# ****DIY Project-2****

# ****Kanada Handwritten digit Classification****

## ****Loading Dataset****

from google.colab import drive  
drive.mount('/content/drive')

Mounted at /content/drive

cd drive/MyDrive/data

/content/drive/MyDrive/data

ls

athlete\_events.csv lda\_10\_iter.pkl titanic\_train.csv  
'BBC News Train.csv' ML\_Project\_1.zip vgsales.csv  
 Dig-MNIST.csv MNIST/ wineQualityReds.csv  
 groceries.csv pokemon.csv wineQualityWhites.csv  
'groceries - groceries.csv' project1.csv  
 housing.csv Sentiment/

%matplotlib inline  
import numpy as np  
import pandas as pd  
import sklearn as sk  
import matplotlib.pyplot as plt

mnist=pd.read\_csv('Dig-MNIST.csv')  
mnist.head()

label pixel0 pixel1 pixel2 ... pixel780 pixel781 pixel782 pixel783  
0 0 0 0 0 ... 0 0 0 0  
1 1 0 0 0 ... 0 0 0 0  
2 2 0 0 0 ... 0 0 0 0  
3 3 0 0 0 ... 0 0 0 0  
4 4 0 0 0 ... 0 0 0 0  
  
[5 rows x 785 columns]

y=mnist['label']  
y.shape

(10240,)

X=mnist.iloc[:,1:]  
X.shape

(10240, 784)

X.head()

pixel0 pixel1 pixel2 pixel3 ... pixel780 pixel781 pixel782 pixel783  
0 0 0 0 0 ... 0 0 0 0  
1 0 0 0 0 ... 0 0 0 0  
2 0 0 0 0 ... 0 0 0 0  
3 0 0 0 0 ... 0 0 0 0  
4 0 0 0 0 ... 0 0 0 0  
  
[5 rows x 784 columns]

## ****Training Models****

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,stratify=y,shuffle=True,random\_state=144,test\_size=0.2)

X\_train.shape

(8192, 784)

X\_test.shape

(2048, 784)

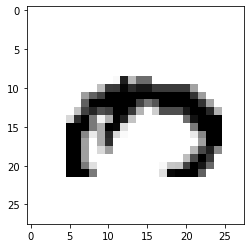
y\_train.shape

(8192,)

### ****Sample Images with Labels****

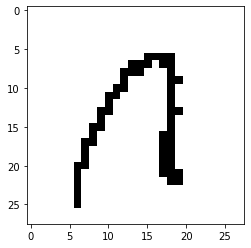
import matplotlib  
import matplotlib.pyplot as plt  
some\_digit=np.array(X.iloc[5000])  
some\_digit\_image=some\_digit.reshape(28,28)  
plt.imshow(some\_digit\_image,cmap=matplotlib.cm.binary)  
print("Label : ",y.iloc[5000])

Label : 0



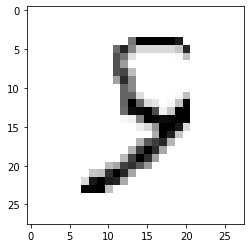
import matplotlib  
import matplotlib.pyplot as plt  
some\_digit=np.array(X.iloc[5001])  
some\_digit\_image=some\_digit.reshape(28,28)  
plt.imshow(some\_digit\_image,cmap=matplotlib.cm.binary)  
print("Label : ",y.iloc[5001])

Label : 1



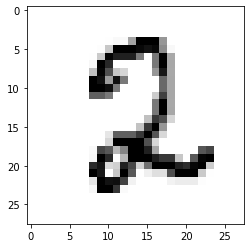
import matplotlib  
import matplotlib.pyplot as plt  
some\_digit=np.array(X.iloc[5002])  
some\_digit\_image=some\_digit.reshape(28,28)  
plt.imshow(some\_digit\_image,cmap=matplotlib.cm.binary)  
print("Label : ",y.iloc[5002])

Label : 2



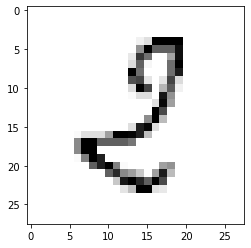
import matplotlib  
import matplotlib.pyplot as plt  
some\_digit=np.array(X.iloc[5003])  
some\_digit\_image=some\_digit.reshape(28,28)  
plt.imshow(some\_digit\_image,cmap=matplotlib.cm.binary)  
print("Label : ",y.iloc[5003])

Label : 3



import matplotlib  
import matplotlib.pyplot as plt  
some\_digit=np.array(X.iloc[5007])  
some\_digit\_image=some\_digit.reshape(28,28)  
plt.imshow(some\_digit\_image,cmap=matplotlib.cm.binary)  
print("Label : ",y.iloc[5007])

Label : 7



from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n\_jobs=1,n\_estimators=10)  
rf.fit(X\_train,y\_train)

RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None,

criterion='gini', max\_depth=None, max\_features='auto',

max\_leaf\_nodes=None, max\_samples=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=10, n\_jobs=1,

oob\_score=False, random\_state=None, verbose=0,

warm\_start=False)

rf.score(X\_test,y\_test)

0.8427734375

pred=rf.predict(X\_test)

from sklearn.metrics import confusion\_matrix  
confusion=confusion\_matrix(y\_test,pred)  
print(confusion)

[[178 13 4 1 0 0 2 0 2 5]

[ 13 176 3 4 0 3 2 1 1 1]

[ 12 5 175 6 1 0 3 2 0 1]

