Code:

```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
import pandas as pd
import itertools
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
from sklearn.model_selection import LeaveOneOut
from sklearn import model_selection
loocv = model_selection.LeaveOneOut()
kf = KFold(n_splits=5,random_state=42, shuffle=True)  #divide n_splits data cross validation
#kf = KFold(loocv,random_state=42, shuffle=True) #if shuffle not true index increases sequentially
svc = SVC(gamma='scale')
scores = list()  #models score append list
TP=0 #total true positive
TN=0 #total true negative
FP=0 #total false positive
FN=0 #total false negative

y=np.array(y).astype(int)
for train_index,test_index in kf.split(X):
    X_train = X.iloc[train_index]
    X_test = X.iloc[test_index]
    #y_train = y.iloc[train_index]
    #y_test = y.iloc[test_index]
    y_train = y[train_index]
    y_test = y[test_index]
    svc.fit(X_train, y_train)  #train model SVM
    y_pred = svc.predict(X_test)  #test model with divided test set
    scores.append(svc.fit(X_train, y_train).score(X_test, y_test))
    cm = confusion_matrix(y_test, y_pred) #confusion matrix created
    #print(cm)
    TN=TN+cm[0,0]
    TP=TP+cm[1,1]
    FP=FP+cm[0,1]
    FN=FN+cm[1,0]
    print(TN)
    print(FP)
    print(FN)
    print(TP)
```

K=5;
Confusion Matrix- Python Results

29	13
12	30

accuracy: 0,70238095
sensitivity: 0,714288571
specificity: 0,69047619

Confusion Matrix- Matlab Results

25	17
14	28

accuracy: 0,63095238
sensitivity: 0,6666666667
specificity: 0,5952381

K=10;
Confusion Matrix- Python Results

30	12
12	30

accuracy: 0,71428571
sensitivity: 0,714288571
specificity: 0,714288571

Confusion Matrix- Matlab Results

29	13
12	30

accuracy: 0,70238095
sensitivity: 0,714288571
specificity: 0,69047619

Make imbalanced some data dropped

```
inbalancedX = X_im.drop([X_im.index[83] , X_im.index[82], X_im.index[81], X_im.index[80]])
inbalancedX=inbalancedX.drop([inbalancedX.index[79] , inbalancedX.index[78], inbalancedX.index[77],
inbalancedX.index[76]])
inbalancedX=inbalancedX.drop([inbalancedX.index[75] , inbalancedX.index[74], inbalancedX.index[73],
inbalancedX.index[72], inbalancedX.index[71], inbalancedX.index[70]])
inbalancedX=inbalancedX.drop([inbalancedX.index[69] , inbalancedX.index[68], inbalancedX.index[67],
inbalancedX.index[66], inbalancedX.index[65], inbalancedX.index[64]])
inbalancedX=inbalancedX.drop([inbalancedX.index[63] , inbalancedX.index[62], inbalancedX.index[61],
inbalancedX.index[60], inbalancedX.index[59], inbalancedX.index[58]])
inbalancedX=inbalancedX.drop([inbalancedX.index[57] , inbalancedX.index[56], inbalancedX.index[55],
inbalancedX.index[54], inbalancedX.index[53], inbalancedX.index[52]])
inbalancedX=inbalancedX.drop([inbalancedX.index[51] , inbalancedX.index[50]])
```