In [50]:

```
import numpy as np
import math
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import glob
import os
import io
import re
import unicodedata
from datetime import datetime
from itertools import groupby
from operator import itemgetter
from sklearn.utils.class_weight import compute_class_weight
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV, cr
from sklearn.metrics import accuracy_score, recall_score, precision_sc
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from sklearn.pipeline import Pipeline
from imblearn.pipeline import Pipeline as ImbPipeline
from sklearn.decomposition import PCA
from imblearn.over_sampling import SMOTE, BorderlineSMOTE
from xgboost import XGBClassifier
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

pivoted_df, Logistic Regression Model, baseline

In [6]:

```
pivoted_df = pd.read_csv(io.BytesIO(uploaded['~/Documents/Flatiron/Pro
```

In [3]:
```
1 pivoted_df.head()
```

Out[3]:

	season	Age	Throws	Surgery	AB_release_speed_weighted_avg	CH_release_speed_weigh
0	2008	37.0	1	0.0	0.0	8
1	2009	38.0	1	0.0	0.0	8
2	2010	39.0	1	0.0	0.0	8
3	2011	40.0	1	0.0	0.0	8
4	2012	41.0	1	0.0	0.0	8

5 rows × 130 columns

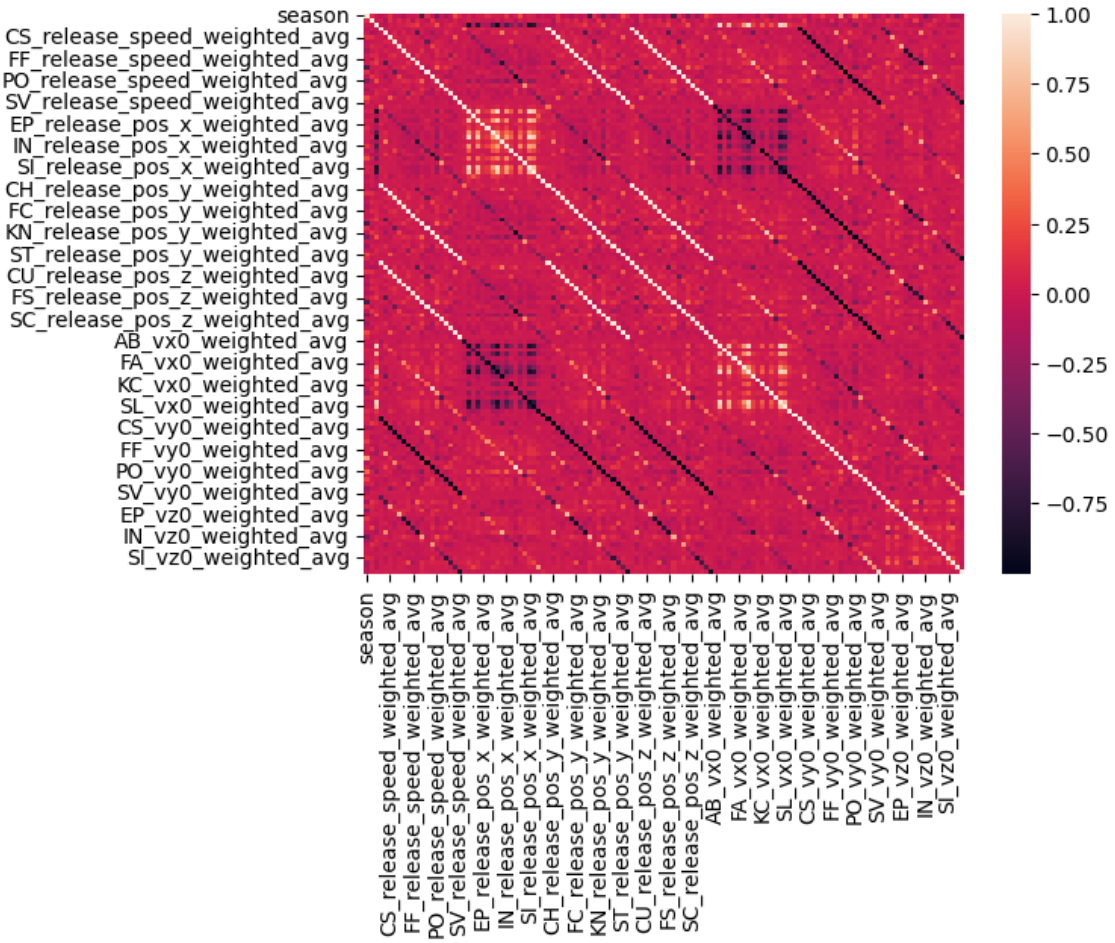
In [4]:
```
1 pivoted_df['Surgery'].value_counts()
```

Out[4]:
```
0.0    2772
1.0     916
Name: Surgery, dtype: int64
```

Can't read this heatmap.

In [110]:
```
1 corr = X_train.corr()
2 sns.heatmap(corr)
```

Out[110]: <Axes: >



In [14]:
```
1 y = pivoted_df['Surgery']
2 X = pivoted_df.drop('Surgery', axis=1)
```

In [16]:
```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [17]:

```
logreg_pipeline = Pipeline([
    ('scale', StandardScaler()),
    ('logreg', LogisticRegression(solver='liblinear'))
])

# Define the parameter grid to search over
param_grid = {
    'logreg__C': [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization str
    'logreg__penalty': ['l1', 'l2']  # Norm used in the penalization
}

# Initialize GridSearchCV with the pipeline, parameter grid, and desir
grid_search = GridSearchCV(logreg_pipeline, param_grid, cv=5, scoring=

# Assuming X_train and y_train are already defined
grid_search.fit(X_train, y_train)

# Best parameters found
print("Best parameters: ", grid_search.best_params_)

# Best cross-validation score
print("Best cross-validation score: {:.2f}".format(grid_search.best_sc

# Test set score using the best parameters
print("Test set score: {:.2f}".format(grid_search.score(X_test, y_test
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: Conver
genceWarning: Liblinear failed to converge, increase the number of iterat
ions.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: Conver
genceWarning: Liblinear failed to converge, increase the number of iterat
ions.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: Conver
genceWarning: Liblinear failed to converge, increase the number of iterat
ions.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: Conver
genceWarning: Liblinear failed to converge, increase the number of iterat
ions.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: Conver
genceWarning: Liblinear failed to converge, increase the number of iterat
ions.
  warnings.warn(

Best parameters:  {'logreg__C': 10, 'logreg__penalty': 'l1'}
Best cross-validation score: 0.76
Test set score: 0.77
```

In [18]:
```
logreg_pipeline = Pipeline([
    ('scale', StandardScaler()),
    ('logreg', LogisticRegression(penalty='l1', C=10.0, solver='liblin
])
```

In [19]:
```
logreg_pipeline.fit(X_train, y_train)
```

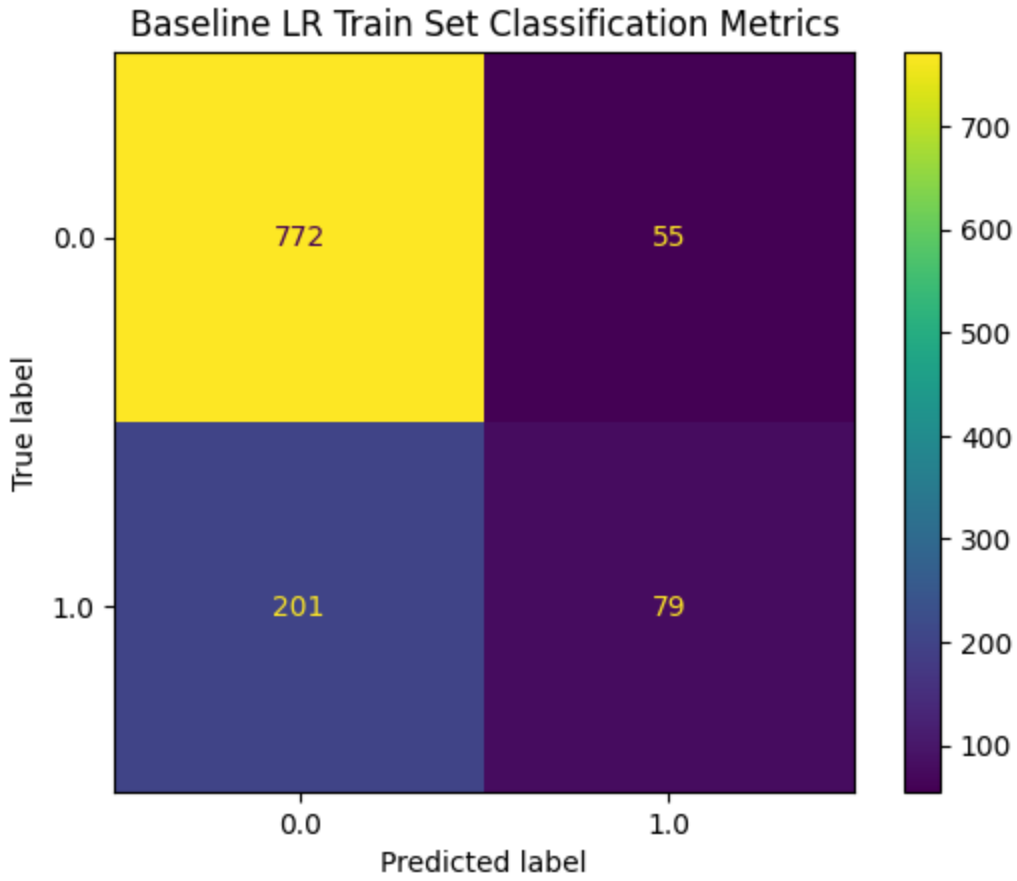
Out[19]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('logreg',
                 LogisticRegression(C=10.0, penalty='l1', solver='libline
ar'))])
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [20]:
```
logreg_pipeline.score(X_test, y_test)
```

Out[20]:
```
0.7687443541102078
```

In [21]:
```
y_pred = logreg_pipeline.predict(X_test)
```

In [22]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('Baseline LR Train Set Classification Metrics')
plt.show()
print(classification_report(y_test, y_pred))
```



	precision	recall	f1-score	support
0.0	0.79	0.93	0.86	827
1.0	0.59	0.28	0.38	280
accuracy			0.77	1107
macro avg	0.69	0.61	0.62	1107
weighted avg	0.74	0.77	0.74	1107

pivoted_df, Logistic Regression Model, SMOTE

In [23]:
```
smote = SMOTE()
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_t

print(pd.Series(y_train_resampled).value_counts())
```

```
1.0    1945
0.0    1945
Name: Surgery, dtype: int64
```

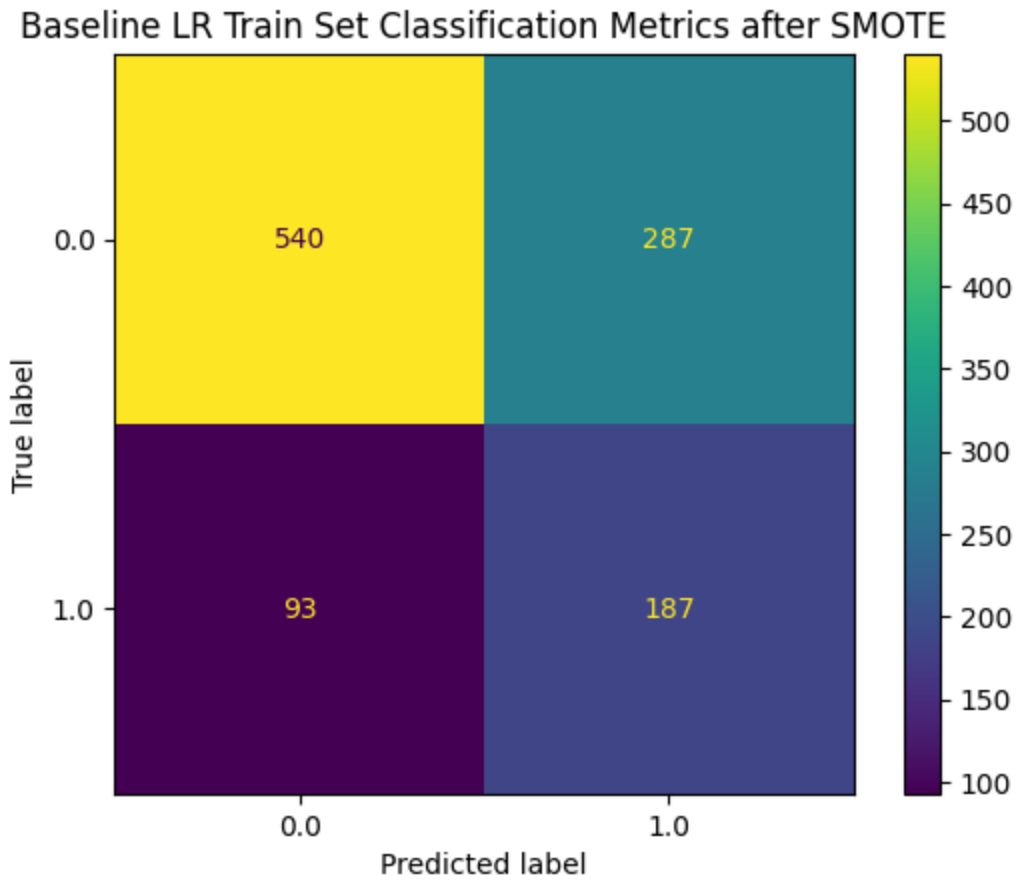
In [24]:
```
1 logreg_pipeline.fit(X_train_resampled, y_train_resampled)
```

Out[24]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('logreg',
                 LogisticRegression(C=10.0, penalty='l1', solver='libline
ar'))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [25]:
```
1 y_pred_resampled = logreg_pipeline.predict(X_test)
```

In [26]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_resampled)
plt.title('Baseline LR Train Set Classification Metrics after SMOTE')
plt.show()
print(classification_report(y_test, y_pred_resampled))
```




```
              precision    recall  f1-score   support

         0.0       0.85      0.65      0.74       827
         1.0       0.39      0.67      0.50       280

    accuracy                           0.66      1107
   macro avg       0.62      0.66      0.62      1107
weighted avg       0.74      0.66      0.68      1107
```

Predicting 1.0 improved, however the rest of the model suffered. Can try condensing the number of features and see how that plays out.

cond_pivoted_df, Logistic Regression, Baseline

In [23]:
```
cond_pivoted_df = pd.read_csv(io.BytesIO(uploaded['~/Documents/Flatiro
```

In [84]:
```
1 cond_pivoted_df.head()
```

Out[84]:

	season	Age	Throws	Surgery	BB_release_speed_weighted_avg	FB_release_speed_weigh
0	2008	37.0	1	0.0	75.425843	9
1	2009	38.0	1	0.0	78.181818	9
2	2010	39.0	1	0.0	74.666667	9
3	2011	40.0	1	0.0	76.885714	9
4	2012	41.0	1	0.0	76.427273	9

5 rows × 39 columns

In [86]:
```
1 cond_pivoted_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3688 entries, 0 to 3687
Data columns (total 39 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   season                          3688 non-null   int64  
 1   Age                             3688 non-null   float64
 2   Throws                          3688 non-null   int64  
 3   Surgery                         3688 non-null   float64
 4   BB_release_speed_weighted_avg   3688 non-null   float64
 5   FB_release_speed_weighted_avg   3688 non-null   float64
 6   OS_release_speed_weighted_avg   3688 non-null   float64
 7   OT_release_speed_weighted_avg   3688 non-null   float64
 8   SB_release_speed_weighted_avg   3688 non-null   float64
 9   BB_release_pos_x_weighted_avg   3688 non-null   float64
 10  FB_release_pos_x_weighted_avg   3688 non-null   float64
 11  OS_release_pos_x_weighted_avg   3688 non-null   float64
 12  OT_release_pos_x_weighted_avg   3688 non-null   float64
 13  SB_release_pos_x_weighted_avg   3688 non-null   float64
 14  BB_release_pos_y_weighted_avg   3688 non-null   float64
 15  FB_release_pos_y_weighted_avg   3688 non-null   float64
 16  OS_release_pos_y_weighted_avg   3688 non-null   float64
 17  OT_release_pos_y_weighted_avg   3688 non-null   float64
 18  SB_release_pos_y_weighted_avg   3688 non-null   float64
 19  BB_release_pos_z_weighted_avg   3688 non-null   float64
 20  FB_release_pos_z_weighted_avg   3688 non-null   float64
 21  OS_release_pos_z_weighted_avg   3688 non-null   float64
 22  OT_release_pos_z_weighted_avg   3688 non-null   float64
 23  SB_release_pos_z_weighted_avg   3688 non-null   float64
 24  BB_vx0_weighted_avg             3688 non-null   float64
 25  FB_vx0_weighted_avg             3688 non-null   float64
 26  OS_vx0_weighted_avg             3688 non-null   float64
 27  OT_vx0_weighted_avg             3688 non-null   float64
 28  SB_vx0_weighted_avg             3688 non-null   float64
 29  BB_vy0_weighted_avg             3688 non-null   float64
 30  FB_vy0_weighted_avg             3688 non-null   float64
 31  OS_vy0_weighted_avg             3688 non-null   float64
 32  OT_vy0_weighted_avg             3688 non-null   float64
 33  SB_vy0_weighted_avg             3688 non-null   float64
 34  BB_vz0_weighted_avg             3688 non-null   float64
 35  FB_vz0_weighted_avg             3688 non-null   float64
 36  OS_vz0_weighted_avg             3688 non-null   float64
 37  OT_vz0_weighted_avg             3688 non-null   float64
 38  SB_vz0_weighted_avg             3688 non-null   float64
dtypes: float64(37), int64(2)
memory usage: 1.1 MB
```

In [31]:
```
1 cond_pivoted_df['Surgery'].value_counts()
```

Out[31]:
```
0.0    2772
1.0     916
Name: Surgery, dtype: int64
```

In [83]:
```
cond_pivoted_df.drop(columns=['pitcher'], inplace=True)
```

In [18]:
```
y = cond_pivoted_df['Surgery']
X = cond_pivoted_df.drop('Surgery', axis=1)
```

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [23]:
```
logreg_pipeline = Pipeline([
    ('scale', StandardScaler()),
    ('logreg', LogisticRegression(solver='liblinear'))
])

# Define the parameter grid to search over
param_grid = {
    'logreg__C': [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization str
    'logreg__penalty': ['l1', 'l2']  # Norm used in the penalization
}

# Initialize GridSearchCV with the pipeline, parameter grid, and desir
grid_search = GridSearchCV(logreg_pipeline, param_grid, cv=5, scoring=

# Assuming X_train and y_train are already defined
grid_search.fit(X_train_resampled, y_train_resampled)

# Best parameters found
print("Best parameters: ", grid_search.best_params_)

# Best cross-validation score
print("Best cross-validation score: {:.2f}".format(grid_search.best_sc

# Test set score using the best parameters
print("Test set score: {:.2f}".format(grid_search.score(X_test, y_test
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: Conver
genceWarning: Liblinear failed to converge, increase the number of iterat
ions.
  warnings.warn(

Best parameters:  {'logreg__C': 10, 'logreg__penalty': 'l1'}
Best cross-validation score: 0.67
Test set score: 0.64
```

In [26]:
```
logreg_pipeline = Pipeline([
    ('scale', StandardScaler()),
    ('logreg', LogisticRegression(penalty='l1', C=10.0, solver='liblin
])
```

In [14]:
```
1 logreg_pipeline.fit(X_train, y_train)
```

Out[14]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('logreg',
                 LogisticRegression(penalty='l1', solver='liblinear'))])
```

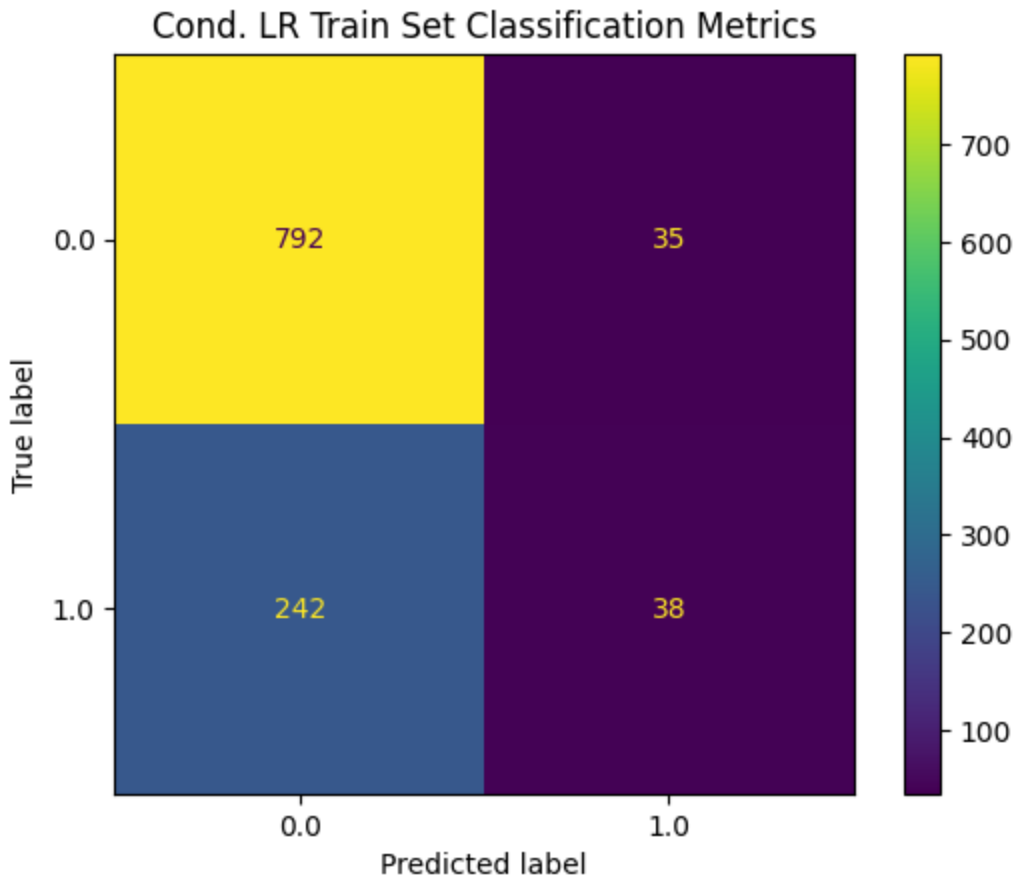
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [15]:
```
1 logreg_pipeline.score(X_test, y_test)
```

Out[15]:
```
0.7497741644083108
```

In [16]:
```
1 y_pred = logreg_pipeline.predict(X_test)
```

In [17]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('Cond. LR Train Set Classification Metrics')
plt.show()
print(classification_report(y_test, y_pred))
```



	precision	recall	f1-score	support
0.0	0.77	0.96	0.85	827
1.0	0.52	0.14	0.22	280
accuracy			0.75	1107
macro avg	0.64	0.55	0.53	1107
weighted avg	0.70	0.75	0.69	1107

Score for 1.0 predicting Surgery too low. Now try with SMOTE.

cond_pivoted_df, Logistic Regression Model, SMOTE

In [25]:
```
smote = SMOTE()
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_t

print(pd.Series(y_train_resampled).value_counts())
```

```
1.0    1945
0.0    1945
Name: Surgery, dtype: int64
```

