loc='best', bbox_to_anchor=(1, 0, 0.5, 0.3))
ax.set_title("For C = 0.1")
```

```
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
from sklearn.model_selection import cross_val_predict

labels=[1,-1]

model = LogisticRegression(solver='lbfgs', penalty='l2', C=0.1, max_iter=50000)
ypred = cross_val_predict(model, X2_poly, y, cv=5)
cm = confusion_matrix(y, ypred,labels=labels)
print("Confusion Matrix for LR with Penalty L2 and C=0.1 for Degree=4")
print(cm)

print('\n')
from sklearn.dummy import DummyClassifier
dummy = DummyClassifier(strategy='stratified').fit(X2_poly, y)
ydummys = dummy.predict(X2_poly)
print("Confusion Matrix for Dummy Classifier with strategy=Stratified for Degree=4")
print(confusion_matrix(y, ydummys))

print('\n')
from sklearn.dummy import DummyClassifier
dummy = DummyClassifier(strategy='most_frequent').fit(X2_poly, y)
ydummym = dummy.predict(X2_poly)
print("Confusion Matrix for Dummy Classifier with strategy=most_frequent for Degree=4")
print(confusion_matrix(y, ydummym))

labels=[-1,1]
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

print('\n')
cm = confusion_matrix(y, ypred, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot()
disp.ax_.set_title("Confusion Matrix for LR with C=0.1 for Degree=4 Visualized")
plt.show()
```

```
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, roc_auc_score, precision_score

model = LogisticRegression(solver='lbfgs', penalty='l2', C=0.1, max_iter=50000)
ypred = cross_val_predict(model, X2_poly, y, cv=5)

fpr_lg, tpr_lg, threshold_lg = roc_curve(y, ypred)
roc_auc_lg = auc(fpr_lg, tpr_lg)

fpr_r, tpr_r, threshold_r = roc_curve(y, ydummys)
roc_auc_r = auc(fpr_r, tpr_r)

fpr_m, tpr_m, threshold_m, = roc_curve(y, ydummym)
roc_auc_m = auc(fpr_m, tpr_m)

plt.plot(fpr_lg,tpr_lg,color='red',label = 'AUC_Logistic_Regression = %0.2f' % roc_auc_lg)
plt.plot(fpr_r,tpr_r,color='blue',label = 'AUC_Dummy_Stratified = %0.2f' % roc_auc_r)
plt.plot(fpr_m,tpr_m,color='green',label = 'AUC_Dummy_MostFreq = %0.2f' % roc_auc_m)

plt.legend(loc='upper right',bbox_to_anchor=(1, 0, 1.2, 0.5))
plt.ylabel('True Positive (Sensitivity)')
plt.xlabel('False Positive Rate (1- specificity)')
plt.title('ROC Curve of LogisticRegression')
plt.plot([0, 1], [0, 1], color='green',linestyle='dashdot')
plt.show()
```

For KNN:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

mean_error=[]
std_error=[]
C_values = [1,5,10,20,30,40,60,100,200]
for n in C_values:
    model = KNeighborsClassifier(n_neighbors=n,weights='distance')
    scores = cross_val_score(model, X, y, cv=5, scoring='f1')
    mean_error.append(np.array(scores).mean())
    std_error.append(np.array(scores).std())

import matplotlib.pyplot as plt
plt.errorbar(C_values,mean_error,yerr=std_error,linewidth=2, capsize=6, fmt='co--')
plt.title('F1 scores for dataset 2')
plt.xlabel('N');
plt.ylabel('F1 score')
plt.show()
```

```
from sklearn.model_selection import cross_val_predict

model = KNeighborsClassifier(n_neighbors=30,weights='uniform')
ypred = cross_val_predict(model, X, y, cv=5)

fig = plt.figure()
fig.set_size_inches(10, 6)
ax = fig.add_subplot(111, projection='3d', xlabel='X0', ylabel='X1', zlabel='target')
ax.set_title('3D scatter plot with k=30')
ax.scatter3D(X[:,0],X[:,1],y, color='red', marker='+',label='original data')
ax.scatter3D(X[:,0],X[:,1],ypred, color='green',marker='o',alpha=0.2, label='predicted data')
ax.legend()
```

```
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

model = KNeighborsClassifier(n_neighbors=60,weights='uniform')
ypred = cross_val_predict(model, X, y, cv=5)

fpr_lg, tpr_lg, threshold_lg = roc_curve(y, ypred)
roc_auc_lg = auc(fpr_lg, tpr_lg)

fpr_r, tpr_r, threshold_r = roc_curve(y, ydummys)
roc_auc_r = auc(fpr_r, tpr_r)

fpr_m, tpr_m, threshold_m, = roc_curve(y, ydummym)
roc_auc_m = auc(fpr_m, tpr_m)

plt.plot(fpr_lg,tpr_lg,color='red',label = 'AUC_KNN = %0.2f' % roc_auc_lg)
plt.plot(fpr_r,tpr_r,color='blue',label = 'AUC_Dummy_Stratified = %0.2f' % roc_auc_r)
plt.plot(fpr_m,tpr_m,color='green',label = 'AUC_Dummy_MostFreq = %0.2f' % roc_auc_m)
plt.legend(loc='upper right',bbox_to_anchor=(1, 0., 0.6, 0.5))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of KNN Classifier')
plt.plot([0, 1], [0, 1], color='green',linestyle='dashdot')
plt.show()

print('\n')
import pandas as pd
i = np.arange(len(tpr_lg))
roc = pd.DataFrame({'tf' : pd.Series(tpr_lg-(1-fpr_lg), index=i), 'thresholds' : pd.Series(threshold_lg, index=i)})
ideal_roc_thresh = roc.iloc[(roc.tf-0).abs().argsort()[:1]]  #Locate the point where the value is close to 0
print("Ideal threshold is: ", ideal_roc_thresh['thresholds'])
```

Thank you.