In [27]:
```
1 logreg_pipeline.fit(X_train_resampled, y_train_resampled)
```

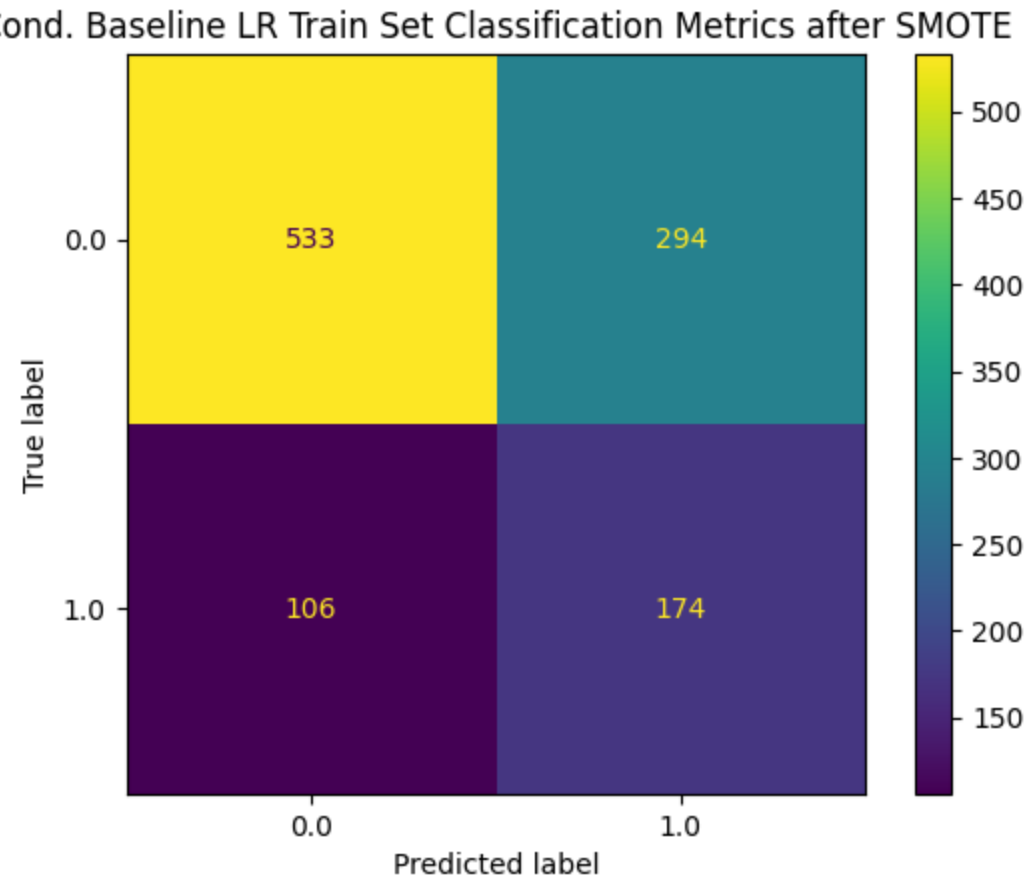
Out[27]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('logreg',
                 LogisticRegression(C=10.0, penalty='l1', solver='libline
ar'))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [28]:
```
1 y_pred_resampled = logreg_pipeline.predict(X_test)
```

In [29]:

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_resampled)
plt.title('Cond. Baseline LR Train Set Classification Metrics after SM
plt.show()
print(classification_report(y_test, y_pred_resampled))
```



	precision	recall	f1-score	support
0.0	0.83	0.64	0.73	827
1.0	0.37	0.62	0.47	280
accuracy			0.64	1107
macro avg	0.60	0.63	0.60	1107
weighted avg	0.72	0.64	0.66	1107

pivot_df is better at predicting TP & TN. cond_pivot_df is better at FP & FN. Trade-off between two models is slim. SMOTE helped, can I address class imbalance further, or will that lead to issues? Still need to see improvement. Need to try PCA, Random Forest, XG Boost

Try to see how GridSearch works when I add in class balance. Will try this out for both pivoted_df and cond_pivoted_df

pivoted_df, Logistic Regression Model, adjust class weights

In [14]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [13]:
```
# Define the parameter grid to search over, including class weights
class_weights = [None, 'balanced', {0: 1, 1: 2}, {0: 1, 1: 3}, {0: 1,
param_grid = {
    'logreg__C': [0.01, 0.1, 1, 10],
    'logreg__penalty': ['l1', 'l2'],
    'logreg__class_weight': class_weights,
    'logreg__max_iter': [5000],
    'logreg__tol': [0.01]
}

# Create a scoring function that focuses on recall for the positive cl
recall_scorer = make_scorer(recall_score, pos_label=1)

# Initialize GridSearchCV with the pipeline, parameter grid, and recal
grid_search = GridSearchCV(weight_logreg_pipeline, param_grid, cv=5, s

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Print the best parameters found and the best recall score
print("Best parameters: ", grid_search.best_params_)
print("Best cross-validation recall score: {:.2f}".format(grid_search.

# Evaluate the best model on the test set
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
print("Test set recall score: {:.2f}".format(recall_score(y_test, y_pr
```

```
Best parameters:  {'logreg__C': 0.01, 'logreg__class_weight': {0: 1, 1:
5}, 'logreg__max_iter': 5000, 'logreg__penalty': 'l1', 'logreg__tol': 0.0
1}
Best cross-validation recall score: 0.87
Test set recall score: 0.85
```

In [16]:
```
weight_logreg_pipeline = Pipeline([
    ('scale', StandardScaler()),
    ('logreg', LogisticRegression(penalty='l1', C=0.01, class_weight={
])
```

In [17]:
```
1 weight_logreg_pipeline.fit(X_train, y_train)
```

Out[17]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('logreg',
                 LogisticRegression(C=0.01, class_weight={0: 1, 1: 5},
                                    penalty='l1', solver='liblinear'))])
```

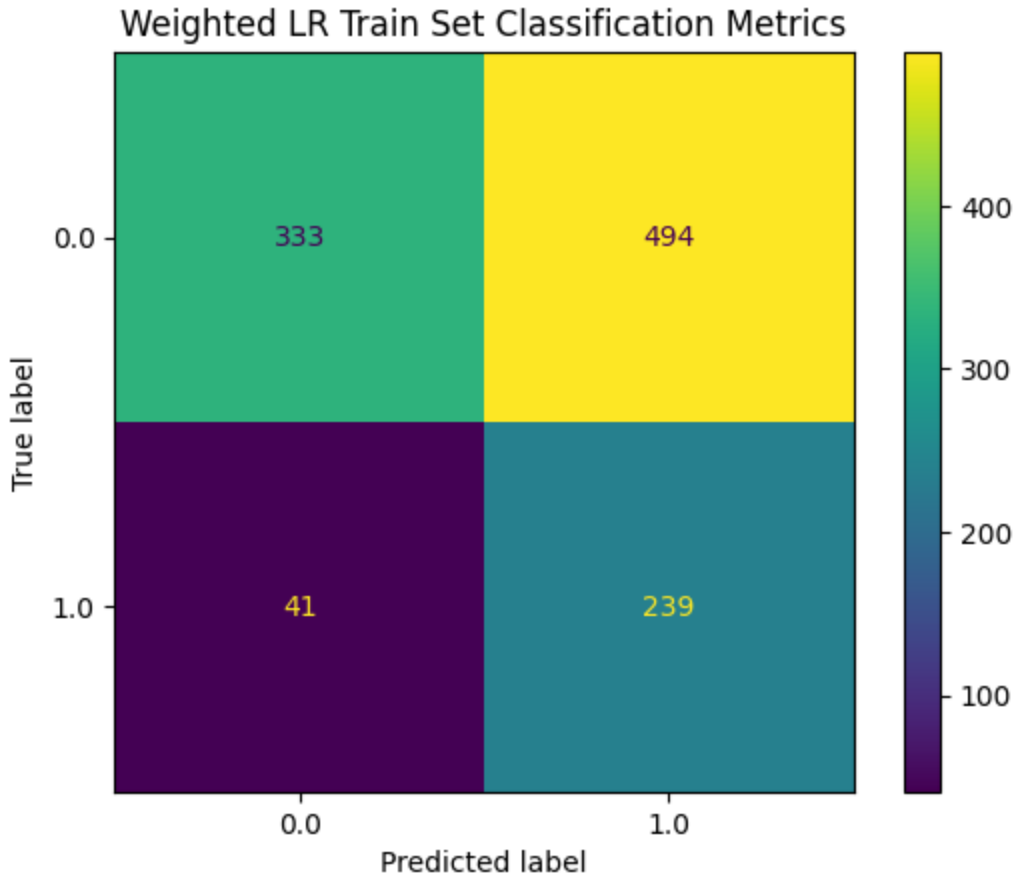
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [18]:
```
1 weight_logreg_pipeline.score(X_test, y_test)
```

Out[18]:
```
0.5167118337850045
```

In [20]:
```
1 y_pred = weight_logreg_pipeline.predict(X_test)
```

In [21]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('Weighted LR Train Set Classification Metrics')
plt.show()
print(classification_report(y_test, y_pred))
```




```
              precision    recall  f1-score   support

         0.0       0.89      0.40      0.55       827
         1.0       0.33      0.85      0.47       280

    accuracy                           0.52      1107
   macro avg       0.61      0.63      0.51      1107
weighted avg       0.75      0.52      0.53      1107
```

This looks better. The number of False negatives has decreased dramatically. Would like to see further improvement to be more accurate all around. Now to test cond_pivoted_df

In [74]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [75]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [10]:
```
logreg_pipeline = Pipeline([
    ('scale', StandardScaler()),
    ('pca', PCA()),
    ('logreg', LogisticRegression(solver='liblinear'))
])

class_weights = [None, 'balanced', {0: 1, 1: 2}, {0: 1, 1: 3}, {0: 1,
param_grid = {
    'pca__n_components': [5, 10, 20, 30],
    'logreg__C': [0.01, 0.1, 1, 10],
    'logreg__penalty': ['l1', 'l2'],
    'logreg__class_weight': class_weights,
    'logreg__max_iter': [5000],
    'logreg__tol': [0.01]
}

recall_scorer = make_scorer(recall_score, pos_label=1)
grid_search = GridSearchCV(logreg_pipeline, param_grid, cv=5, scoring=
grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best cross-validation recall score: {:.2f}".format(grid_search.

best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
print("Test set recall score: {:.2f}".format(recall_score(y_test, y_pr
```

```
Best parameters:  {'logreg__C': 0.1, 'logreg__class_weight': {0: 1, 1:
5}, 'logreg__max_iter': 5000, 'logreg__penalty': 'l1', 'logreg__tol': 0.0
1, 'pca__n_components': 5}
Best cross-validation recall score: 0.92
Test set recall score: 0.90
```

In [76]:
```
logreg_pipeline = Pipeline([
    ('scale', StandardScaler()),
    ('pca', PCA(n_components=5)),
    ('logreg', LogisticRegression(penalty='l1', C=0.1, class_weight={0
])
```

In [77]:
```
logreg_pipeline.fit(X_train, y_train)
```

Out[77]:
```
Pipeline(steps=[('scale', StandardScaler()), ('pca', PCA(n_components=
5)),
                ('logreg',
                 LogisticRegression(C=0.1, class_weight={0: 1, 1: 5},
                                    max_iter=5000, penalty='l1',
                                    solver='liblinear', tol=0.01))])
```

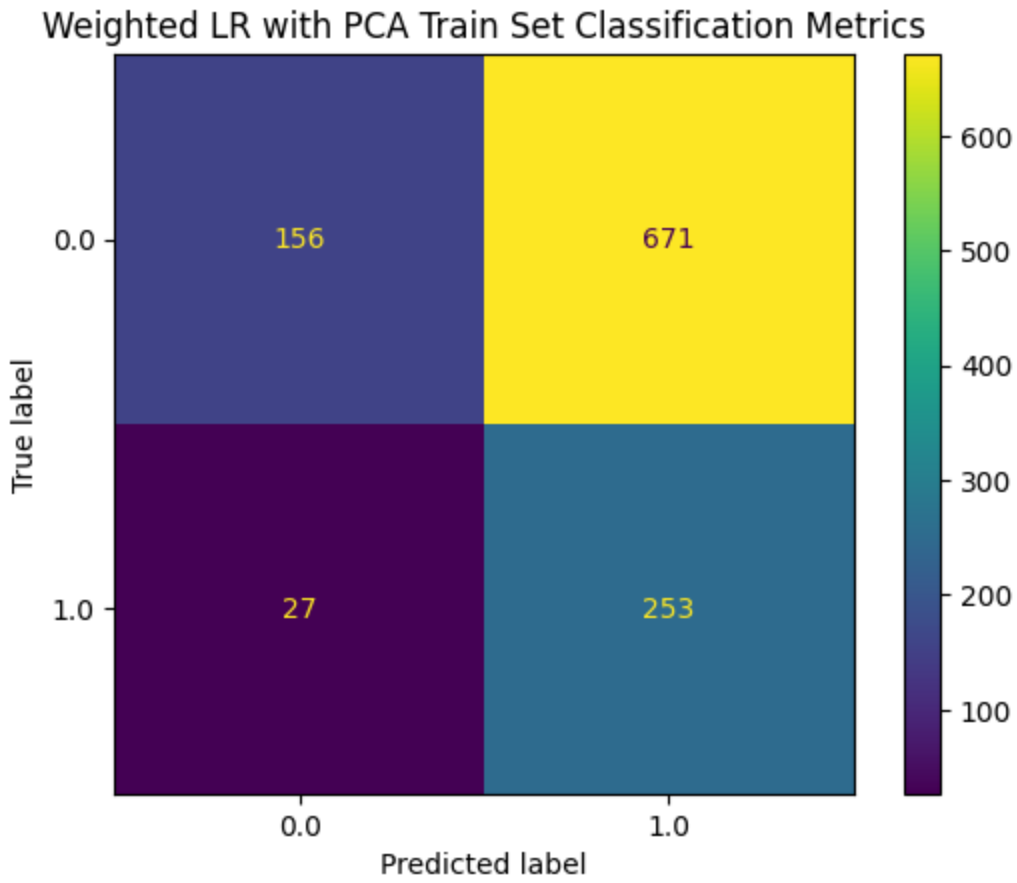
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [78]:
```
1 logreg_pipeline.score(X_test, y_test)
```

Out[78]: 0.36946702800361336

In [79]:
```
1 y_pred = logreg_pipeline.predict(X_test)
```

In [80]:
```
1 ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
2 plt.title('Weighted LR with PCA Train Set Classification Metrics')
3 plt.show()
4 print(classification_report(y_test, y_pred))
```



	precision	recall	f1-score	support
0.0	0.85	0.19	0.31	827
1.0	0.27	0.90	0.42	280
accuracy			0.37	1107
macro avg	0.56	0.55	0.36	1107
weighted avg	0.71	0.37	0.34	1107

Too strong on recall info. Trues weakened significantly compared to the previous model. Will try a weak version of SMOTE.

In [123]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [124]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [131]:
```
logreg_pipeline = ImbPipeline([
    ('scale', StandardScaler()),
    ('smote', SMOTE()),
    ('pca', PCA()),
    ('logreg', LogisticRegression(solver='liblinear'))
])

param_grid = {
    'pca__n_components': [5, 10, 15, 20],
    'smote__sampling_strategy': [0.55, 0.6, 0.65],
    'logreg__C': [0.01, 0.05, 0.1, 1, 10],
    'logreg__penalty': ['l1', 'l2'],
    'logreg__class_weight': [None, 'balanced', {0: 1, 1: 2}],
    'logreg__max_iter': [5000],
    'logreg__tol': [0.01]
}

recall_scorer = make_scorer(recall_score, pos_label=1)
grid_search = GridSearchCV(logreg_pipeline, param_grid, cv=5, scoring=
grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best cross-validation recall score: {:.2f}".format(grid_search.

best_logreg = grid_search.best_estimator_
y_pred = best_logreg.predict(X_test)
print("Test set recall score: {:.2f}".format(recall_score(y_test, y_pr
```

```
Best parameters:  {'logreg__C': 0.1, 'logreg__class_weight': {0: 1, 1:
2}, 'logreg__max_iter': 5000, 'logreg__penalty': 'l2', 'logreg__tol': 0.0
1, 'pca__n_components': 5, 'smote__sampling_strategy': 0.65}
Best cross-validation recall score: 0.82
Test set recall score: 0.78
```

In [132]:
```
best_logreg.fit(X_train, y_train)
```

Out[132]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('smote', SMOTE(sampling_strategy=0.65)),
                ('pca', PCA(n_components=5)),
                ('logreg',
                 LogisticRegression(C=0.1, class_weight={0: 1, 1: 2},
                                    max_iter=5000, solver='liblinear',
                                    tol=0.01))])
```

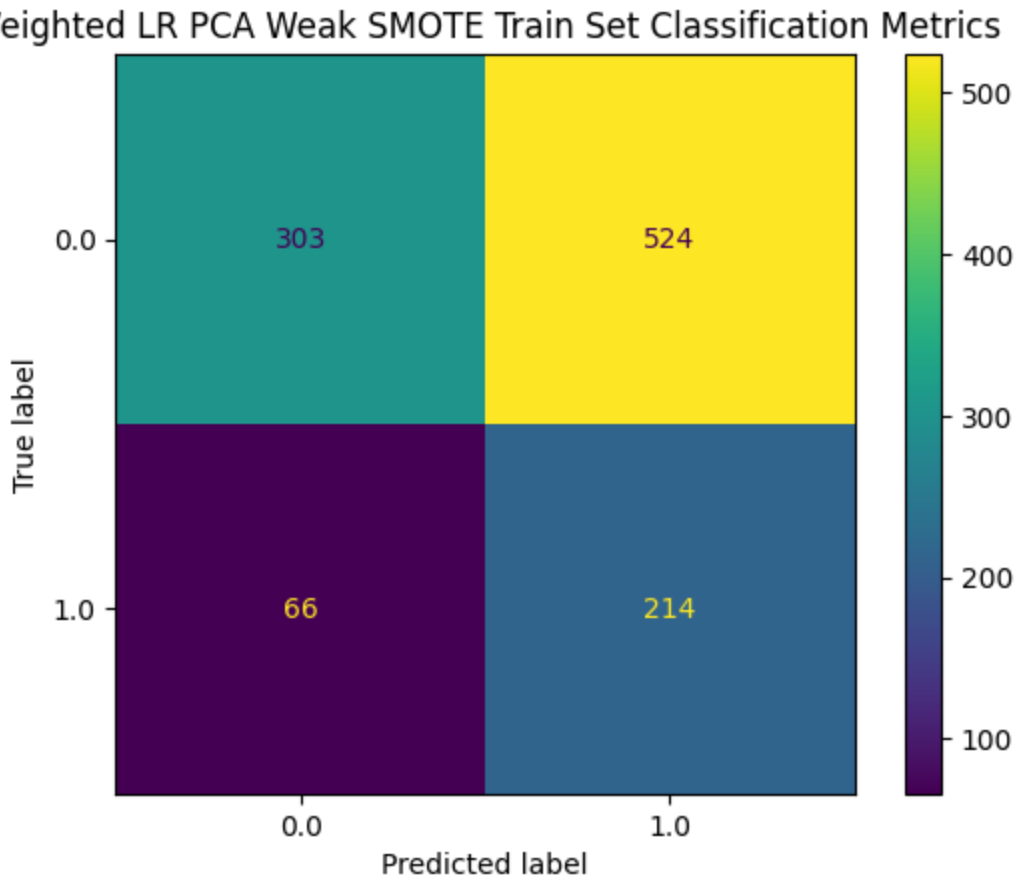
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [133]:
```
1 best_logreg.score(X_test, y_test)
```

Out[133]: 0.4670280036133695

In [134]:
```
1 y_pred = best_logreg.predict(X_test)
```

In [135]:
```
1 ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
2 plt.title('Weighted LR PCA Weak SMOTE Train Set Classification Metrics
3 plt.show()
4 print(classification_report(y_test, y_pred))
```



	precision	recall	f1-score	support
0.0	0.82	0.37	0.51	827
1.0	0.29	0.76	0.42	280
accuracy			0.47	1107
macro avg	0.56	0.57	0.46	1107
weighted avg	0.69	0.47	0.48	1107

Can try BorderlineSMOTE

In [136]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [137]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [138]:
```
logreg_pipeline = ImbPipeline([
    ('scale', StandardScaler()),
    ('blsmote', BorderlineSMOTE()),
    ('pca', PCA()),
    ('logreg', LogisticRegression(solver='liblinear'))
])

param_grid = {
    'pca__n_components': [5, 10, 15, 20],
    'blsmote__sampling_strategy': [0.55, 0.6, 0.65],
    'logreg__C': [0.01, 0.05, 0.1, 1, 10],
    'logreg__penalty': ['l1', 'l2'],
    'logreg__class_weight': [None, 'balanced', {0: 1, 1: 2}],
    'logreg__max_iter': [5000],
    'logreg__tol': [0.01]
}

recall_scorer = make_scorer(recall_score, pos_label=1)
grid_search = GridSearchCV(logreg_pipeline, param_grid, cv=5, scoring=
grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best cross-validation recall score: {:.2f}".format(grid_search.

best_logreg = grid_search.best_estimator_
y_pred = best_logreg.predict(X_test)
print("Test set recall score: {:.2f}".format(recall_score(y_test, y_pr
```

```
Best parameters:  {'blsmote__sampling_strategy': 0.65, 'logreg__C': 1, 'l
ogreg__class_weight': {0: 1, 1: 2}, 'logreg__max_iter': 5000, 'logreg__pe
nalty': 'l2', 'logreg__tol': 0.01, 'pca__n_components': 5}
Best cross-validation recall score: 0.82
Test set recall score: 0.77
```

In [139]:
```
best_logreg.fit(X_train, y_train)
```

Out[139]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('blsmote', BorderlineSMOTE(sampling_strategy=0.65)),
                ('pca', PCA(n_components=5)),
                ('logreg',
                 LogisticRegression(C=1, class_weight={0: 1, 1: 2},
                                    max_iter=5000, solver='liblinear',
                                    tol=0.01))])
```

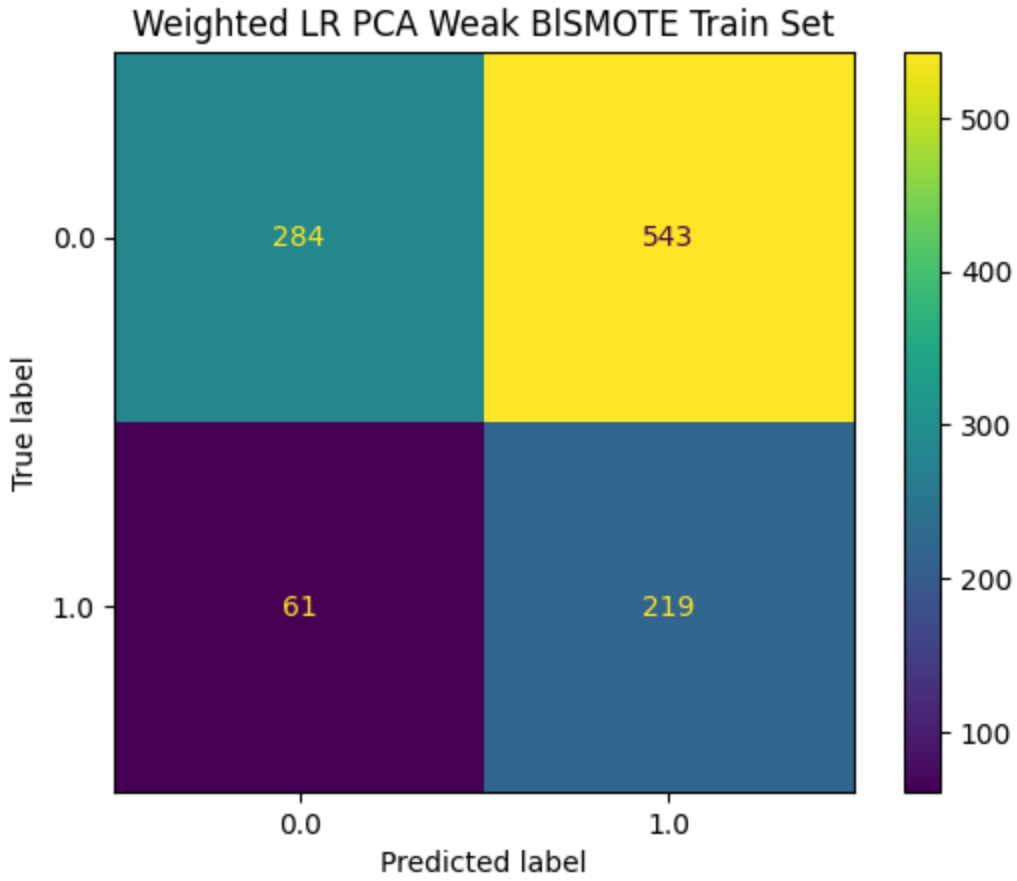
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [140]:
```
1 best_logreg.score(X_test, y_test)
```

Out[140]: 0.45438121047877145

In [141]:
```
1 y_pred = best_logreg.predict(X_test)
```

In [142]:
```
1 ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
2 plt.title('Weighted LR PCA Weak BlSMOTE Train Set')
3 plt.show()
4 print(classification_report(y_test, y_pred))
```



	precision	recall	f1-score	support
0.0	0.82	0.34	0.48	827
1.0	0.29	0.78	0.42	280
accuracy			0.45	1107
macro avg	0.56	0.56	0.45	1107
weighted avg	0.69	0.45	0.47	1107

cond_pivoted_df, Logistic Regression Model, adjust class weights.

In [41]:
```
y = cond_pivoted_df['Surgery']
X = cond_pivoted_df.drop('Surgery', axis=1)
```

In [42]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [32]:
```
class_weights = [None, 'balanced', {0: 1, 1: 2}, {0: 1, 1: 3}, {0: 1,
param_grid = {
    'logreg__C': [0.01, 0.1, 1, 10, 100],
    'logreg__penalty': ['l1', 'l2'],
    'logreg__class_weight': class_weights,
    'logreg__max_iter': [5000],
    'logreg__tol': [0.01]
}

recall_scorer = make_scorer(recall_score, pos_label=1)
grid_search = GridSearchCV(weight_logreg_pipeline, param_grid, cv=5, s
grid_search.fit(X_train_resampled, y_train_resampled)

print("Best parameters: ", grid_search.best_params_)
print("Best cross-validation recall score: {:.2f}".format(grid_search.
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
print("Test set recall score: {:.2f}".format(recall_score(y_test, y_pr
```

```
Best parameters:  {'logreg__C': 0.01, 'logreg__class_weight': {0: 1, 1:
4}, 'logreg__max_iter': 5000, 'logreg__penalty': 'l1', 'logreg__tol': 0.0
1}
Best cross-validation recall score: 1.00
Test set recall score: 0.99
```

In [43]:
```
smote = SMOTE()
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_t

print(pd.Series(y_train_resampled).value_counts())
```

```
1.0    1945
0.0    1945
Name: Surgery, dtype: int64
```

In [44]:
```
weight_logreg_pipeline = Pipeline([
    ('scale', StandardScaler()),
    ('logreg', LogisticRegression(penalty='l1', C=0.01, class_weight={
])
```

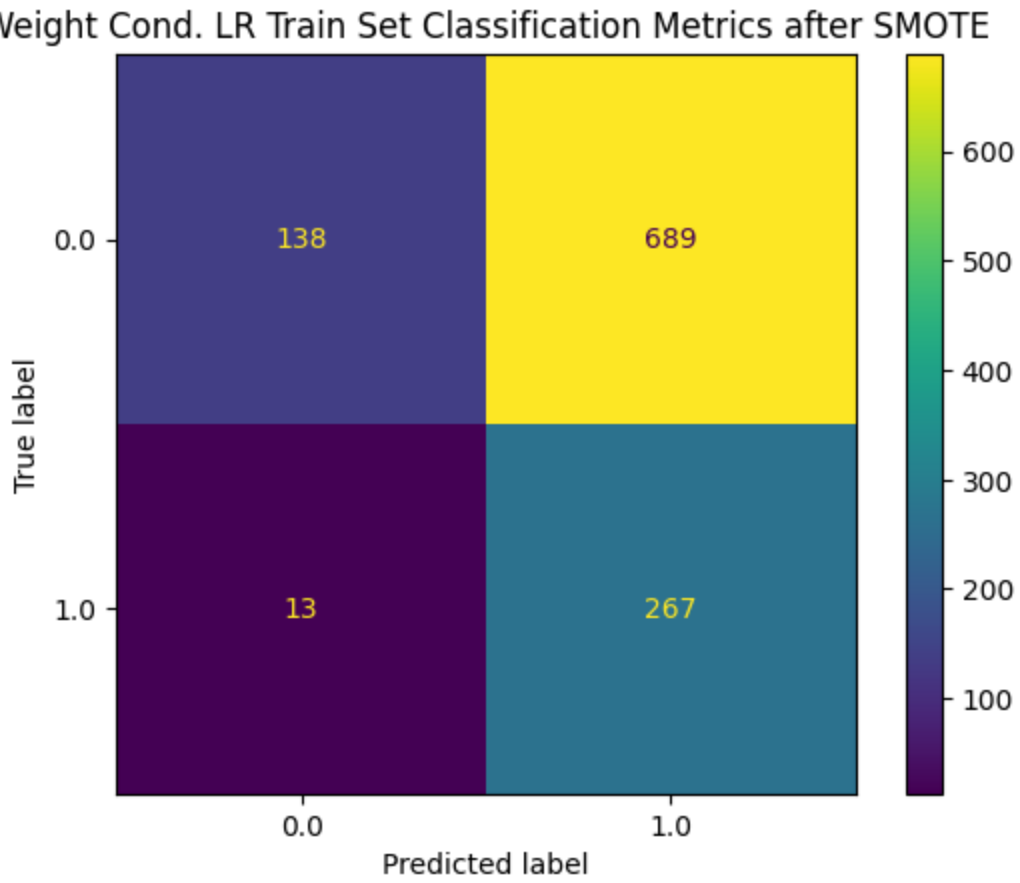
In [45]:
```
1 weight_logreg_pipeline.fit(X_train_resampled, y_train_resampled)
```

Out[45]:
```
Pipeline(steps=[('scale', StandardScaler()),
                ('logreg',
                 LogisticRegression(C=0.01, class_weight={0: 1, 1: 3},
                                    penalty='l1', solver='liblinear'))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [46]:
```
1 y_pred_resampled = weight_logreg_pipeline.predict(X_test)
```

In [47]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_resampled)
plt.title('Weight Cond. LR Train Set Classification Metrics after SMOT
plt.show()
print(classification_report(y_test, y_pred_resampled))
```



	precision	recall	f1-score	support
0.0	0.91	0.17	0.28	827
1.0	0.28	0.95	0.43	280
accuracy			0.37	1107
macro avg	0.60	0.56	0.36	1107
weighted avg	0.75	0.37	0.32	1107

Can revisit the Logistic Regression model later. Time to try Random Forest, XG Boost, maybe CatBoost (not sure if it applies..)

pivoted_df, Decision Tree (Baseline Model)

In [50]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [52]:
```
tree_clf = DecisionTreeClassifier(criterion='gini', max_depth=5)
tree_clf.fit(X_train, y_train)
```

Out[52]:
```
DecisionTreeClassifier(max_depth=5)
```

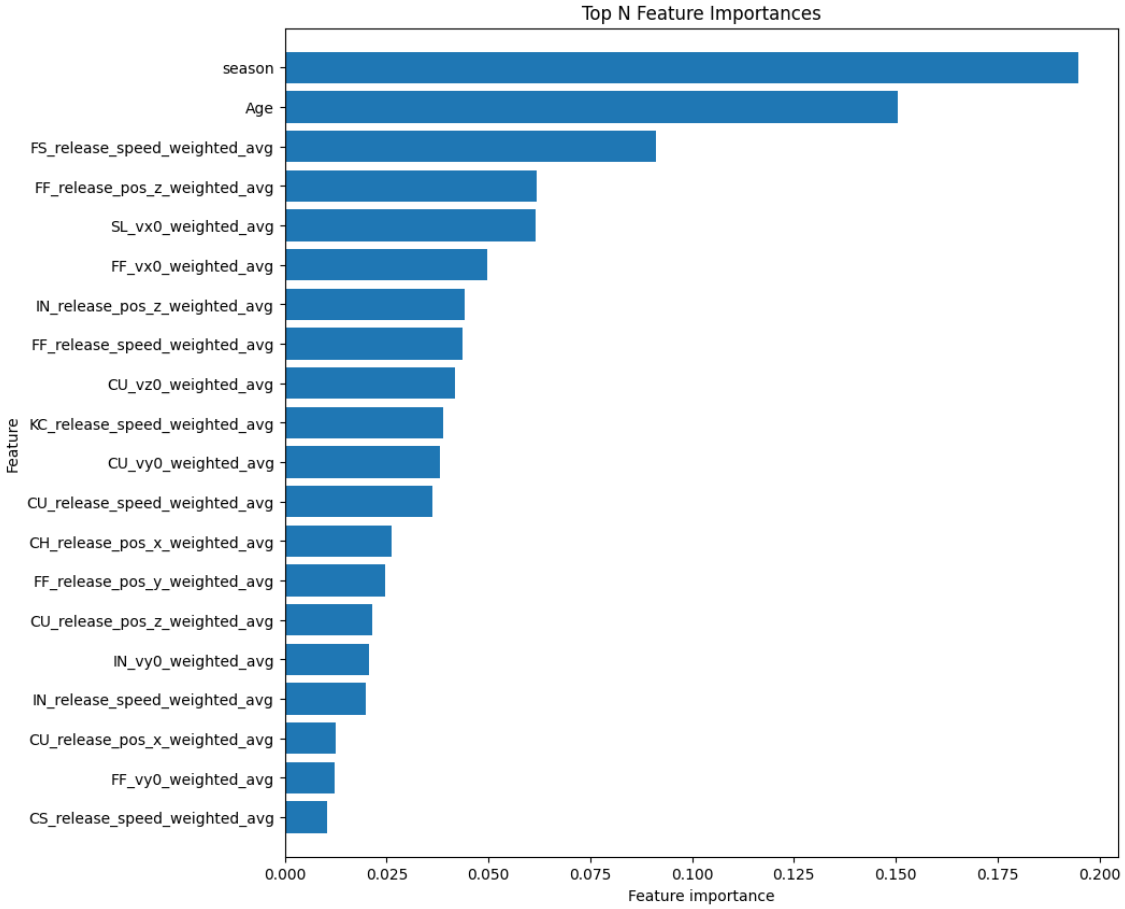
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [53]:
```
tree_clf.feature_importances_
```

Out[53]:
```
array([0.19477055, 0.15049678, 0.        , 0.        , 0.        ,
       0.01033349, 0.03611204, 0.        , 0.        , 0.        ,
       0.04363543, 0.09102768, 0.01991462, 0.03879671, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.02612076, 0.        , 0.01246673,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.02465828, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.02142519, 0.        , 0.        , 0.        , 0.06182032,
       0.        , 0.04426144, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.04978206, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.06168103, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.0381441 , 0.        , 0.        , 0.        ,
       0.0121275 , 0.        , 0.02063147, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.04179382,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        ])
```

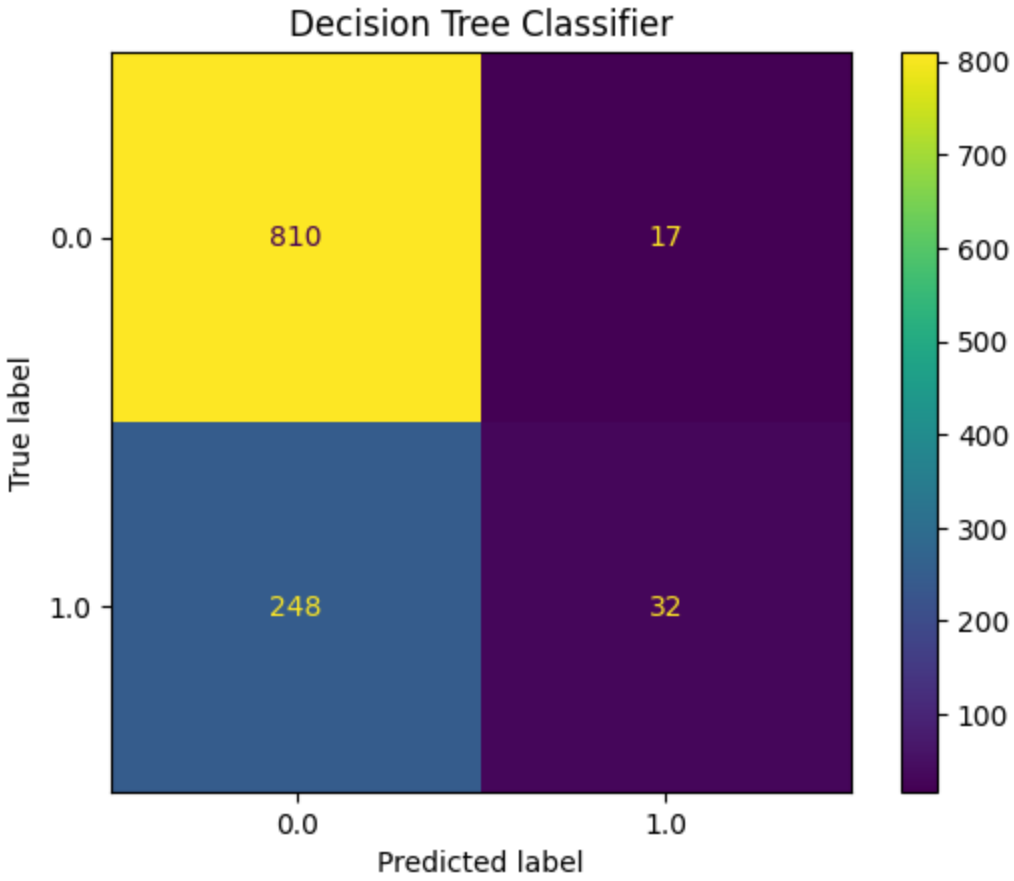
In [57]:

```
def plot_feature_importances(model, n_top_features=20):
    importances = model.feature_importances_
    indices = np.argsort(importances)[-n_top_features:]
    plt.figure(figsize=(10,10))
    plt.title('Top N Feature Importances')
    plt.barh(range(n_top_features), importances[indices], align='cente
    plt.yticks(range(n_top_features), [X_train.columns[i] for i in ind
    plt.xlabel('Feature importance')
    plt.ylabel('Feature')
    plt.ylim(-1, n_top_features)

plot_feature_importances(tree_clf, n_top_features=20)
plt.show()
```



In [59]:

```
pred = tree_clf.predict(X_test)

ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('pivoted_df Decision Tree Classifier')
plt.show()
print(classification_report(y_test, pred))
```




```
              precision    recall  f1-score   support

         0.0       0.77      0.98      0.86       827
         1.0       0.65      0.11      0.19       280

    accuracy                           0.76      1107
   macro avg       0.71      0.55      0.53      1107
weighted avg       0.74      0.76      0.69      1107
```

In [66]:
```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': [None, 'balanced', {0:1, 1:2}, {0:1, 1:3}]
}

tree_clf = DecisionTreeClassifier()
scorer = make_scorer(recall_score)
grid_search = GridSearchCV(estimator=tree_clf, param_grid=param_grid,

grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

best_tree = grid_search.best_estimator_
y_pred = best_tree.predict(X_test)
print("Test set accuracy:", accuracy_score(y_test, y_pred))
```

```
Best parameters: {'class_weight': 'balanced', 'criterion': 'gini', 'max_d
epth': 5, 'min_samples_leaf': 4, 'min_samples_split': 10}
Best score: 0.6540600393700788
Test set accuracy: 0.5817524841915086
```

In [61]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

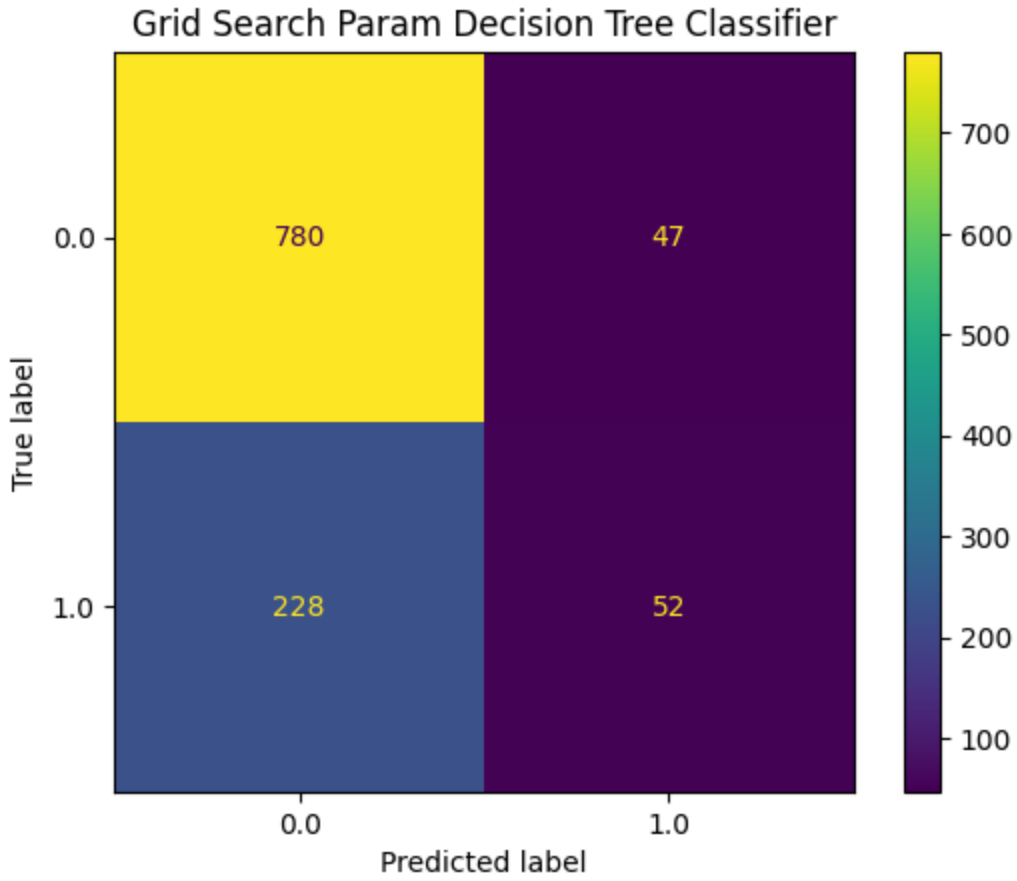
In [62]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [63]:
```
tree_clf = DecisionTreeClassifier(criterion='entropy', max_depth=5, mi
tree_clf.fit(X_train, y_train)
```

Out[63]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_spli
t=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [64]:
```
pred = tree_clf.predict(X_test)

ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('Grid Search Param Decision Tree Classifier')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.77	0.94	0.85	827
1.0	0.53	0.19	0.27	280
accuracy			0.75	1107
macro avg	0.65	0.56	0.56	1107
weighted avg	0.71	0.75	0.70	1107

Grid Search improved on TN & FN but worse on TP & TN. Would rather improve on TN than FN. Can try rerunning.

In [176]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [177]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [182]:
```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': ['balanced', {0:1, 1:2}, {0:1, 1:3}]
}

tree_clf = DecisionTreeClassifier()
scorer = make_scorer(recall_score)
grid_search = GridSearchCV(estimator=tree_clf, param_grid=param_grid,
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

best_tree = grid_search.best_estimator_
y_pred = best_tree.predict(X_test)
print("Test recall score:", recall_score(y_test, y_pred))
```

```
Best parameters: {'class_weight': 'balanced', 'criterion': 'gini', 'max_d
epth': 5, 'min_samples_leaf': 4, 'min_samples_split': 2}
Best score: 0.6540600393700788
Test recall score: 0.7607142857142857
```

In [183]:
```
best_tree.fit(X_train, y_train)
```

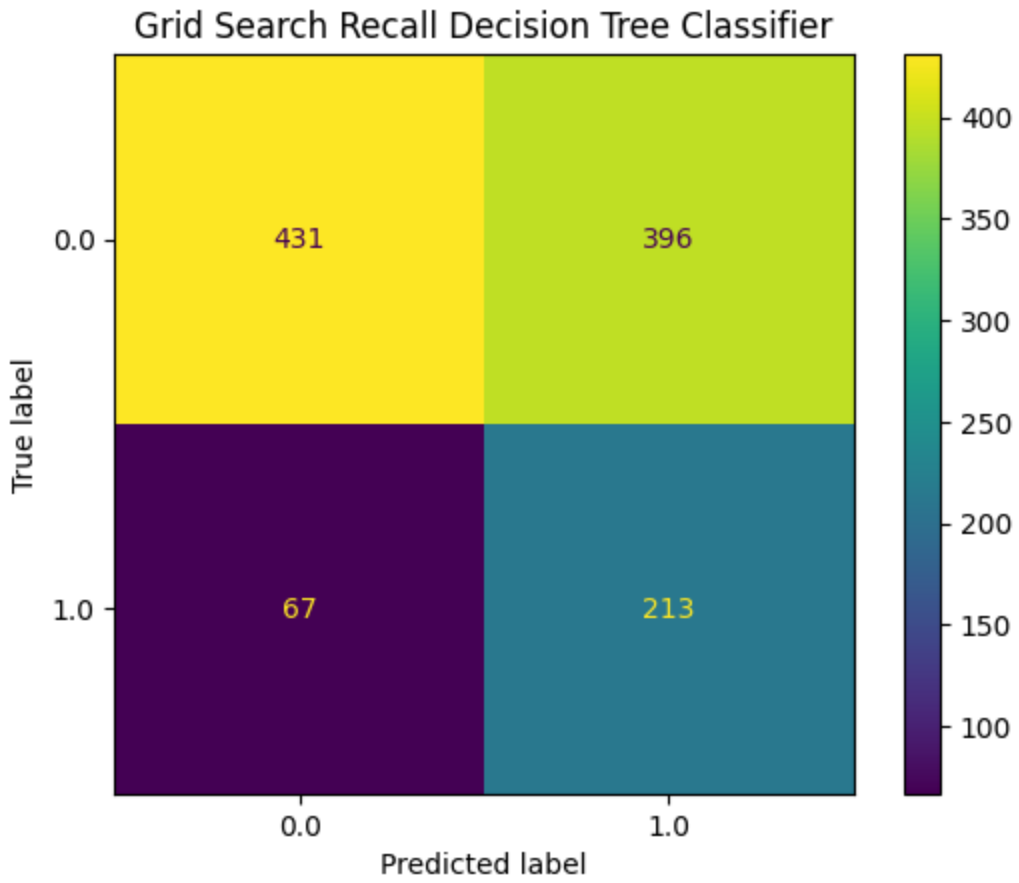
Out[183]:
```
DecisionTreeClassifier(class_weight='balanced', max_depth=5, min_samples_
leaf=4)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [184]:
```
pred = best_tree.predict(X_test)
```

In [186]:

```
ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('Grid Search Recall Decision Tree Classifier')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.87	0.52	0.65	827
1.0	0.35	0.76	0.48	280
accuracy			0.58	1107
macro avg	0.61	0.64	0.56	1107
weighted avg	0.74	0.58	0.61	1107

In [ ]:

```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': ['balanced', {0:1, 1:2}, {0:1, 1:3}]
}

tree_clf = DecisionTreeClassifier()
scorer = make_scorer(recall_score)
grid_search = GridSearchCV(estimator=tree_clf, param_grid=param_grid,
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

best_tree = grid_search.best_estimator_
y_pred = best_tree.predict(X_test)
print("Test recall score:", recall_score(y_test, y_pred))
```

In [187]:

```
tree_clf = DecisionTreeClassifier(criterion='gini', class_weight='bala
tree_clf.fit(X_train, y_train)
```

Out[187]:

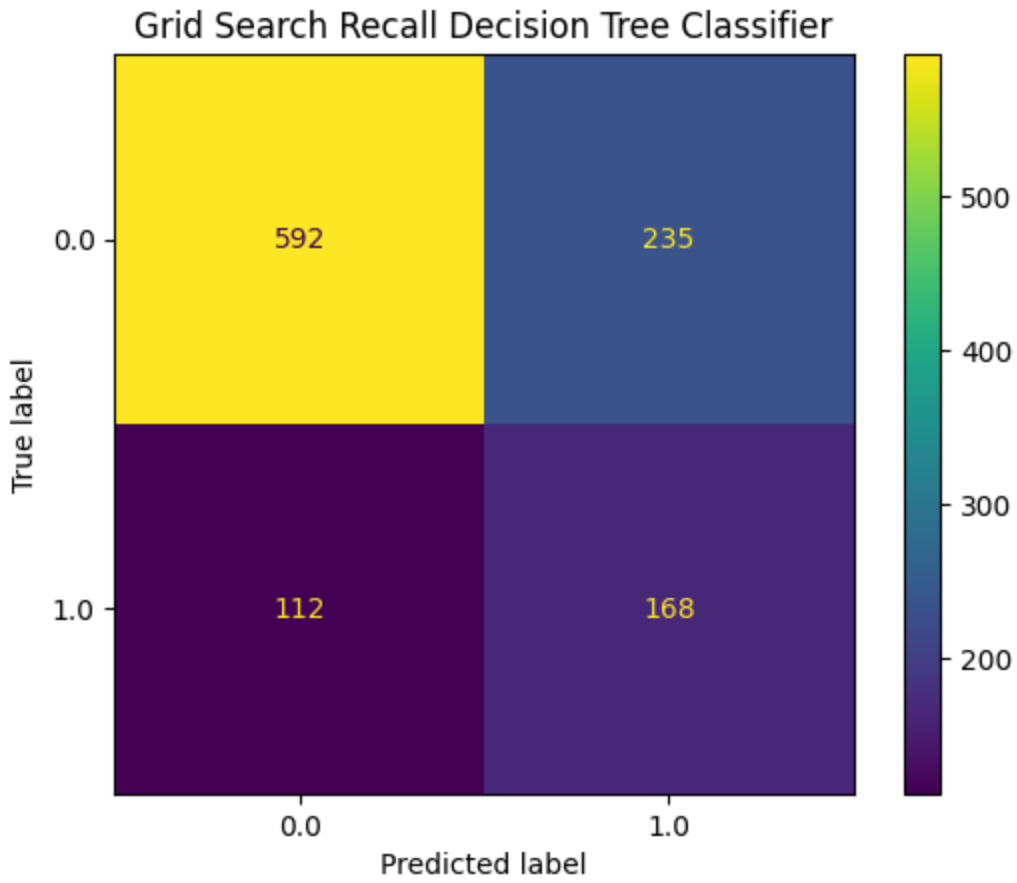
```
DecisionTreeClassifier(class_weight='balanced', max_depth=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [188]:

```
pred = tree_clf.predict(X_test)
```

In [189]:
```
ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('Grid Search Recall Decision Tree Classifier')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.84	0.72	0.77	827
1.0	0.42	0.60	0.49	280
accuracy			0.69	1107
macro avg	0.63	0.66	0.63	1107
weighted avg	0.73	0.69	0.70	1107

In [143]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [144]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [153]:
```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'ccp_alpha': [0.0, 0.01, 0.1],
    'class_weight': [None, 'balanced', {0:1, 1:2}, {0:1, 1:3}]
}

tree_clf = DecisionTreeClassifier()
scorer = make_scorer(recall_score)
grid_search = GridSearchCV(estimator=tree_clf, param_grid=param_grid,
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

best_tree = grid_search.best_estimator_
y_pred = best_tree.predict(X_test)
print("Test recall score:", recall_score(y_test, y_pred))
```

```
Best parameters: {'ccp_alpha': 0.01, 'class_weight': 'balanced', 'criteri
on': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_spli
t': 2}
Best score: 0.7420398622047244
Test recall score: 0.7285714285714285
```

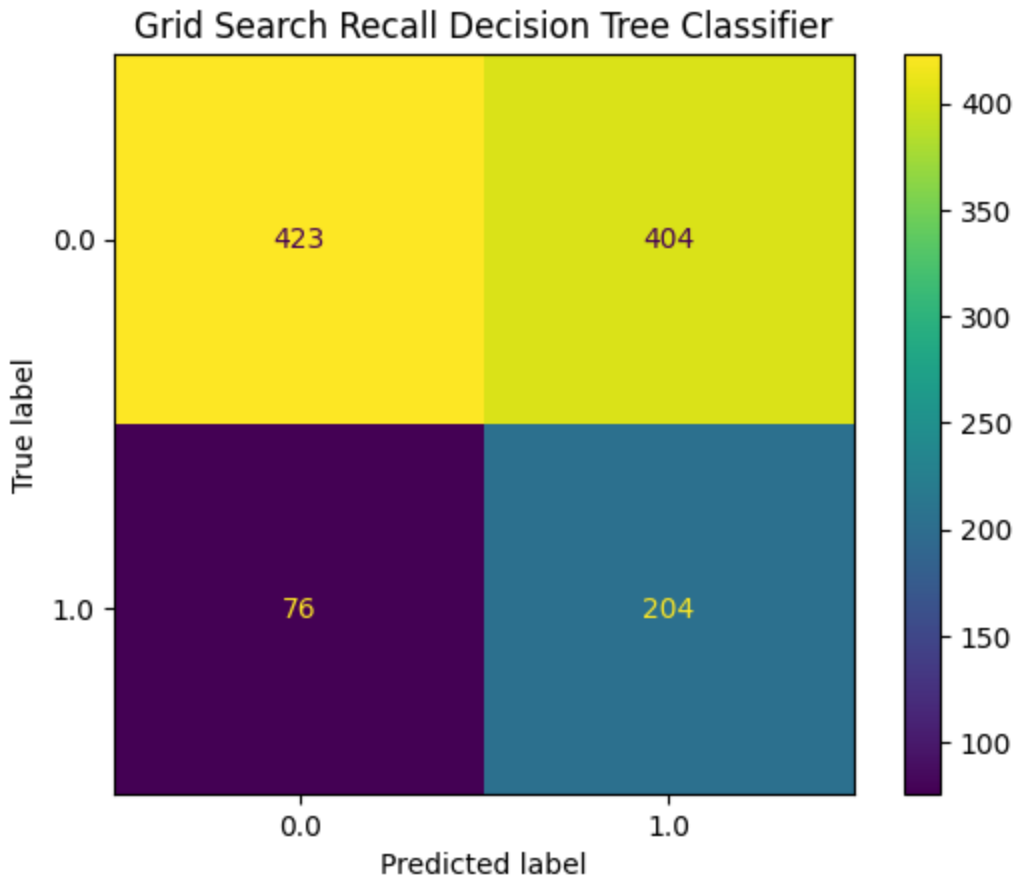
In [154]:
```
best_tree.fit(X_train, y_train)
```

Out[154]:
```
DecisionTreeClassifier(ccp_alpha=0.01, class_weight='balanced',
                       criterion='entropy', max_depth=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [155]:
```
pred = best_tree.predict(X_test)
```

In [156]:
```
ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('Grid Search Recall Decision Tree Classifier')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.85	0.51	0.64	827
1.0	0.34	0.73	0.46	280
accuracy			0.57	1107
macro avg	0.59	0.62	0.55	1107
weighted avg	0.72	0.57	0.59	1107

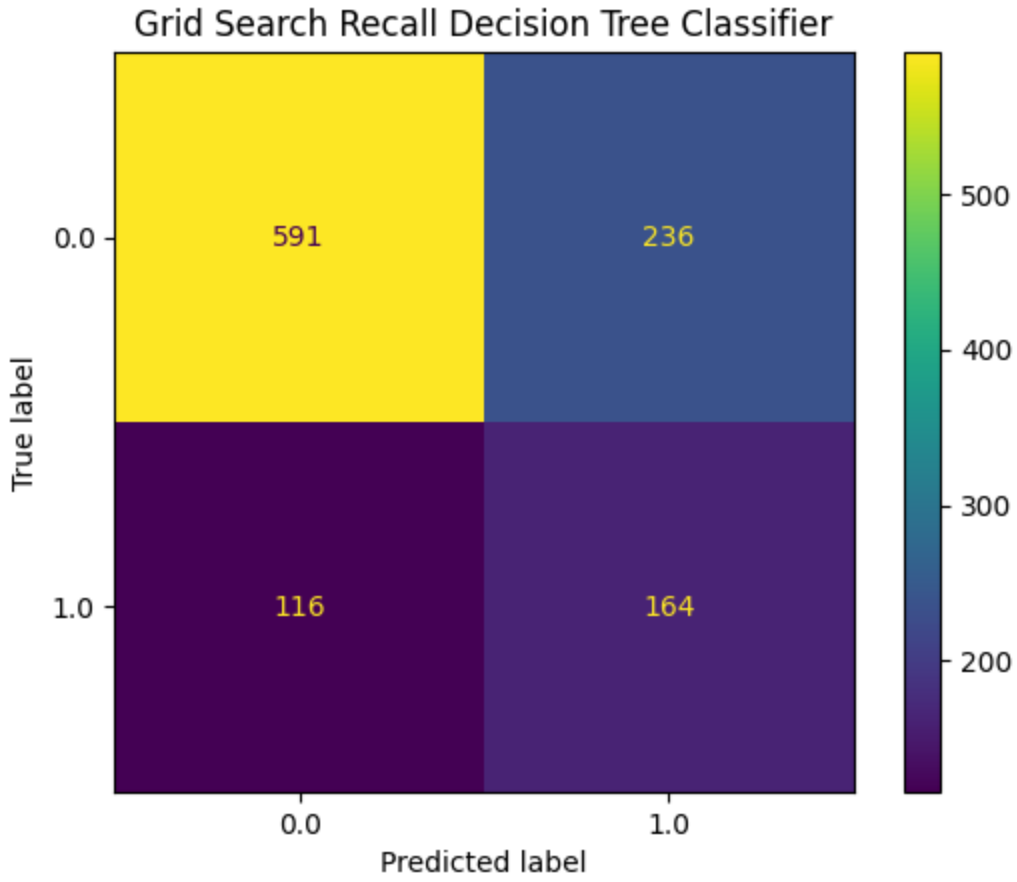
In [117]:
```
tree_clf = DecisionTreeClassifier(criterion='gini', class_weight='bala
tree_clf.fit(X_train, y_train)
```

Out[117]:
```
DecisionTreeClassifier(class_weight='balanced', max_depth=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [118]:
```
pred = tree_clf.predict(X_test)
```

In [119]:
```
ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('Grid Search Recall Decision Tree Classifier')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.84	0.71	0.77	827
1.0	0.41	0.59	0.48	280
accuracy			0.68	1107
macro avg	0.62	0.65	0.63	1107
weighted avg	0.73	0.68	0.70	1107

Try to improve on this model, attempt PCA

In [25]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [19]:
```
tree_pipeline = ImbPipeline([
    ('pca', PCA()),
    ('smote', SMOTE()),
    ('tree_clf', DecisionTreeClassifier())
])

param_grid = {
    'pca__n_components': [5, 10, 15],
    'smote__sampling_strategy': [0.55, 0.6, 0.65, 0.7],
    'tree_clf__criterion': ['gini', 'entropy'],
    'tree_clf__max_depth': [3, 4, 5, 10],
    'tree_clf__min_samples_split': [5, 10],
    'tree_clf__min_samples_leaf': [1, 2, 3],
    'tree_clf__class_weight': [None, 'balanced', {0:1, 1:1}, {0:1, 1:2
}

scorer = make_scorer(recall_score)
grid_search = GridSearchCV(tree_pipeline, param_grid, scoring=scorer,
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

best_tree = grid_search.best_estimator_
y_pred = best_tree.predict(X_test)
print("Test set recall:", recall_score(y_test, y_pred))
```

```
Best parameters: {'pca__n_components': 15, 'smote__sampling_strategy': 0.
7, 'tree_clf__class_weight': {0: 1, 1: 2}, 'tree_clf__criterion': 'gini',
'tree_clf__max_depth': 3, 'tree_clf__min_samples_leaf': 2, 'tree_clf__min
_samples_split': 10}
Best score: 0.8226747047244094
Test set recall: 0.6535714285714286
```

In [20]:
```
best_tree
```

Out[20]:
```
Pipeline(steps=[('pca', PCA(n_components=15)),
                ('smote', SMOTE(sampling_strategy=0.7)),
                ('tree_clf',
                 DecisionTreeClassifier(class_weight={0: 1, 1: 2}, max_de
pth=3,
                                        min_samples_leaf=2,
                                        min_samples_split=10))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

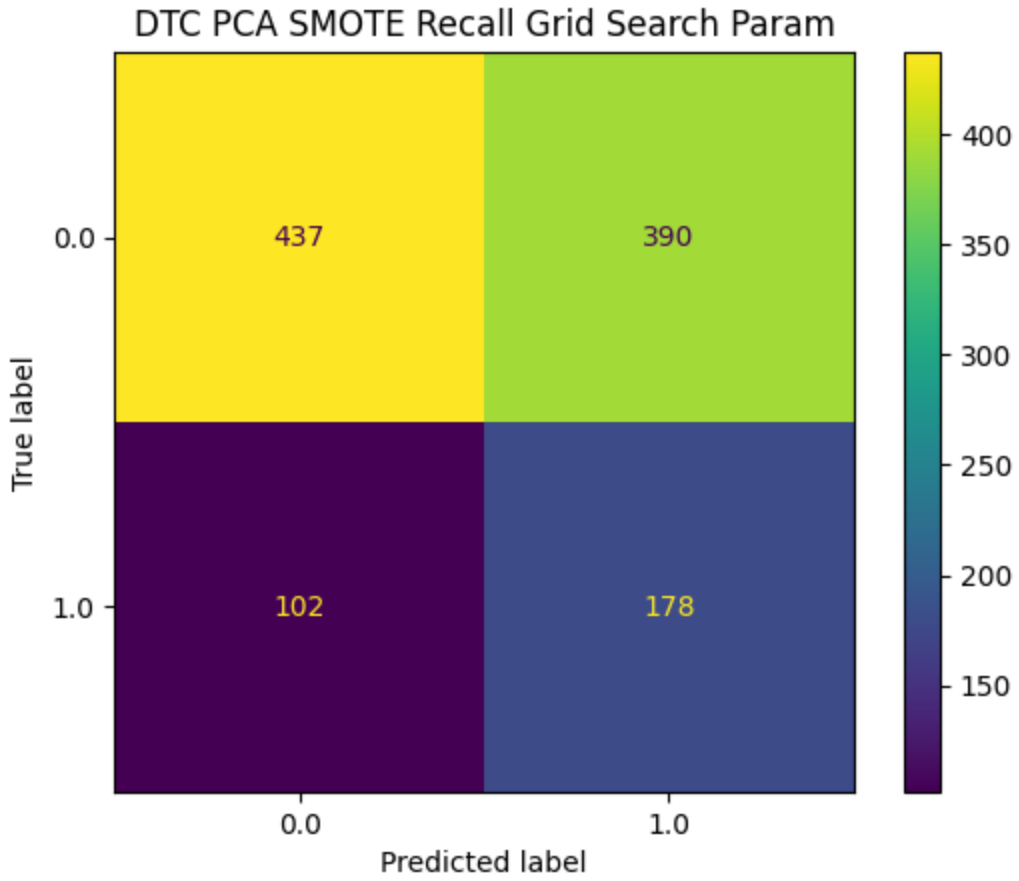
In [21]:
```
1 best_tree.fit(X_train, y_train)
```

Out[21]:
```
Pipeline(steps=[('pca', PCA(n_components=15)),
                ('smote', SMOTE(sampling_strategy=0.7)),
                ('tree_clf',
                 DecisionTreeClassifier(class_weight={0: 1, 1: 2}, max_de
pth=3,
                                        min_samples_leaf=2,
                                        min_samples_split=10))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [22]:
```
1 y_pred = best_tree.predict(X_test)
```

In [23]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('DTC PCA SMOTE Recall Grid Search Param')
plt.show()
print(classification_report(y_test, y_pred))
```




```
              precision    recall  f1-score   support

         0.0       0.81      0.53      0.64       827
         1.0       0.31      0.64      0.42       280

    accuracy                           0.56      1107
   macro avg       0.56      0.58      0.53      1107
weighted avg       0.68      0.56      0.58      1107
```

Try to further improve. Will try Borderline SMOTE

In [28]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [40]:
```
tree_pipeline = ImbPipeline([
    ('pca', PCA()),
    ('sm', BorderlineSMOTE()),
    ('tree_clf', DecisionTreeClassifier())
])

param_grid = {
    'pca__n_components': [5, 10, 15],
    'sm__sampling_strategy': [0.55, 0.6, 0.65, 0.7],
    'tree_clf__criterion': ['gini', 'entropy'],
    'tree_clf__max_depth': [3, 4, 5, 10],
    'tree_clf__min_samples_split': [5, 10],
    'tree_clf__min_samples_leaf': [1, 2, 3],
    'tree_clf__class_weight': [None, 'balanced', {0:1, 1:1}, {0:1, 1:2
}

scorer = make_scorer(recall_score)
grid_search = GridSearchCV(tree_pipeline, param_grid, scoring=scorer,
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

best_tree = grid_search.best_estimator_
y_pred = best_tree.predict(X_test)
print("Test set recall score:", recall_score(y_test, y_pred))
```

```
Best parameters: {'pca__n_components': 10, 'sm__sampling_strategy': 0.7,
'tree_clf__class_weight': {0: 1, 1: 2}, 'tree_clf__criterion': 'gini', 't
ree_clf__max_depth': 3, 'tree_clf__min_samples_leaf': 2, 'tree_clf__min_s
amples_split': 5}
Best score: 0.872613188976378
Test set recall score: 0.8642857142857143
```

In [41]:
```
best_tree.fit(X_train, y_train)
```

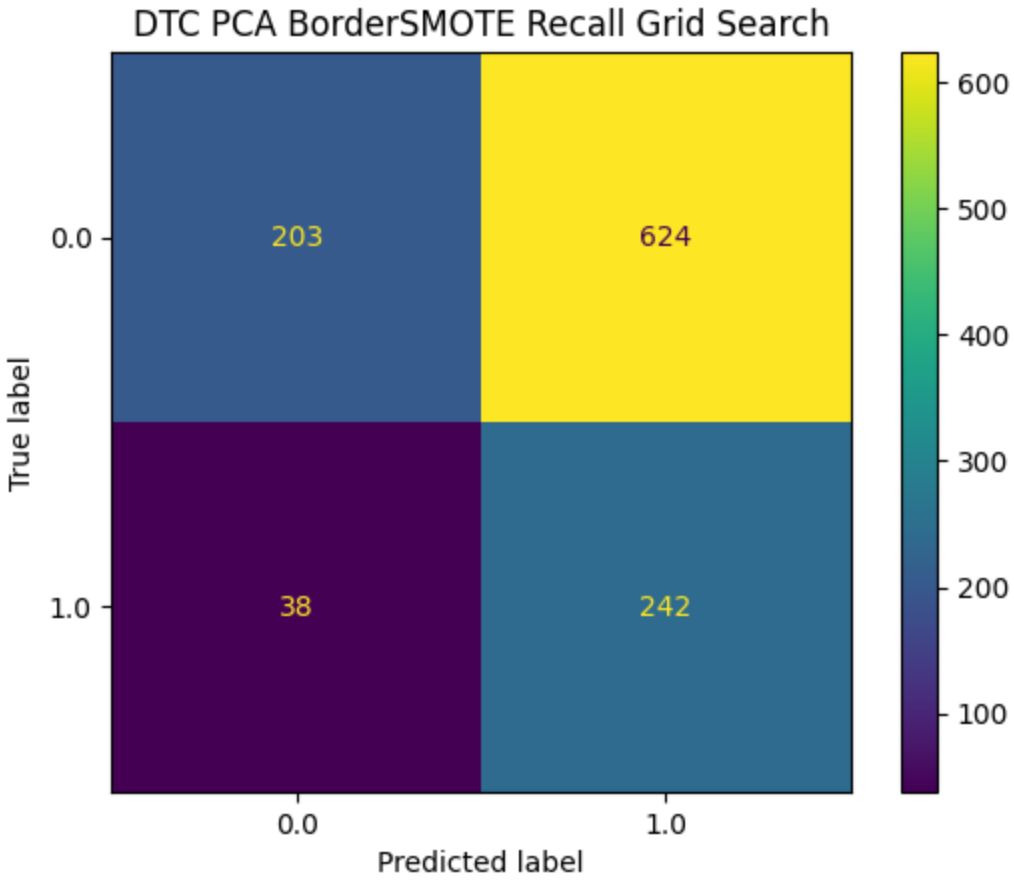
Out[41]:
```
Pipeline(steps=[('pca', PCA(n_components=10)),
                ('sm', BorderlineSMOTE(sampling_strategy=0.7)),
                ('tree_clf',
                 DecisionTreeClassifier(class_weight={0: 1, 1: 2}, max_de
pth=3,
                                        min_samples_leaf=2,
                                        min_samples_split=5))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:
```
y_pred = best_tree.predict(X_test)
```

In [42]:

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('DTC PCA BorderSMOTE Recall Grid Search')
plt.show()
print(classification_report(y_test, y_pred))
```




```
              precision    recall  f1-score   support

         0.0       0.84      0.25      0.38       827
         1.0       0.28      0.86      0.42       280

    accuracy                           0.40      1107
   macro avg       0.56      0.55      0.40      1107
weighted avg       0.70      0.40      0.39      1107
```

cond_pivoted_df, Decision Tree Classifier

In [89]:
```
1 cond_pivoted_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3688 entries, 0 to 3687
Data columns (total 39 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   season                          3688 non-null   int64
 1   Age                             3688 non-null   float64
 2   Throws                          3688 non-null   int64
 3   Surgery                         3688 non-null   float64
 4   BB_release_speed_weighted_avg   3688 non-null   float64
 5   FB_release_speed_weighted_avg   3688 non-null   float64
 6   OS_release_speed_weighted_avg   3688 non-null   float64
 7   OT_release_speed_weighted_avg   3688 non-null   float64
 8   SB_release_speed_weighted_avg   3688 non-null   float64
 9   BB_release_pos_x_weighted_avg   3688 non-null   float64
 10  FB_release_pos_x_weighted_avg   3688 non-null   float64
 11  OS_release_pos_x_weighted_avg   3688 non-null   float64
 12  OT_release_pos_x_weighted_avg   3688 non-null   float64
 13  SB_release_pos_x_weighted_avg   3688 non-null   float64
 14  BB_release_pos_y_weighted_avg   3688 non-null   float64
 15  FB_release_pos_y_weighted_avg   3688 non-null   float64
 16  OS_release_pos_y_weighted_avg   3688 non-null   float64
 17  OT_release_pos_y_weighted_avg   3688 non-null   float64
 18  SB_release_pos_y_weighted_avg   3688 non-null   float64
 19  BB_release_pos_z_weighted_avg   3688 non-null   float64
 20  FB_release_pos_z_weighted_avg   3688 non-null   float64
 21  OS_release_pos_z_weighted_avg   3688 non-null   float64
 22  OT_release_pos_z_weighted_avg   3688 non-null   float64
 23  SB_release_pos_z_weighted_avg   3688 non-null   float64
 24  BB_vx0_weighted_avg             3688 non-null   float64
 25  FB_vx0_weighted_avg             3688 non-null   float64
 26  OS_vx0_weighted_avg             3688 non-null   float64
 27  OT_vx0_weighted_avg             3688 non-null   float64
 28  SB_vx0_weighted_avg             3688 non-null   float64
 29  BB_vy0_weighted_avg             3688 non-null   float64
 30  FB_vy0_weighted_avg             3688 non-null   float64
 31  OS_vy0_weighted_avg             3688 non-null   float64
 32  OT_vy0_weighted_avg             3688 non-null   float64
 33  SB_vy0_weighted_avg             3688 non-null   float64
 34  BB_vz0_weighted_avg             3688 non-null   float64
 35  FB_vz0_weighted_avg             3688 non-null   float64
 36  OS_vz0_weighted_avg             3688 non-null   float64
 37  OT_vz0_weighted_avg             3688 non-null   float64
 38  SB_vz0_weighted_avg             3688 non-null   float64
dtypes: float64(37), int64(2)
memory usage: 1.1 MB
```

In [95]:
```
1 y = cond_pivoted_df['Surgery']
2 X = cond_pivoted_df.drop('Surgery', axis=1)
```

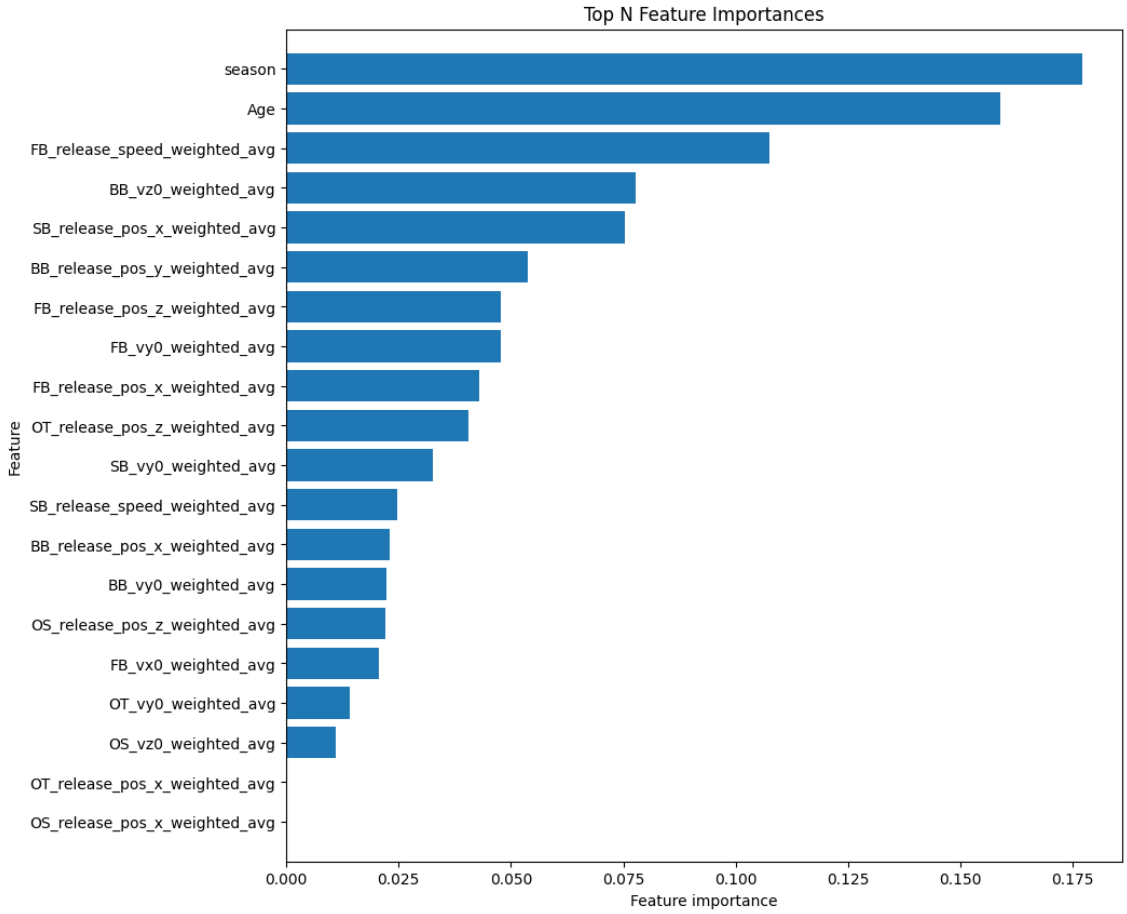
In [96]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [92]:
```
tree_clf = DecisionTreeClassifier(criterion='gini', max_depth=5)
tree_clf.fit(X_train, y_train)
```

Out[92]:
```
DecisionTreeClassifier(max_depth=5)
```

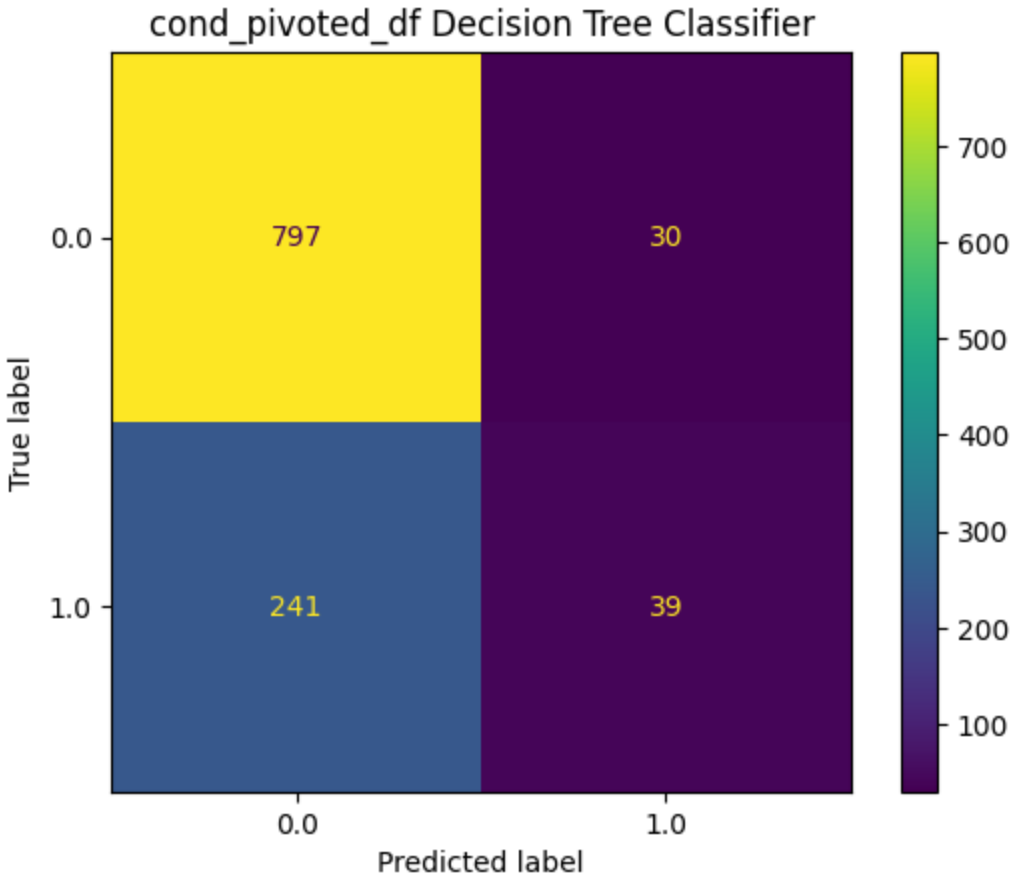
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [93]:
```
plot_feature_importances(tree_clf, n_top_features=20)
plt.show()
```



In [94]:

```
pred = tree_clf.predict(X_test)

ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('cond_pivoted_df Decision Tree Classifier')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.77	0.96	0.85	827
1.0	0.57	0.14	0.22	280
accuracy			0.76	1107
macro avg	0.67	0.55	0.54	1107
weighted avg	0.72	0.76	0.70	1107

cond_pivoted_df not predicting TP & FP well.

In [97]:
```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': [None, 'balanced', {0:1, 1:2}, {0:1, 1:3}]
}

tree_clf = DecisionTreeClassifier()
scorer = make_scorer(recall_score)
grid_search = GridSearchCV(estimator=tree_clf, param_grid=param_grid,

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Print the best parameters and the best score
print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

# Evaluate the best model found by GridSearchCV on the test set
best_tree = grid_search.best_estimator_
y_pred = best_tree.predict(X_test)
print("Test set accuracy:", accuracy_score(y_test, y_pred))

```

```
Best parameters: {'class_weight': 'balanced', 'criterion': 'entropy', 'ma
x_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 10}
Best score: 0.7031373031496063
Test set accuracy: 0.6621499548328816
```

In [98]:
```
y = cond_pivoted_df['Surgery']
X = cond_pivoted_df.drop('Surgery', axis=1)
```

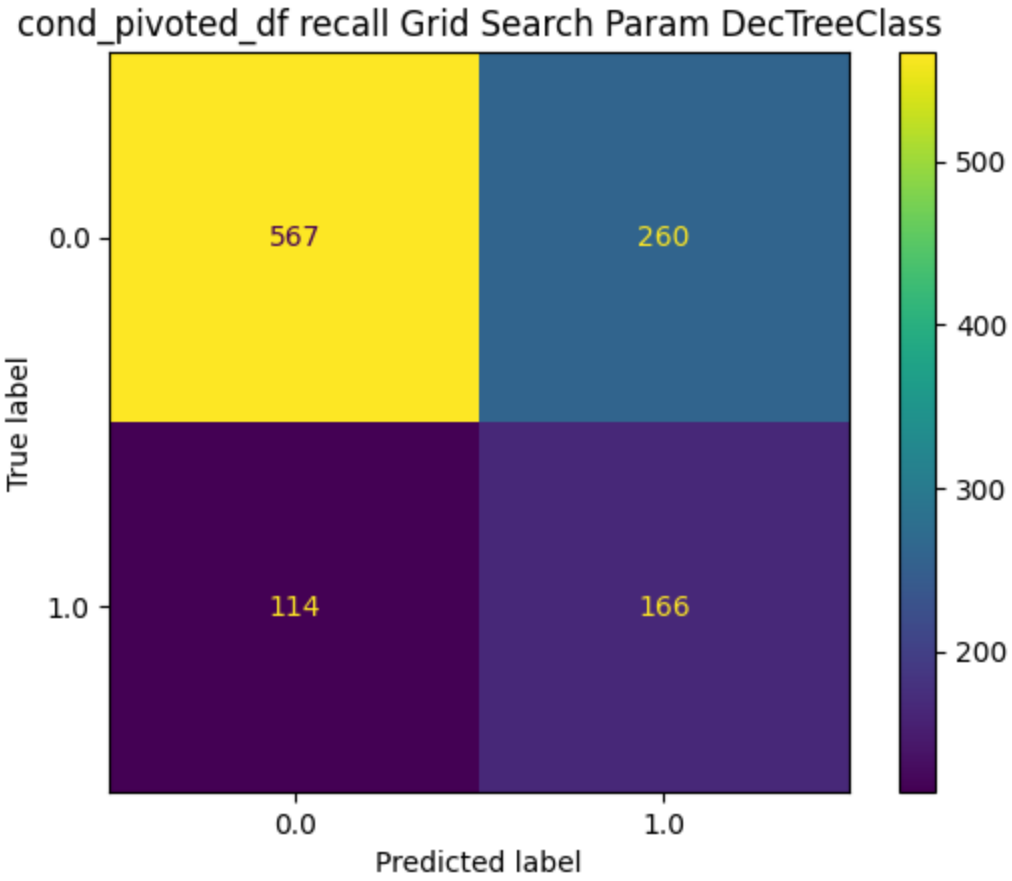
In [99]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [100]:
```
tree_clf = DecisionTreeClassifier(class_weight='balanced', criterion='
tree_clf.fit(X_train, y_train)
```

Out[100]:
```
DecisionTreeClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=5, min_samples_leaf=4, min_samples_split
=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [101]:
```
pred = tree_clf.predict(X_test)

ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('cond_pivoted_df recall Grid Search Param DecTreeClass')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.83	0.69	0.75	827
1.0	0.39	0.59	0.47	280
accuracy			0.66	1107
macro avg	0.61	0.64	0.61	1107
weighted avg	0.72	0.66	0.68	1107

Better, but FN is high compared to other models.

pivoted_df, Random Forest

In [102]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [103]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [104]:
```
forest_clf = RandomForestClassifier(n_estimators=100, max_depth=5)
forest_clf.fit(X_train, y_train)
```

Out[104]:
```
RandomForestClassifier(max_depth=5)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

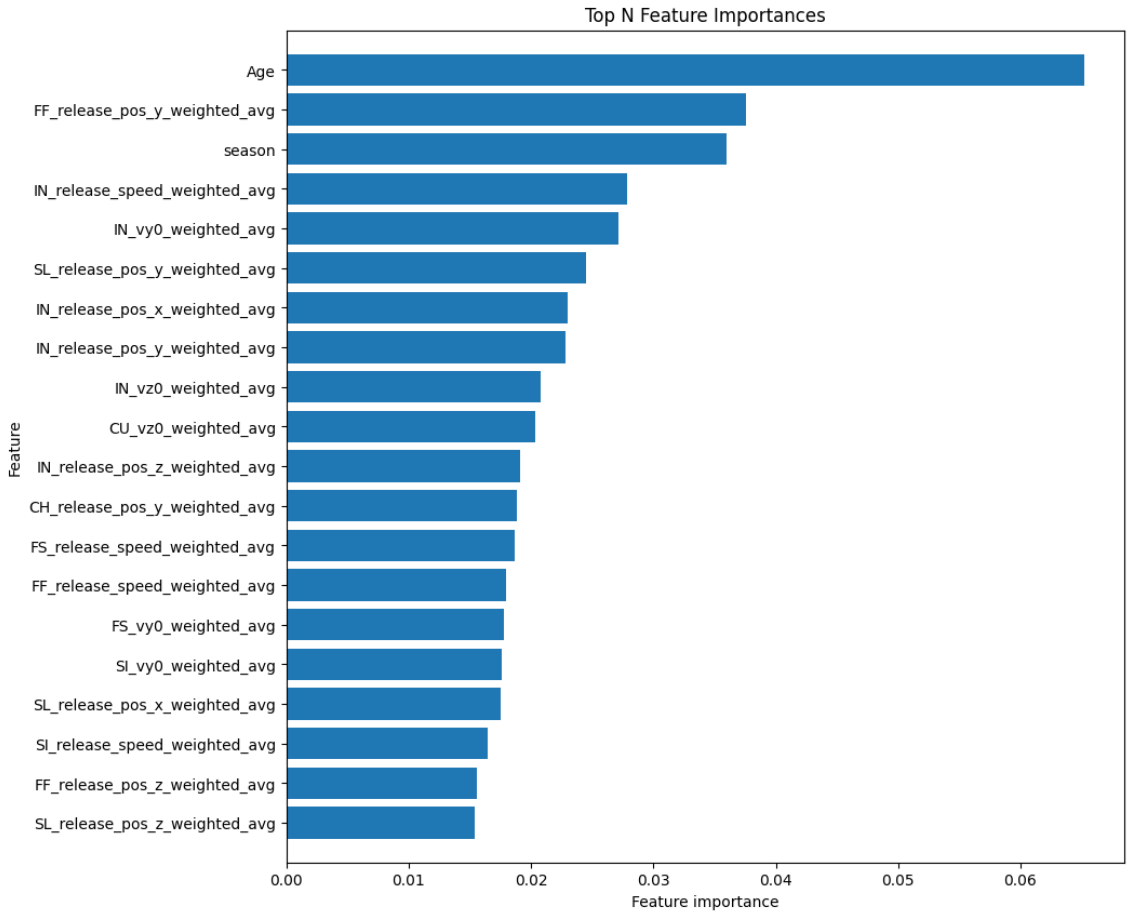
In [105]:
```
forest_clf.score(X_train, y_train)
```

Out[105]:
```
0.7667570709027509
```

In [106]:
```
forest_clf.score(X_test, y_test)
```

Out[106]:
```
0.7515808491418248
```

In [107]:
```
plot_feature_importances(forest_clf, n_top_features=20)
plt.show()
```



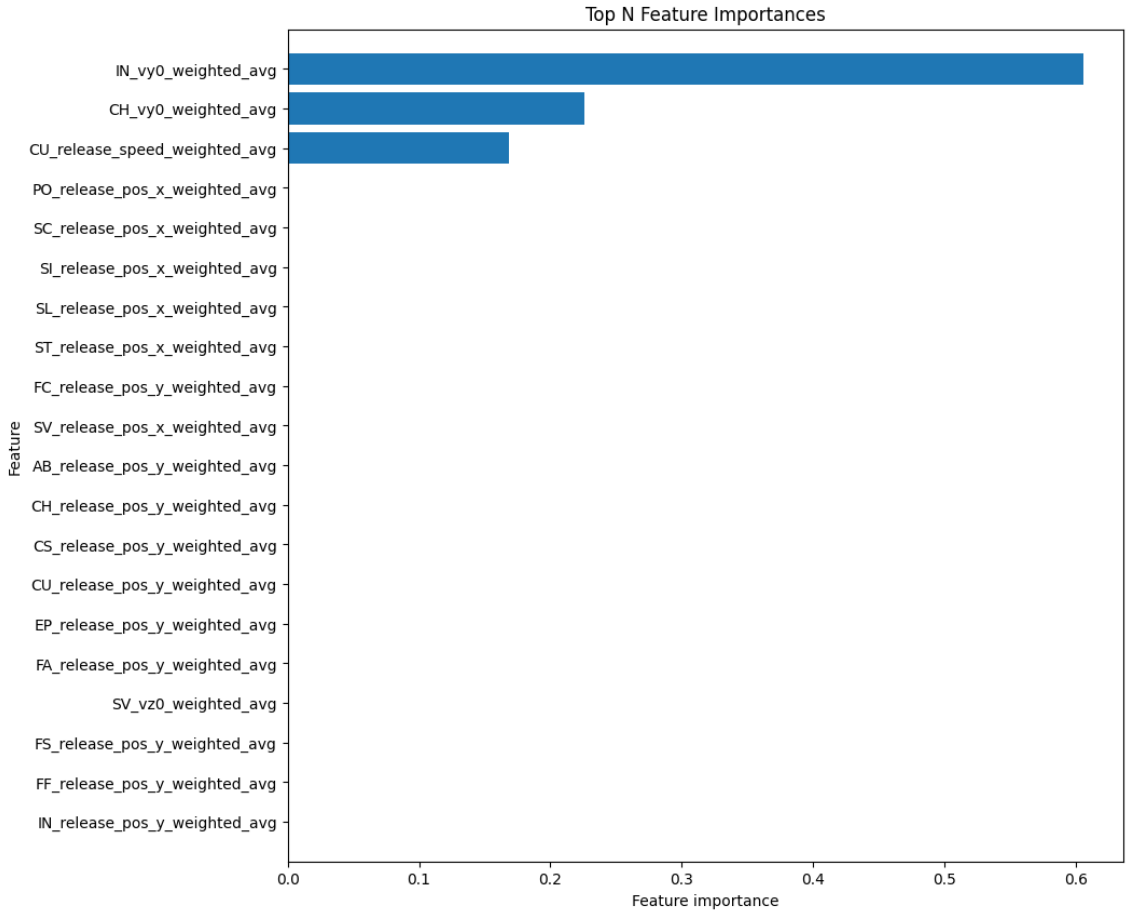
In [108]:
```
forest2_clf = RandomForestClassifier(n_estimators=5, max_features=10,
forest2_clf.fit(X_train, y_train)
```

Out[108]:
```
RandomForestClassifier(max_depth=2, max_features=10, n_estimators=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

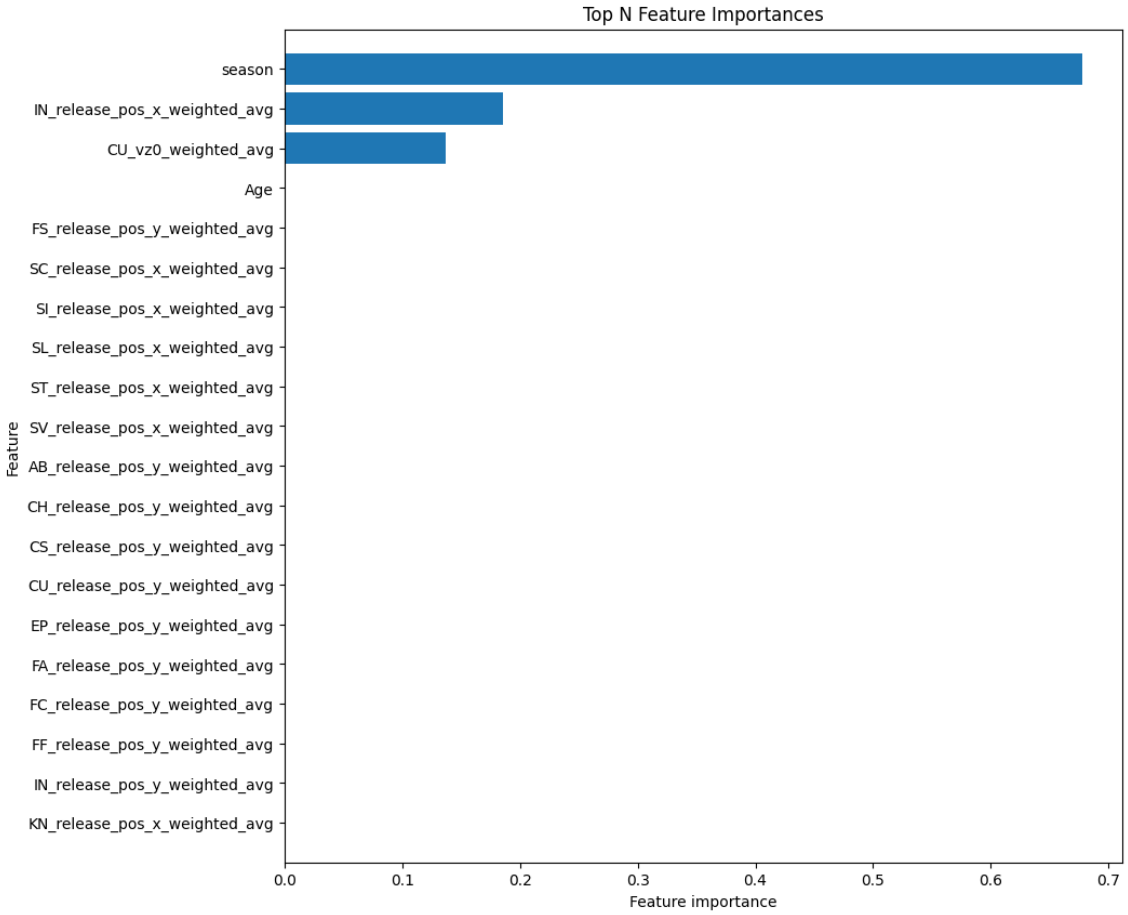
In [109]:
```
rf_tree_1 = forest2_clf.estimators_[0]
```

In [110]:
```
plot_feature_importances(rf_tree_1, n_top_features=20)
plt.show()
```



In [111]:
```
rf_tree_2 = forest2_clf.estimators_[1]
```

In [112]:
```
plot_feature_importances(rf_tree_2, n_top_features=20)
plt.show()
```



In [43]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [46]:
```
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': [None, 'balanced', {0:1, 1:2}, {0:1, 1:3}]
}

forest_clf = RandomForestClassifier()
accuracy_scorer = make_scorer(recall_score)
grid_search_rf = GridSearchCV(estimator=forest_clf, param_grid=param_g
grid_search_rf.fit(X_train, y_train)

print("Best parameters:", grid_search_rf.best_params_)
print("Best accuracy score:", grid_search_rf.best_score_)

best_rf = grid_search_rf.best_estimator_
y_pred_rf = best_rf.predict(X_test)

print("Test set recall score:", recall_score(y_test, y_pred_rf))
```

```
Best parameters: {'class_weight': 'balanced', 'criterion': 'entropy', 'ma
x_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimator
s': 100}
Best accuracy score: 0.6069635826771653
Test set recall score: 0.6071428571428571
```

In [47]:
```
#forest_clf = RandomForestClassifier(class_weight='balanced', criterio
best_rf.fit(X_train, y_train)
```

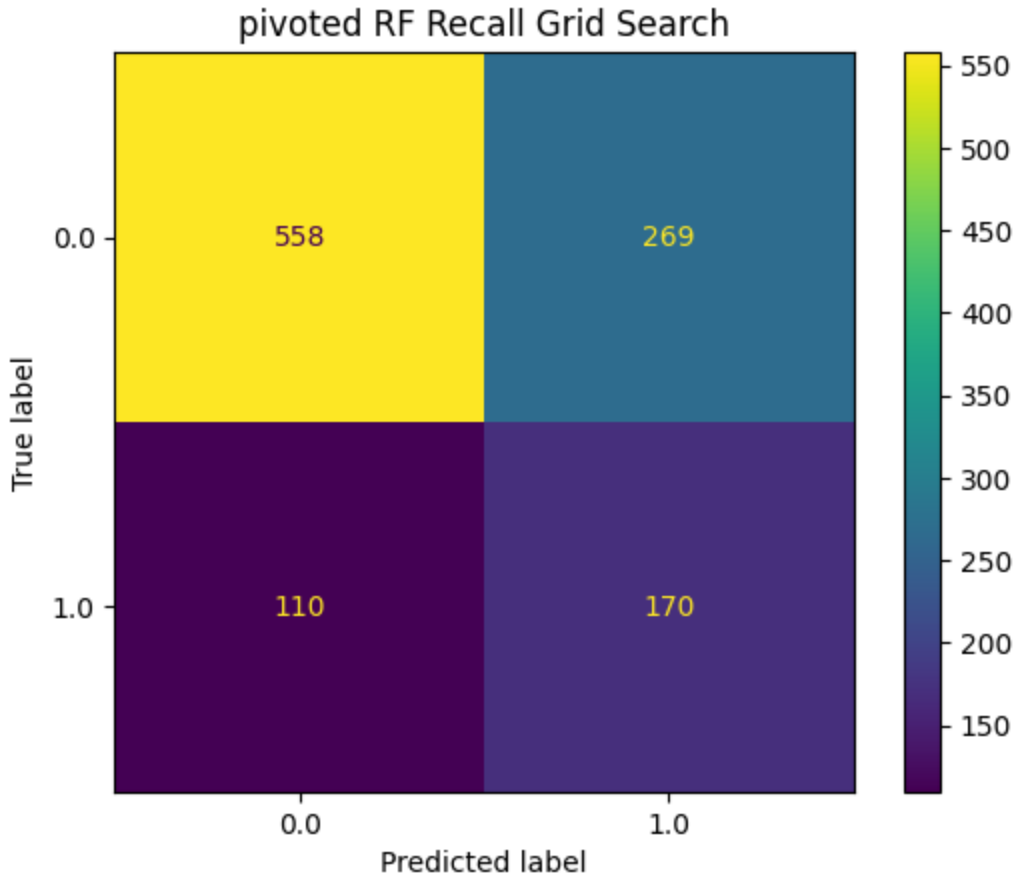
Out[47]:
```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=5, min_samples_leaf=4)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [48]:
```
pred = best_rf.predict(X_test)
```

In [49]:

```
1
2 ConfusionMatrixDisplay.from_predictions(y_test, pred)
3 plt.title('pivoted RF Recall Grid Search')
4 plt.show()
5 print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.84	0.67	0.75	827
1.0	0.39	0.61	0.47	280
accuracy			0.66	1107
macro avg	0.61	0.64	0.61	1107
weighted avg	0.72	0.66	0.68	1107

Again, TN & TP are good, but FP is too small compared to FN. Will try this with recall score.

In [132]:
```
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': [None, 'balanced', {0:1, 1:2}, {0:1, 1:3}]
}

forest_clf = RandomForestClassifier()
recall_scorer = make_scorer(recall_score)
grid_search_rf = GridSearchCV(estimator=forest_clf, param_grid=param_g
grid_search_rf.fit(X_train, y_train)

print("Best parameters:", grid_search_rf.best_params_)
print("Best accuracy score:", grid_search_rf.best_score_)

best_rf = grid_search_rf.best_estimator_
y_pred_rf = best_rf.predict(X_test)

print("Test set accuracy score:", recall_score(y_test, y_pred_rf))
```

```
Best parameters: {'class_weight': 'balanced', 'criterion': 'entropy', 'ma
x_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estima
tors': 100}
Best accuracy score: 0.810923935045657
Test set accuracy score: 0.34285714285714286
```

In [133]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

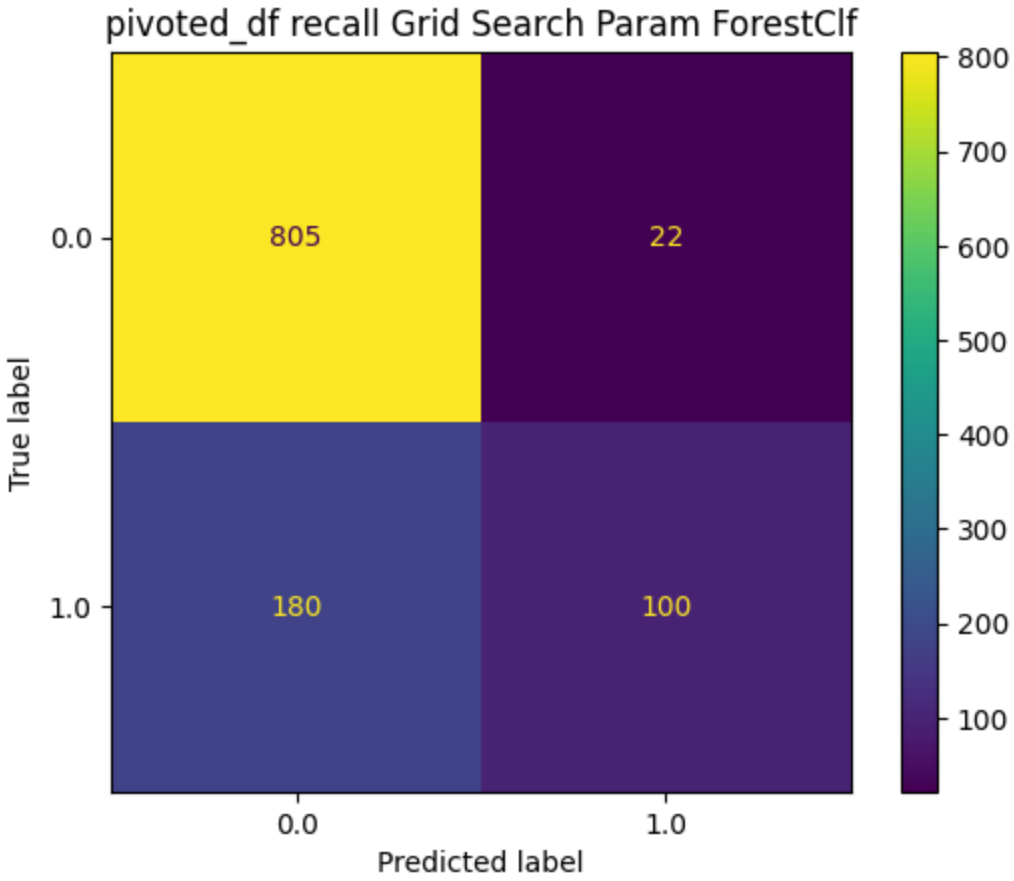
In [134]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [136]:
```
forest_clf = RandomForestClassifier(class_weight='balanced', criterion
forest_clf.fit(X_train, y_train)
```

Out[136]:
```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       min_samples_leaf=2, min_samples_split=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [137]:
```
pred = forest_clf.predict(X_test)

ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('pivoted_df recall Grid Search Param ForestClf')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.82	0.97	0.89	827
1.0	0.82	0.36	0.50	280
accuracy			0.82	1107
macro avg	0.82	0.67	0.69	1107
weighted avg	0.82	0.82	0.79	1107

Not great. Would like to see lower FN and higher TP & FP.

cond_pivoted_df, Random Forest

In [138]:
```
y = cond_pivoted_df['Surgery']
X = cond_pivoted_df.drop('Surgery', axis=1)
```

In [139]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [140]:
```
forest_clf = RandomForestClassifier(n_estimators=100, max_depth=5)
forest_clf.fit(X_train, y_train)
```

Out[140]:
```
RandomForestClassifier(max_depth=5)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

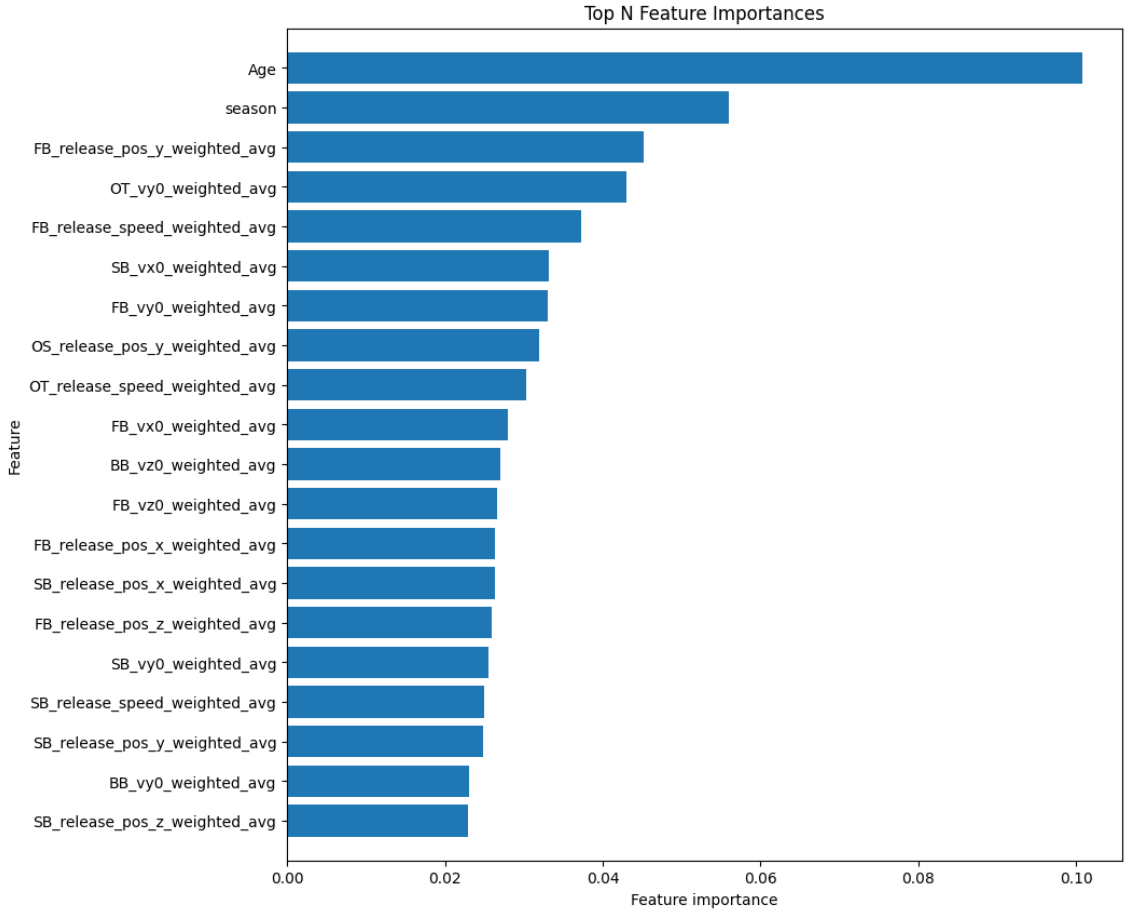
In [141]:
```
forest_clf.score(X_train, y_train)
```

Out[141]:
```
0.7729562185199536
```

In [142]:
```
forest_clf.score(X_test, y_test)
```

Out[142]:
```
0.7497741644083108
```

In [143]:
```
plot_feature_importances(forest_clf, n_top_features=20)
plt.show()
```



In [146]:
```
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': [None, 'balanced', {0:1, 1:2}, {0:1, 1:3}]
}

forest_clf = RandomForestClassifier()
accuracy_scorer = make_scorer(accuracy_score)
grid_search_rf = GridSearchCV(estimator=forest_clf, param_grid=param_g
grid_search_rf.fit(X_train, y_train)

print("Best parameters:", grid_search_rf.best_params_)
print("Best accuracy score:", grid_search_rf.best_score_)

best_rf = grid_search_rf.best_estimator_
y_pred_rf = best_rf.predict(X_test)

print("Test set accuracy score:", accuracy_score(y_test, y_pred_rf))
```

```
Best parameters: {'class_weight': 'balanced', 'criterion': 'entropy', 'ma
x_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimator
s': 200}
Best accuracy score: 0.8085953548348404
Test set accuracy score: 0.8121047877145439
```

In [147]:
```
y = cond_pivoted_df['Surgery']
X = cond_pivoted_df.drop('Surgery', axis=1)
```

In [148]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

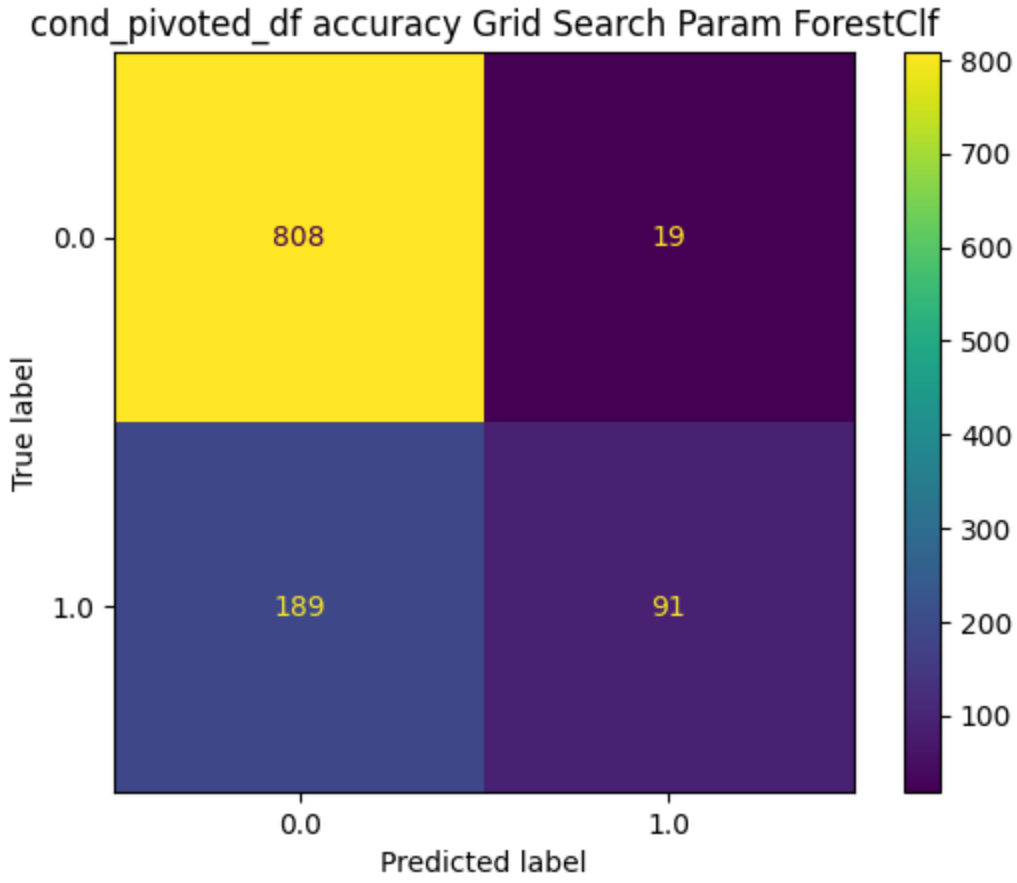
In [149]:
```
forest_clf = RandomForestClassifier(class_weight='balanced', criterion
forest_clf.fit(X_train, y_train)
```

Out[149]:
```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=15, min_samples_leaf=2, min_samples_spli
t=5,
                       n_estimators=200)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [151]:

```
pred = forest_clf.predict(X_test)

ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('cond_pivoted_df accuracy Grid Search Param ForestClf')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.81	0.98	0.89	827
1.0	0.83	0.33	0.47	280
accuracy			0.81	1107
macro avg	0.82	0.65	0.68	1107
weighted avg	0.81	0.81	0.78	1107

Try recall score.

In [154]:
```
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': [None, 'balanced', {0:1, 1:2}, {0:1, 1:3}]
}

forest_clf = RandomForestClassifier()
recall_scorer = make_scorer(recall_score)
grid_search_rf = GridSearchCV(estimator=forest_clf, param_grid=param_g
grid_search_rf.fit(X_train, y_train)

print("Best parameters:", grid_search_rf.best_params_)
print("Best accuracy score:", grid_search_rf.best_score_)

best_rf = grid_search_rf.best_estimator_
y_pred_rf = best_rf.predict(X_test)

print("Test set accuracy score:", recall_score(y_test, y_pred_rf))
```

```
Best parameters: {'class_weight': 'balanced', 'criterion': 'entropy', 'ma
x_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimator
s': 100}
Best accuracy score: 0.8074348132487668
Test set accuracy score: 0.33214285714285713
```

In [155]:
```
y = cond_pivoted_df['Surgery']
X = cond_pivoted_df.drop('Surgery', axis=1)
```

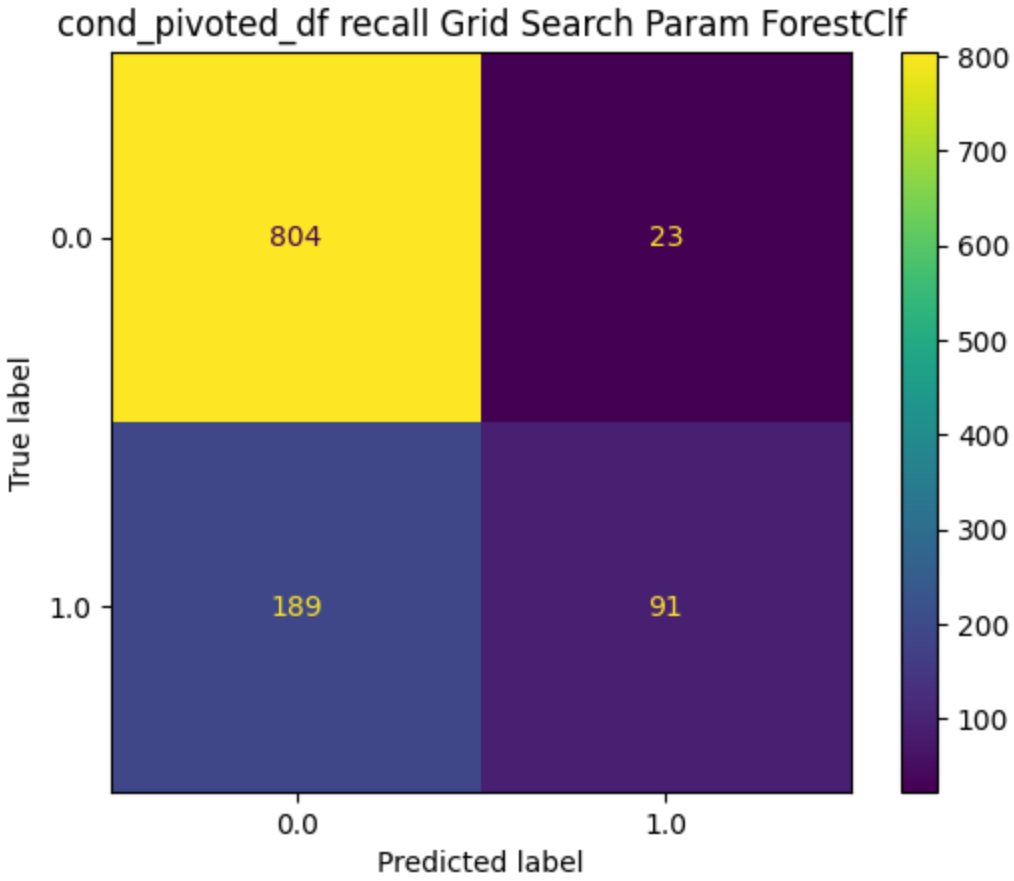
In [156]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [157]:
```
forest_clf = RandomForestClassifier(class_weight='balanced', criterion
forest_clf.fit(X_train, y_train)
```

Out[157]:
```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=15, min_samples_leaf=2, min_samples_spli
t=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [158]:
```
pred = forest_clf.predict(X_test)

ConfusionMatrixDisplay.from_predictions(y_test, pred)
plt.title('cond_pivoted_df recall Grid Search Param ForestClf')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.81	0.97	0.88	827
1.0	0.80	0.33	0.46	280
accuracy			0.81	1107
macro avg	0.80	0.65	0.67	1107
weighted avg	0.81	0.81	0.78	1107

Odd. Scores are very similar. Is there a way to better these scores? May just have to move onto another model. XG Boost?

pivoted_df XG Boost

In [171]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [172]:
```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [167]:
```
1 xgb_clf = XGBClassifier(use_label_encoder=False, eval_metric='logloss'
```

In [168]:
```
1 xgb_clf.fit(X_train, y_train)
```

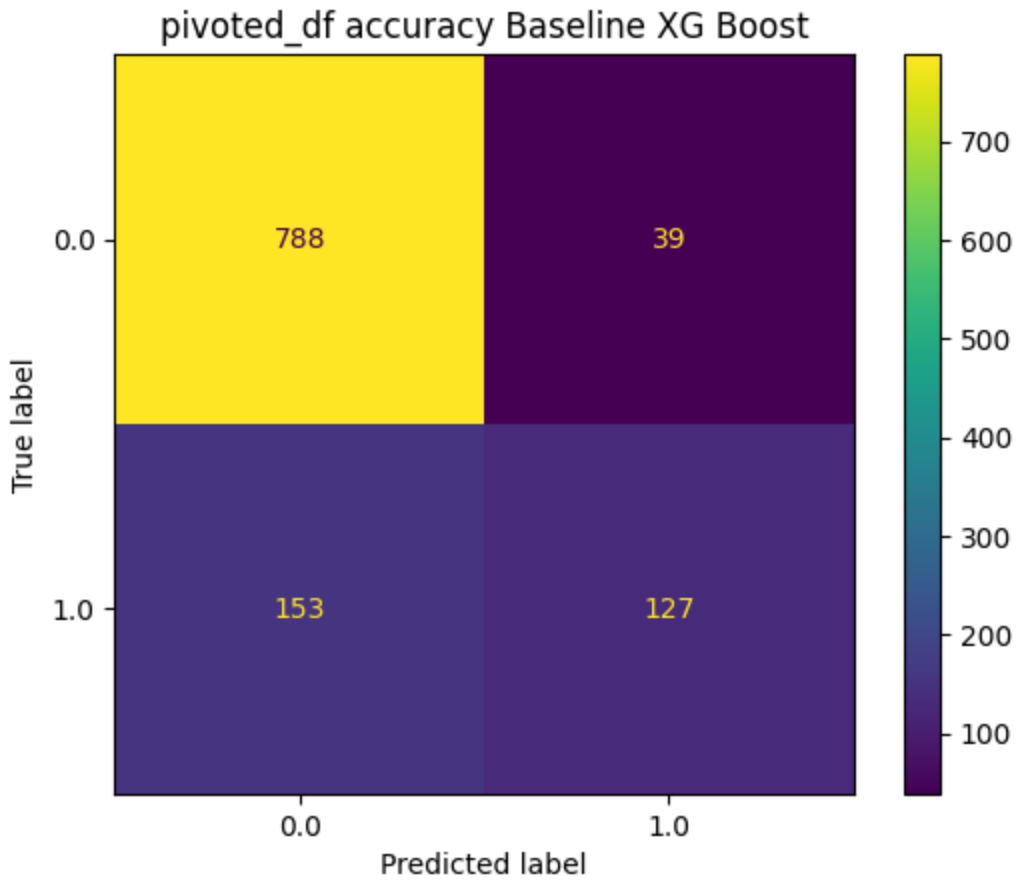
Out[168]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=N
one,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=Non
e,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimator
s=None,
              n_jobs=None, num_parallel_tree=None, random_state=None,
...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [169]:
```
1 y_pred = xgb_clf.predict(X_test)
```

In [170]:

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('pivoted_df accuracy Baseline XG Boost')
plt.show()
print(classification_report(y_test, pred))
```




```
              precision    recall  f1-score   support

         0.0       0.81      0.97      0.88       827
         1.0       0.80      0.33      0.46       280

    accuracy                           0.81      1107
   macro avg       0.80      0.65      0.67      1107
weighted avg       0.81      0.81      0.78      1107
```

Pretty good. Would like to see lower FN and higher scores for everything else. Will try GridSearch.

In [183]:
```
param_grid_xgb = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1],
    'colsample_bytree': [0.8, 1]
}

xgb_clf = XGBClassifier(objective='binary:logistic', use_label_encoder

grid_search_xgb = GridSearchCV(estimator=xgb_clf, param_grid=param_gri
grid_search_xgb.fit(X_train, y_train)

print("Best parameters:", grid_search_xgb.best_params_)
print("Best recall score:", grid_search_xgb.best_score_)

best_xgb = grid_search_xgb.best_estimator_
y_pred_best_xgb = best_xgb.predict(X_test)
print("Test set recall score:", recall_score(y_test, y_pred_best_xgb))
print(classification_report(y_test, y_pred_best_xgb))
```

```
Best parameters: {'colsample_bytree': 1, 'learning_rate': 0.2, 'max_dept
h': 7, 'n_estimators': 200, 'subsample': 0.8}
Best recall score: 0.4323941929133858
Test set recall score: 0.45357142857142857
              precision    recall  f1-score   support

         0.0       0.84      0.96      0.89       827
         1.0       0.79      0.45      0.58       280

    accuracy                           0.83      1107
   macro avg       0.81      0.71      0.74      1107
weighted avg       0.83      0.83      0.81      1107
```

In [184]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [185]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [186]:
```
xgb_clf = XGBClassifier(colsample_bytree=1, learning_rate=0.2, max_dep
```

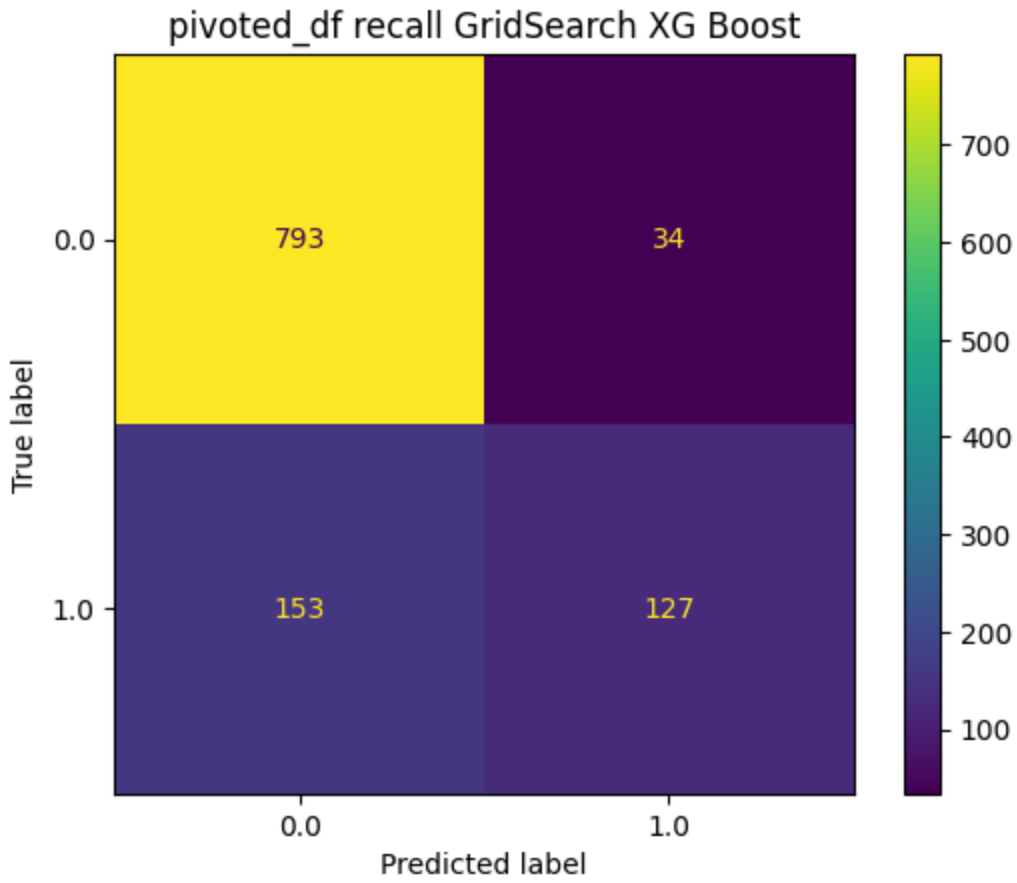
In [187]:
```
1 xgb_clf.fit(X_train, y_train)
```

Out[187]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None, colsample_by
tree=1,
              device=None, early_stopping_rounds=None, enable_categorical
=False,
              eval_metric='logloss', feature_types=None, gamma=None,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.2, max_bin=No
ne,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=7, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=No
ne,
              multi_strategy=None, n_estimators=200, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [188]:
```
1 y_pred = xgb_clf.predict(X_test)
```

In [190]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('pivoted_df recall GridSearch XG Boost')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.81	0.97	0.88	827
1.0	0.80	0.33	0.46	280
accuracy			0.81	1107
macro avg	0.80	0.65	0.67	1107
weighted avg	0.81	0.81	0.78	1107

Essentially the same score as the previous model. Will try with cond_pivoted_df now.

cond_pivoted_df, XG Boost

In [191]:
```
y = cond_pivoted_df['Surgery']
X = cond_pivoted_df.drop('Surgery', axis=1)
```

In [192]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [193]:
```
1 xgb_clf = XGBClassifier(use_label_encoder=False, eval_metric='logloss'
```

In [194]:
```
1 xgb_clf.fit(X_train, y_train)
```

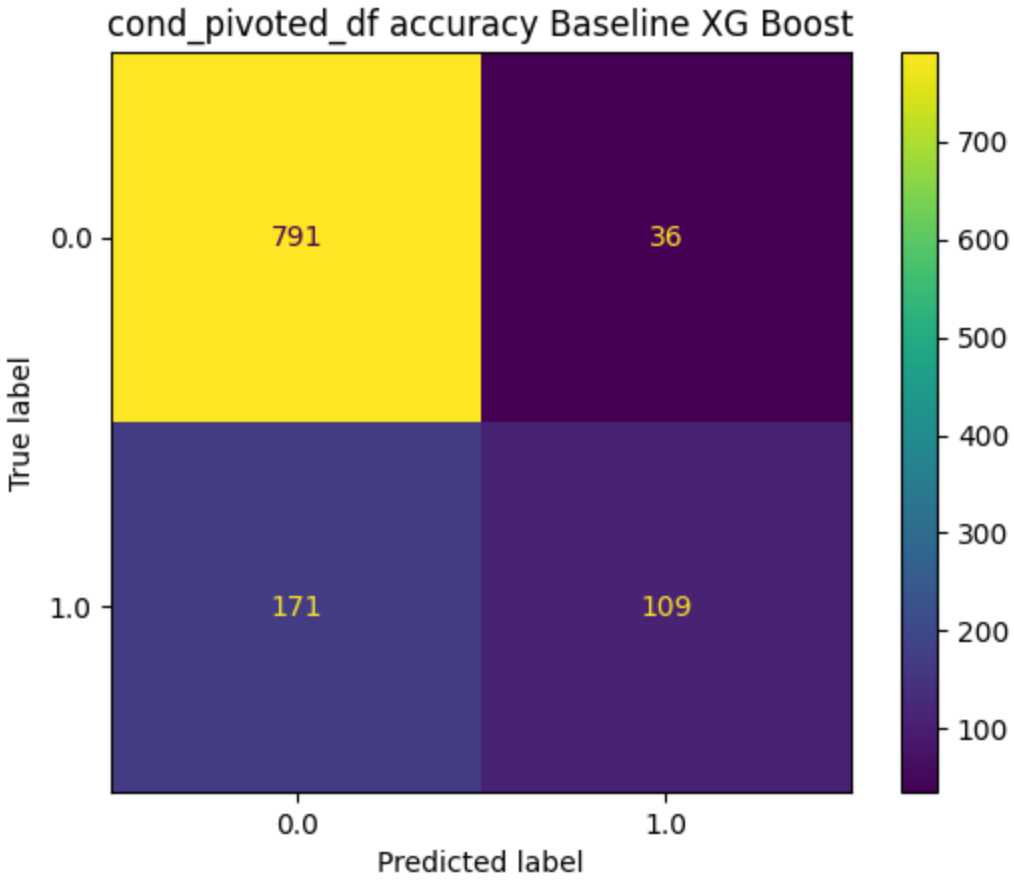
Out[194]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=N
one,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=Non
e,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimator
s=None,
              n_jobs=None, num_parallel_tree=None, random_state=None,
...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [195]:
```
1 y_pred = xgb_clf.predict(X_test)
```

In [196]:

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('cond_pivoted_df accuracy Baseline XG Boost')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.81	0.97	0.88	827
1.0	0.80	0.33	0.46	280
accuracy			0.81	1107
macro avg	0.80	0.65	0.67	1107
weighted avg	0.81	0.81	0.78	1107

In [202]:
```
param_grid_xgb = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1],
    'colsample_bytree': [0.8, 1]
}

cond_xgb_clf = XGBClassifier(objective='binary:logistic', use_label_en

grid_search_xgb = GridSearchCV(estimator=cond_xgb_clf, param_grid=para
grid_search_xgb.fit(X_train, y_train)

print("Best parameters:", grid_search_xgb.best_params_)
print("Best recall score:", grid_search_xgb.best_score_)

best_xgb = grid_search_xgb.best_estimator_
y_pred_best_xgb = best_xgb.predict(X_test)
print("Test set recall score:", recall_score(y_test, y_pred_best_xgb))
print(classification_report(y_test, y_pred_best_xgb))
```

```
Best parameters: {'colsample_bytree': 1, 'learning_rate': 0.2, 'max_dept
h': 7, 'n_estimators': 200, 'subsample': 0.8}
Best recall score: 0.4323941929133858
Test set recall score: 0.45357142857142857
              precision    recall  f1-score   support

         0.0       0.84      0.96      0.89       827
         1.0       0.79      0.45      0.58       280

    accuracy                           0.83      1107
   macro avg       0.81      0.71      0.74      1107
weighted avg       0.83      0.83      0.81      1107
```

In [203]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [204]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [205]:
```
cond_xgb_clf = XGBClassifier(colsample_bytree=1, learning_rate=0.2, ma
```

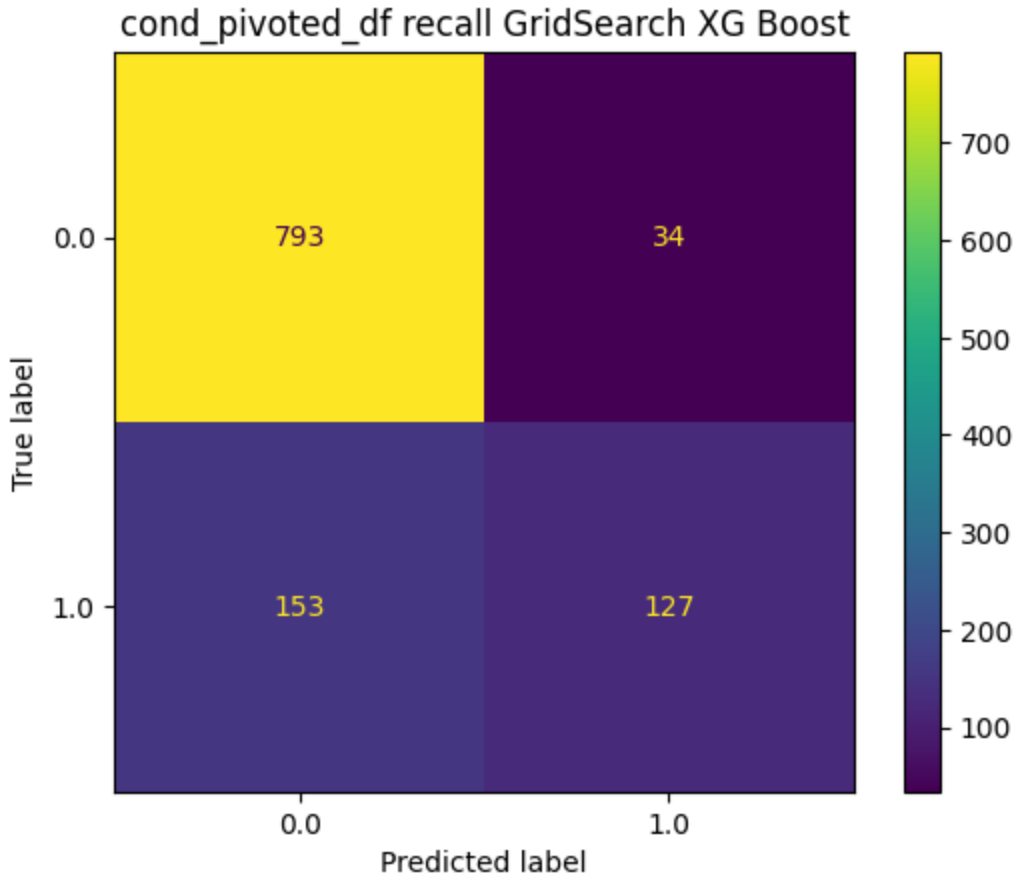
In [206]:
```
1 cond_xgb_clf.fit(X_train, y_train)
```

Out[206]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None, colsample_by
tree=1,
              device=None, early_stopping_rounds=None, enable_categorical
=False,
              eval_metric='logloss', feature_types=None, gamma=None,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.2, max_bin=No
ne,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=7, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=No
ne,
              multi_strategy=None, n_estimators=200, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [207]:
```
1 y_pred = cond_xgb_clf.predict(X_test)
```

In [208]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('cond_pivoted_df recall GridSearch XG Boost')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.81	0.97	0.88	827
1.0	0.80	0.33	0.46	280
accuracy			0.81	1107
macro avg	0.80	0.65	0.67	1107
weighted avg	0.81	0.81	0.78	1107

A little bit better. Will have to look into how to better tune all the various models parameters tomorrow.

Try to reduce dimensionality. Will work with Clustering and K Means on pivoted_df.

In [77]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [78]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [79]:
```
def cross_validation(X_train, y_train, k, num_split = 5):

    X_train = X_train.values
    y_train = y_train.values

    recall_train_list = []
    recall_val_list = []

    for train_index, valid_index in KFold(n_splits = num_split, shuffl

        # train and validation splitting
        X_train_fold, X_val_fold = X_train[train_index], X_train[valid
        y_train_fold, y_val_fold = y_train[train_index], y_train[valid

        #create/fit the Standard scaler on the train fold
        scaler = StandardScaler()
        X_train_fold_scaled = scaler.fit_transform(X_train_fold)
        # transform validation fold
        X_val_fold_scaled = scaler.transform(X_val_fold)

        # create/fit knearest neighbor
        knn = KNeighborsClassifier(n_neighbors = k)
        knn.fit(X_train_fold_scaled, y_train_fold)

        # now how did we do?
        y_train_pred = knn.predict(X_train_fold_scaled)
        y_val_pred = knn.predict(X_val_fold_scaled)

        recall_train = recall_score(y_train_fold, y_train_pred)
        recall_val = recall_score(y_val_fold, y_val_pred)

        recall_train_list.append(recall_train)
        recall_val_list.append(recall_val)

    return {'k': k, 'train_recall': np.mean(recall_train_list), 'valid
```

In [80]:
```
results = []
for k in np.arange(1, 100):
    results.append(cross_validation(X_train, y_train, k, 5))

# Create DataFrame from results
crossval_df = pd.DataFrame(results)

print(crossval_df.head())
```

```
   k  train_recall  validation_recall
0  1      1.000000           0.539758
1  2      0.522691           0.261689
2  3      0.665922           0.399245
3  4      0.380519           0.184001
4  5      0.487719           0.274063
```

In [81]:

```
# empty dataframe
crossval_df = pd.DataFrame(columns = ['k', 'train_recall', 'validation_
# append results for each value of k
for k in np.arange(1,100):
    crossval_df = crossval_df.append(cross_validation(X_train, y_train
```

```
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
```

```
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
```

```
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
```

```
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
```

```
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
```

```
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
```

```
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
```

```
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
```

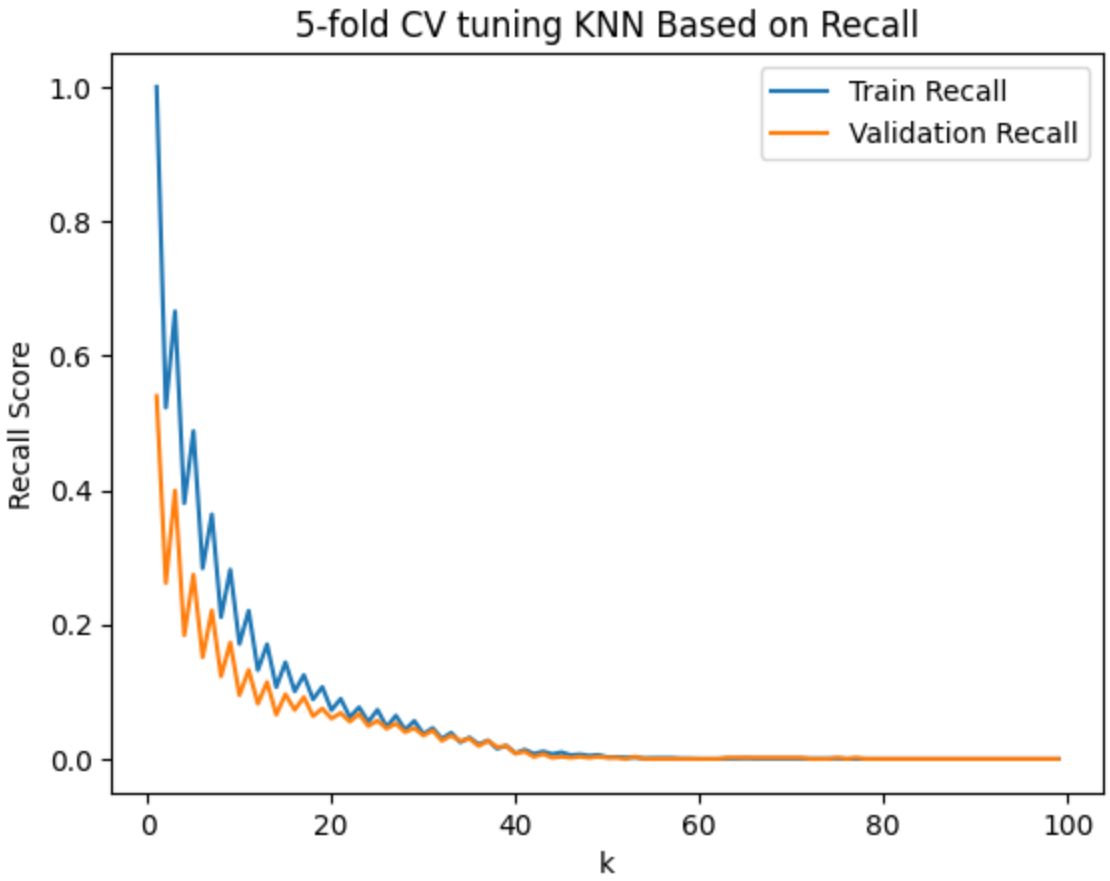
```
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-81-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
```

In [82]:

```
crossval_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   k                  99 non-null     float64
 1   train_recall       99 non-null     float64
 2   validation_recall  99 non-null     float64
dtypes: float64(3)
memory usage: 2.4 KB
```

In [83]:
```
fig, ax = plt.subplots()
sns.lineplot(x = 'k', y = 'train_recall',
             data = crossval_df,
             ax = ax, label = 'Train Recall')
sns.lineplot(x = 'k', y = 'validation_recall',
             data = crossval_df,
             ax = ax, label = 'Validation Recall')
ax.set_ylabel('Recall Score')
ax.set_title('5-fold CV tuning KNN Based on Recall')
plt.show()
```



This looks bizzare. Will try again without shuffle.

In [111]:
```
y = pivoted_df['Surgery']
X = pivoted_df.drop('Surgery', axis=1)
```

In [112]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [95]:

```
def cross_validation(X_train, y_train, k, num_split = 5):

    X_train = X_train.values
    y_train = y_train.values

    recall_train_list = []
    recall_val_list = []

    for train_index, valid_index in KFold(n_splits = num_split).split(

        # train and validation splitting
        X_train_fold, X_val_fold = X_train[train_index], X_train[valid
        y_train_fold, y_val_fold = y_train[train_index], y_train[valid

        #create/fit the Standard scaler on the train fold
        scaler = StandardScaler()
        X_train_fold_scaled = scaler.fit_transform(X_train_fold)
        # transform validation fold
        X_val_fold_scaled = scaler.transform(X_val_fold)

        smote=SMOTE(random_state=42)
        X_train_fold_smote, y_train_fold_smote = smote.fit_resample(X_

        # create/fit knearest neighbor
        knn = KNeighborsClassifier(n_neighbors = k)
        knn.fit(X_train_fold_smote, y_train_fold_smote)

        # now how did we do?
        y_train_pred = knn.predict(X_train_fold_scaled)
        y_val_pred = knn.predict(X_val_fold_scaled)

        recall_train = recall_score(y_train_fold, y_train_pred)
        recall_val = recall_score(y_val_fold, y_val_pred)

        recall_train_list.append(recall_train)
        recall_val_list.append(recall_val)

    return {'k': k, 'train_recall': np.mean(recall_train_list), 'valid
```

In [96]:

```
# empty dataframe
crossval_df = pd.DataFrame(columns = ['k', 'train_recall', 'validation_
# append results for each value of k
for k in np.arange(1,100):
    crossval_df = crossval_df.append(cross_validation(X_train, y_train
```

```
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
```

```
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
```

```
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
```

```
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
```

```
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
```

```
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
```

```
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
```

```
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k, 5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
```

```
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
<ipython-input-96-9036620c5700>:5: FutureWarning: The frame.append method
is deprecated and will be removed from pandas in a future version. Use pa
ndas.concat instead.
  crossval_df = crossval_df.append(cross_validation(X_train, y_train, k,
5), ignore_index = True)
```

In [97]:

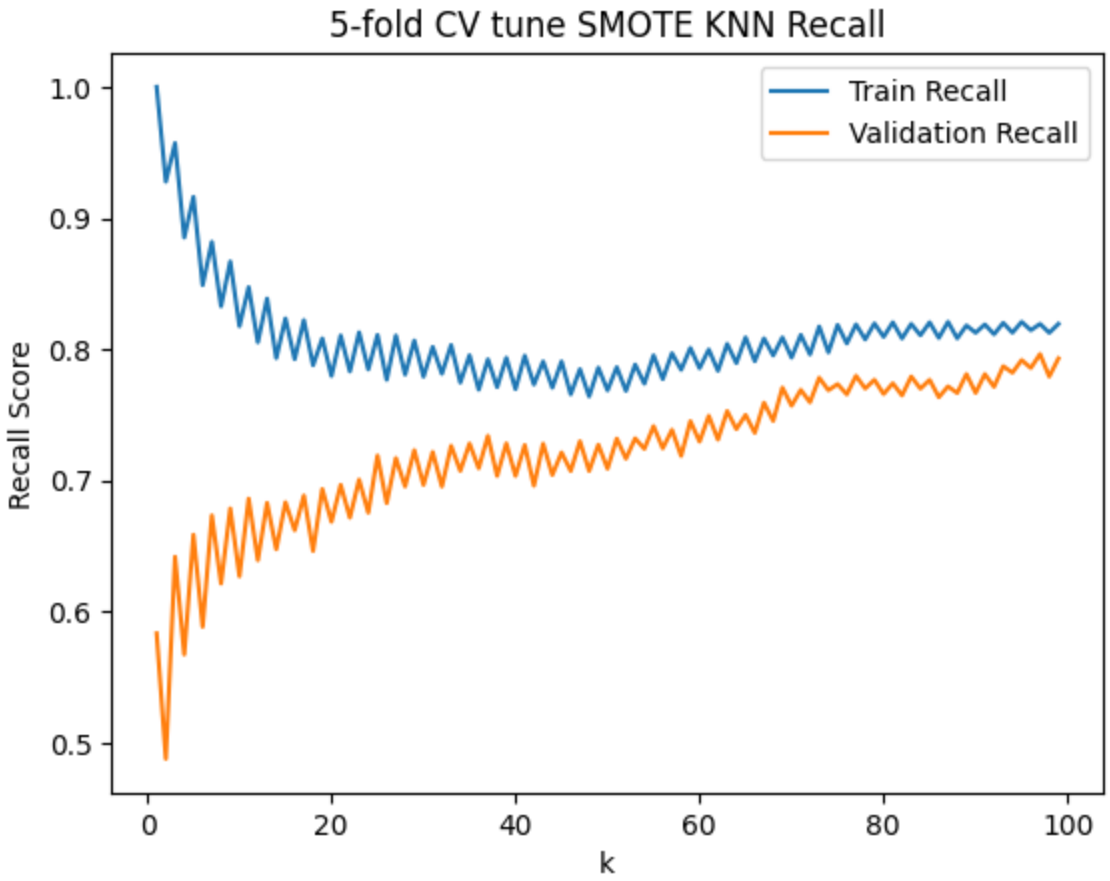
```python
results = []
for k in np.arange(1, 100):
    results.append(cross_validation(X_train, y_train, k, 5))

# Create DataFrame from results
crossval_df = pd.DataFrame(results)

print(crossval_df.head())
```

```
   k  train_recall  validation_recall
0  1      1.000000           0.583578
1  2      0.927756           0.487959
2  3      0.957156           0.641969
3  4      0.885197           0.567472
4  5      0.916181           0.658766
```

In [98]:
```
1 crossval_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   k                  99 non-null     int64
 1   train_recall       99 non-null     float64
 2   validation_recall  99 non-null     float64
dtypes: float64(2), int64(1)
memory usage: 2.4 KB
```

In [100]:
```
1  fig, ax = plt.subplots()
2  sns.lineplot(x = 'k', y = 'train_recall',
3               data = crossval_df,
4               ax = ax, label = 'Train Recall')
5  sns.lineplot(x = 'k', y = 'validation_recall',
6               data = crossval_df,
7               ax = ax, label = 'Validation Recall')
8  ax.set_ylabel('Recall Score')
9  ax.set_title('5-fold CV tune SMOTE KNN Recall')
10 plt.show()
```



In [102]:
```
crossval_df.iloc[crossval_df['validation_recall'].idxmax()]
```

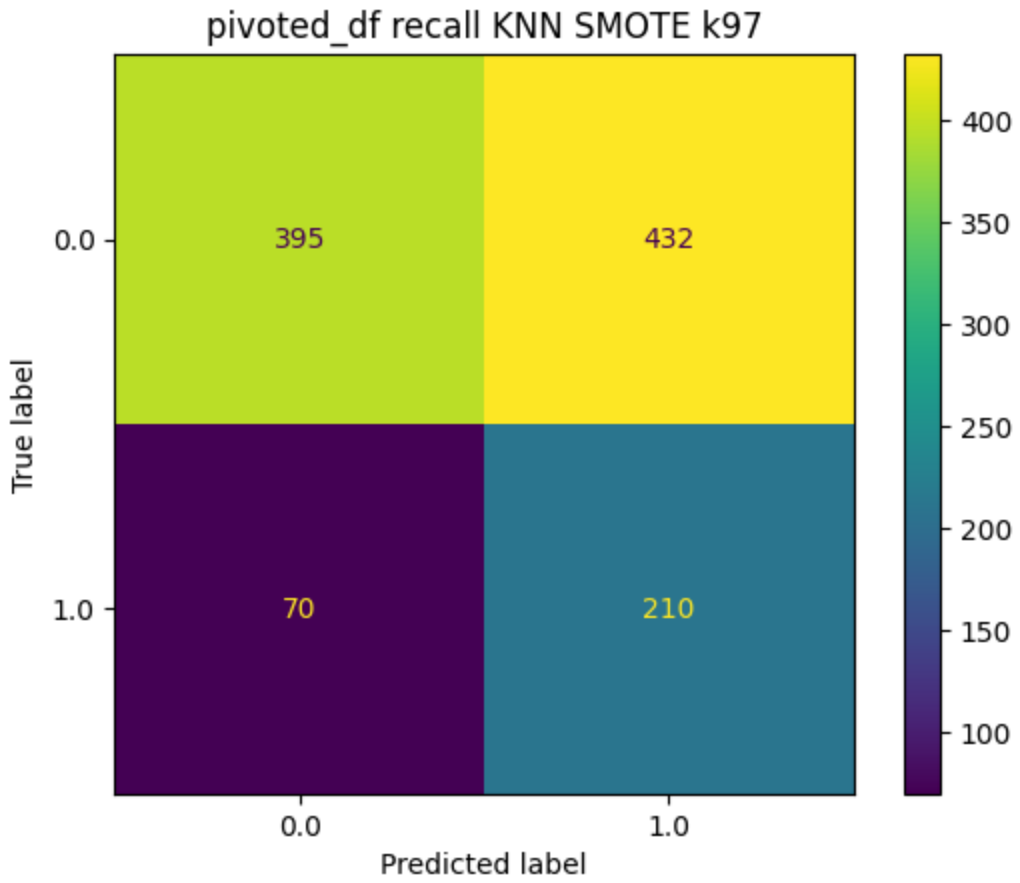
Out[102]:
```
k                   97.000000
train_recall         0.819350
validation_recall    0.796382
Name: 96, dtype: float64
```

In [107]:
```
fulltrain_scaler = StandardScaler()
X_train_sc = fulltrain_scaler.fit_transform(X_train)

smote=SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_sc, y_train)

best_estimator = KNeighborsClassifier(n_neighbors = 97)
best_estimator.fit(X_train_smote, y_train_smote)

X_test_sc = fulltrain_scaler.transform(X_test)

# get predictions
y_pred = best_estimator.predict(X_test_sc)
test_recall = recall_score(y_test, y_pred)
print(f'Test Recall: {test_recall}')
```

```
Test Recall: 0.75
```

In [108]:

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('pivoted_df recall KNN SMOTE k97')
plt.show()
print(classification_report(y_test, pred))
```



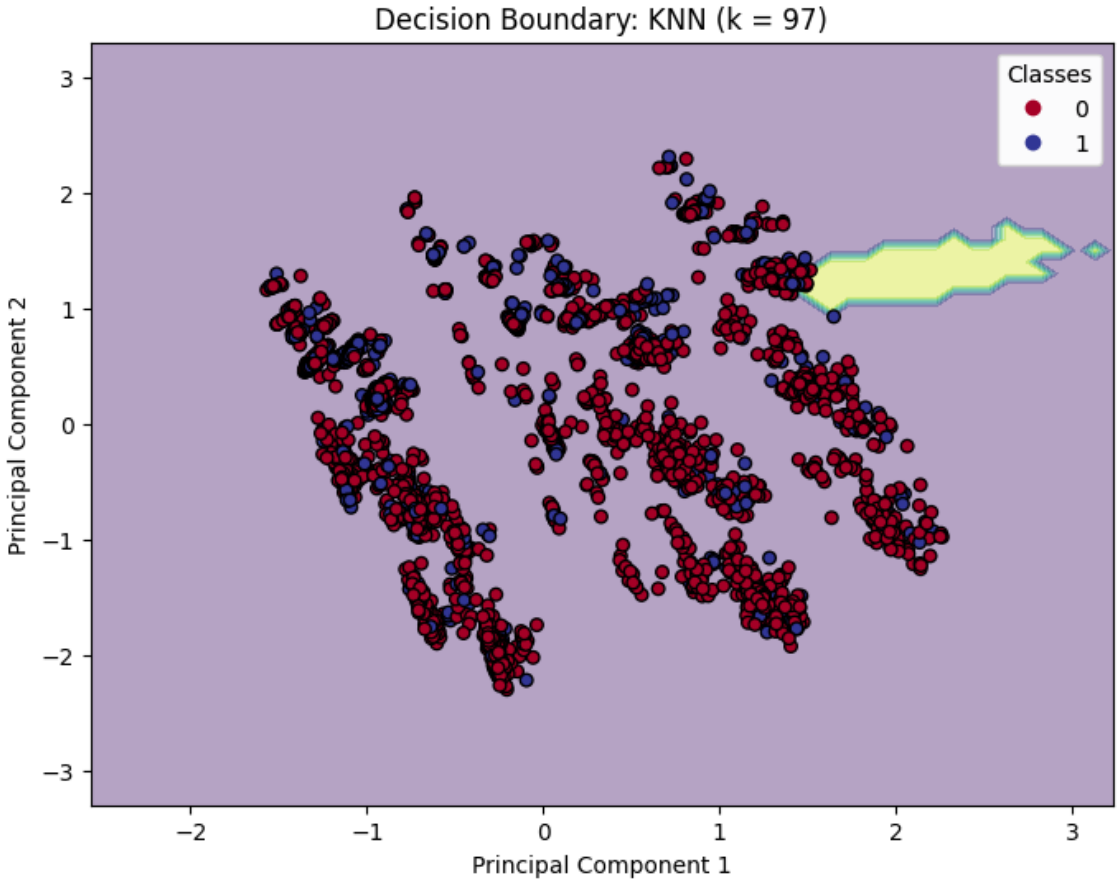
	precision	recall	f1-score	support
0.0	0.84	0.67	0.75	827
1.0	0.39	0.61	0.47	280
accuracy			0.66	1107
macro avg	0.61	0.64	0.61	1107
weighted avg	0.72	0.66	0.68	1107

Want to pull more info from this. Cluster, analyze, label (power pitcher, etc.)? Hyperparameter tune?

To visualize the decision boundary, need to apply PCA (too many features)

In [109]:

```
# Reduce the data to two components for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Scale the PCA-transformed data
fulltrain_scaler = StandardScaler()
X_full_sc = fulltrain_scaler.fit_transform(X_pca)

# Define the meshgrid for the contour plot based on the scaled data
x_min, x_max = X_full_sc[:, 0].min() - 1, X_full_sc[:, 0].max() + 1
y_min, y_max = X_full_sc[:, 1].min() - 1, X_full_sc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

# Train the best estimator on the PCA-transformed and scaled data
best_estimator = KNeighborsClassifier(n_neighbors=97)
best_estimator.fit(X_full_sc, y)  # Make sure y is your target array

# Perform the prediction on the meshgrid
Z = best_estimator.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundary
f, ax = plt.subplots(figsize=(8, 6))
ax.contourf(xx, yy, Z, alpha=0.4)

# Scatter plot of the actual data points
scatter = ax.scatter(X_full_sc[:, 0], X_full_sc[:, 1], c=y, s=30, edge

# Create a legend
legend1 = ax.legend(*scatter.legend_elements(), loc="upper right", tit
ax.add_artist(legend1)

# Label the axes and title
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_title('Decision Boundary: KNN (k = 97)')

plt.show()

```



Don't think this makes sense. Will retry things with a smaller k value

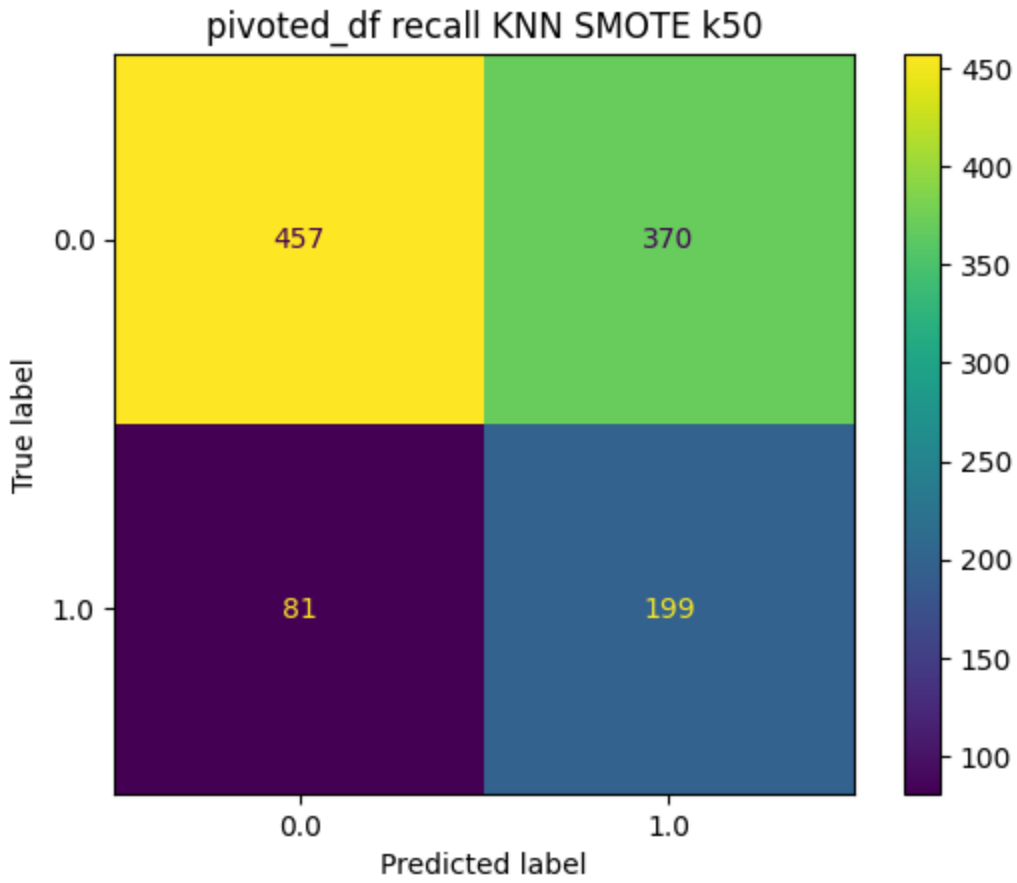
In [113]:

```
fulltrain_scaler = StandardScaler()
X_train_sc = fulltrain_scaler.fit_transform(X_train)

smote=SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_sc, y_train)

best_estimator = KNeighborsClassifier(n_neighbors = 50)
best_estimator.fit(X_train_smote, y_train_smote)

X_test_sc = fulltrain_scaler.transform(X_test)

# get predictions
y_pred = best_estimator.predict(X_test_sc)
test_recall = recall_score(y_test, y_pred)
print(f'Test Recall: {test_recall}')
```

```
Test Recall: 0.7107142857142857
```

In [114]:

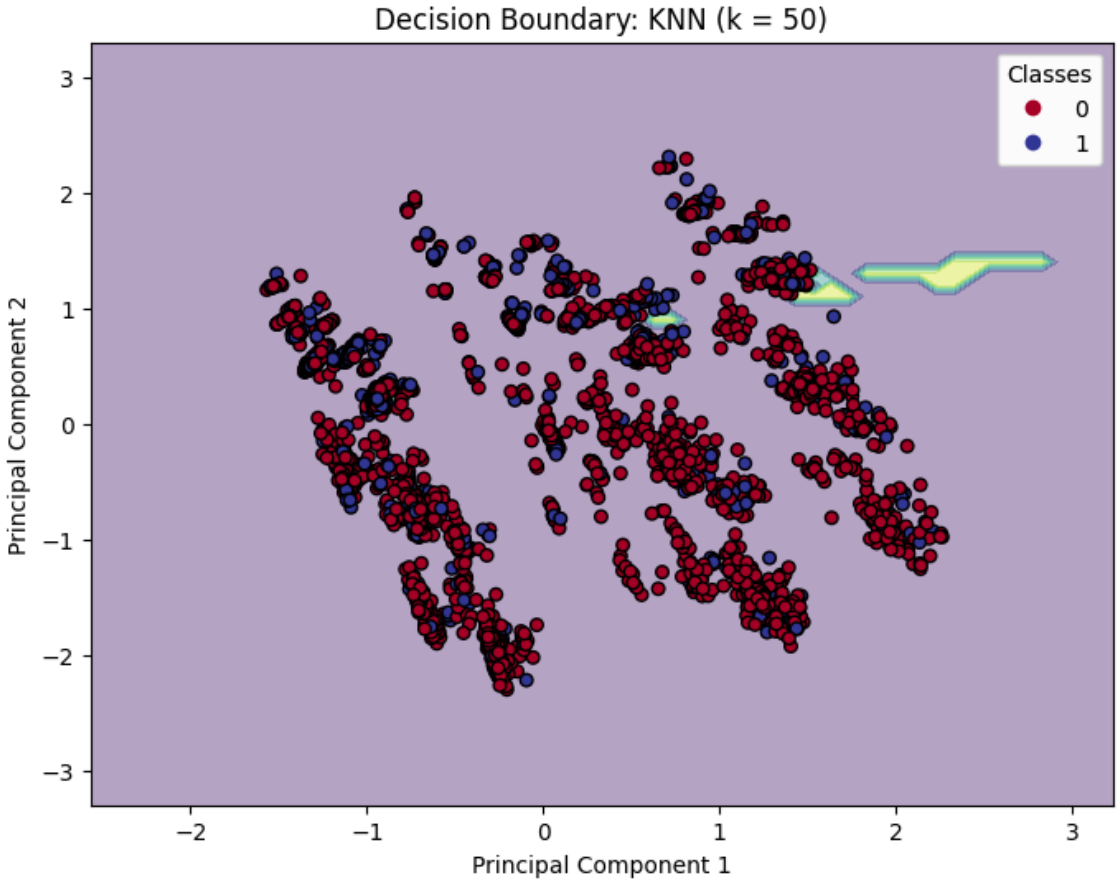
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('pivoted_df recall KNN SMOTE k50')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.84	0.67	0.75	827
1.0	0.39	0.61	0.47	280
accuracy			0.66	1107
macro avg	0.61	0.64	0.61	1107
weighted avg	0.72	0.66	0.68	1107

In [116]:

```
# Reduce the data to two components for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Scale the PCA-transformed data
fulltrain_scaler = StandardScaler()
X_full_sc = fulltrain_scaler.fit_transform(X_pca)

# Define the meshgrid for the contour plot based on the scaled data
x_min, x_max = X_full_sc[:, 0].min() - 1, X_full_sc[:, 0].max() + 1
y_min, y_max = X_full_sc[:, 1].min() - 1, X_full_sc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

# Train the best estimator on the PCA-transformed and scaled data
best_estimator = KNeighborsClassifier(n_neighbors=50)
best_estimator.fit(X_full_sc, y)  # Make sure y is your target array

# Perform the prediction on the meshgrid
Z = best_estimator.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundary
f, ax = plt.subplots(figsize=(8, 6))
ax.contourf(xx, yy, Z, alpha=0.4)

# Scatter plot of the actual data points
scatter = ax.scatter(X_full_sc[:, 0], X_full_sc[:, 1], c=y, s=30, edge

# Create a legend
legend1 = ax.legend(*scatter.legend_elements(), loc="upper right", tit
ax.add_artist(legend1)

# Label the axes and title
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_title('Decision Boundary: KNN (k = 50)')

plt.show()
```



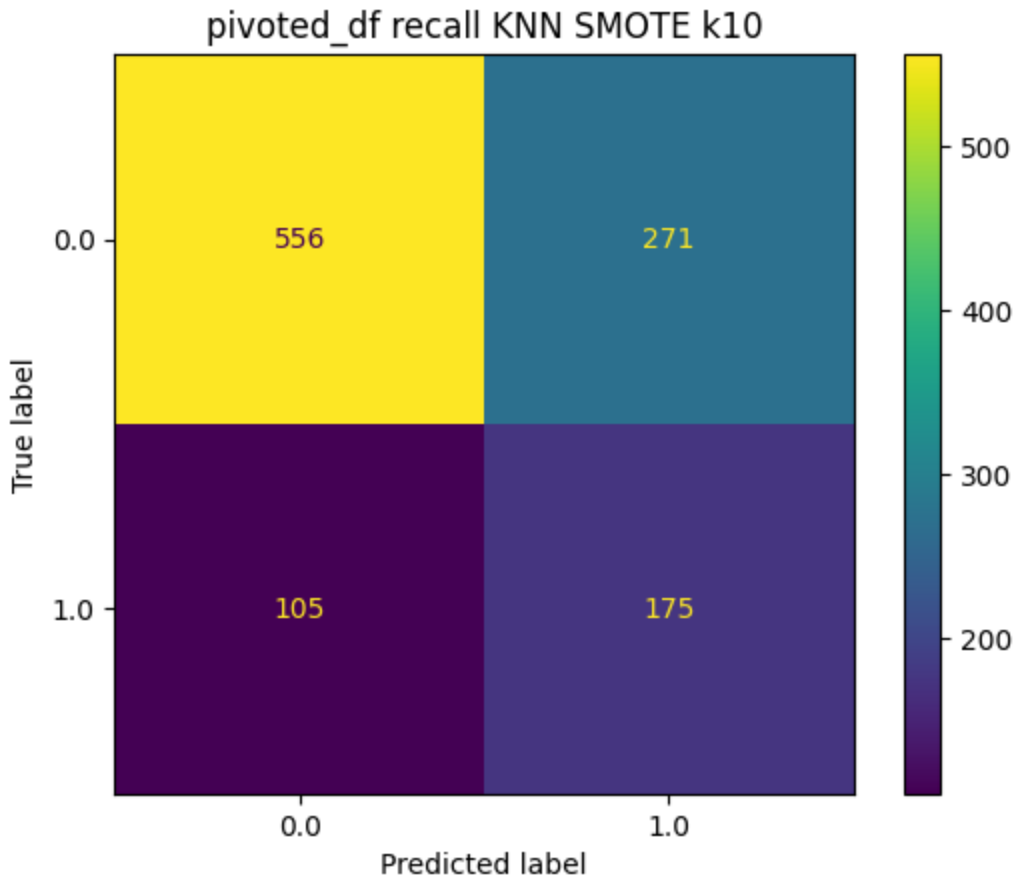
In [117]:

```
fulltrain_scaler = StandardScaler()
X_train_sc = fulltrain_scaler.fit_transform(X_train)

smote=SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_sc, y_train)

best_estimator = KNeighborsClassifier(n_neighbors = 10)
best_estimator.fit(X_train_smote, y_train_smote)

X_test_sc = fulltrain_scaler.transform(X_test)

# get predictions
y_pred = best_estimator.predict(X_test_sc)
test_recall = recall_score(y_test, y_pred)
print(f'Test Recall: {test_recall}')
```

```
Test Recall: 0.625
```

In [119]:

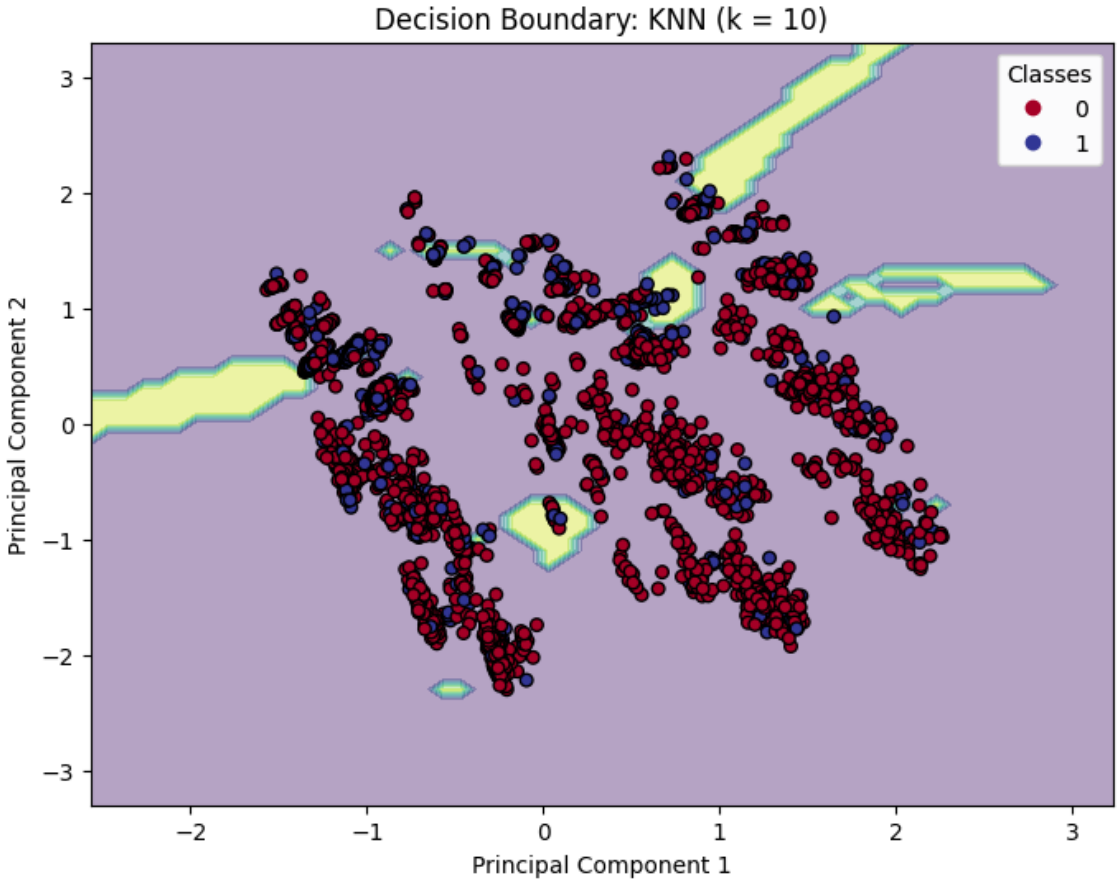
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.title('pivoted_df recall KNN SMOTE k10')
plt.show()
print(classification_report(y_test, pred))
```



	precision	recall	f1-score	support
0.0	0.84	0.67	0.75	827
1.0	0.39	0.61	0.47	280
accuracy			0.66	1107
macro avg	0.61	0.64	0.61	1107
weighted avg	0.72	0.66	0.68	1107

In [120]:

```
# Reduce the data to two components for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Scale the PCA-transformed data
fulltrain_scaler = StandardScaler()
X_full_sc = fulltrain_scaler.fit_transform(X_pca)

# Define the meshgrid for the contour plot based on the scaled data
x_min, x_max = X_full_sc[:, 0].min() - 1, X_full_sc[:, 0].max() + 1
y_min, y_max = X_full_sc[:, 1].min() - 1, X_full_sc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

# Train the best estimator on the PCA-transformed and scaled data
best_estimator = KNeighborsClassifier(n_neighbors=10)
best_estimator.fit(X_full_sc, y)  # Make sure y is your target array

# Perform the prediction on the meshgrid
Z = best_estimator.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundary
f, ax = plt.subplots(figsize=(8, 6))
ax.contourf(xx, yy, Z, alpha=0.4)

# Scatter plot of the actual data points
scatter = ax.scatter(X_full_sc[:, 0], X_full_sc[:, 1], c=y, s=30, edge

# Create a legend
legend1 = ax.legend(*scatter.legend_elements(), loc="upper right", tit
ax.add_artist(legend1)

# Label the axes and title
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_title('Decision Boundary: KNN (k = 10)')

plt.show()
```



This isn't working. KMeans and KNN may not be applicable to my dataset. Not enough data to differentiate groups of pitchers. Also dataset is very imbalanced, SMOTE doesn't work well for this.

In [ ]:

```
1
```