[ 3 6 6 173 3 2 5 6 1 0]

[ 3 4 2 1 176 9 2 1 3 4]

[ 2 4 3 8 9 170 2 1 5 0]

[ 4 0 7 7 5 2 157 20 0 3]

[ 1 3 7 8 1 1 15 166 0 3]

[ 5 3 4 1 3 2 1 2 178 6]

[ 4 0 0 0 3 0 10 4 7 177]]

from sklearn import svm  
svm\_model=svm.LinearSVC(C=100)  
#change C=10 and C=100 to see different results  
svm\_model.fit(X\_train,y\_train)

LinearSVC(C=100, class\_weight=None, dual=True, fit\_intercept=True,

intercept\_scaling=1, loss='squared\_hinge', max\_iter=1000,

multi\_class='ovr', penalty='l2', random\_state=None, tol=0.0001,

verbose=0)

svm\_model.score(X\_train,y\_train)

0.8912353515625

svm\_model.score(X\_test,y\_test)

0.71533203125

pred=svm\_model.predict(X\_test)

from sklearn.metrics import confusion\_matrix  
confusion=confusion\_matrix(y\_test,pred)  
print(confusion)

[[163 14 11 3 1 1 1 2 7 2]

[ 14 147 7 1 4 11 2 2 13 3]

[ 6 7 164 4 7 3 4 1 5 4]

[ 15 6 11 146 4 3 5 8 4 3]

[ 6 2 9 6 140 14 4 1 10 13]

[ 6 9 7 4 7 150 0 1 14 6]

[ 11 3 7 13 8 6 116 24 8 9]

[ 9 11 4 15 3 3 18 133 1 8]

[ 11 2 4 0 7 4 1 2 161 13]

[ 13 3 5 3 10 4 5 6 11 145]]

from sklearn.preprocessing import StandardScaler  
scaler=StandardScaler().fit(X\_train)  
X\_train\_scaled=scaler.transform(X\_train)  
X\_test\_scaled=scaler.transform(X\_test)

from sklearn import svm  
svm\_model=svm.LinearSVC(C=10)  
#change C=10 and C=100 to see different results  
svm\_model.fit(X\_train\_scaled,y\_train)

LinearSVC(C=10, class\_weight=None, dual=True, fit\_intercept=True,

intercept\_scaling=1, loss='squared\_hinge', max\_iter=1000,

multi\_class='ovr', penalty='l2', random\_state=None, tol=0.0001,

verbose=0)

svm\_model.score(X\_train\_scaled,y\_train)

0.913330078125

svm\_model.score(X\_test\_scaled,y\_test)

0.724609375

pred=svm\_model.predict(X\_test\_scaled)

from sklearn.metrics import confusion\_matrix  
confusion=confusion\_matrix(y\_test,pred)  
print(confusion)

[[159 12 10 3 0 0 4 1 10 6]

[ 15 139 6 2 4 8 3 7 16 4]

[ 6 9 161 4 6 2 5 4 5 3]

[ 5 4 8 144 6 5 19 9 4 1]

[ 3 2 3 6 141 18 7 2 11 12]

[ 5 2 3 4 4 170 3 1 10 2]

[ 1 3 7 13 12 8 120 30 4 7]

[ 3 14 4 6 6 4 16 145 1 6]

[ 6 6 2 2 6 4 3 3 155 18]

[ 4 3 2 2 16 2 8 7 11 150]]

from sklearn.neural\_network import MLPClassifier  
model=MLPClassifier()  
model

MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,

beta\_2=0.999, early\_stopping=False, epsilon=1e-08,

hidden\_layer\_sizes=(100,), learning\_rate='constant',

learning\_rate\_init=0.001, max\_fun=15000, max\_iter=200,

momentum=0.9, n\_iter\_no\_change=10, nesterovs\_momentum=True,

power\_t=0.5, random\_state=None, shuffle=True, solver='adam',

tol=0.0001, validation\_fraction=0.1, verbose=False,

warm\_start=False)

model.fit(X\_train,y\_train)

model.score(X\_test,y\_test)

0.8427734375

pred=model.predict(X\_test)

from sklearn.metrics import confusion\_matrix  
confusion=confusion\_matrix(y\_test,pred)  
print(confusion)

[[177 8 5 1 1 1 5 1 5 1]

[ 8 177 6 3 1 2 0 2 3 2]

[ 7 4 176 4 0 3 4 3 2 2]

[ 2 5 3 173 1 7 8 5 1 0]

[ 0 2 1 2 169 15 7 0 6 3]

[ 1 2 1 7 10 181 0 1 1 0]

[ 4 2 3 7 5 2 159 16 0 7]

[ 2 4 4 2 0 1 24 160 1 7]

[ 4 2 5 3 2 3 2 0 174 10]

[ 4 0 1 3 1 0 8 3 5 180]]

model=MLPClassifier(hidden\_layer\_sizes=(200,200,200,200,200,200))

model.fit(X\_train,y\_train)

MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,

beta\_2=0.999, early\_stopping=False, epsilon=1e-08,

hidden\_layer\_sizes=(200, 200, 200, 200, 200, 200),

learning\_rate='constant', learning\_rate\_init=0.001, max\_fun=15000,

max\_iter=200, momentum=0.9, n\_iter\_no\_change=10,

nesterovs\_momentum=True, power\_t=0.5, random\_state=None,

shuffle=True, solver='adam', tol=0.0001, validation\_fraction=0.1,

verbose=False, warm\_start=False)

model.score(X\_test,y\_test)

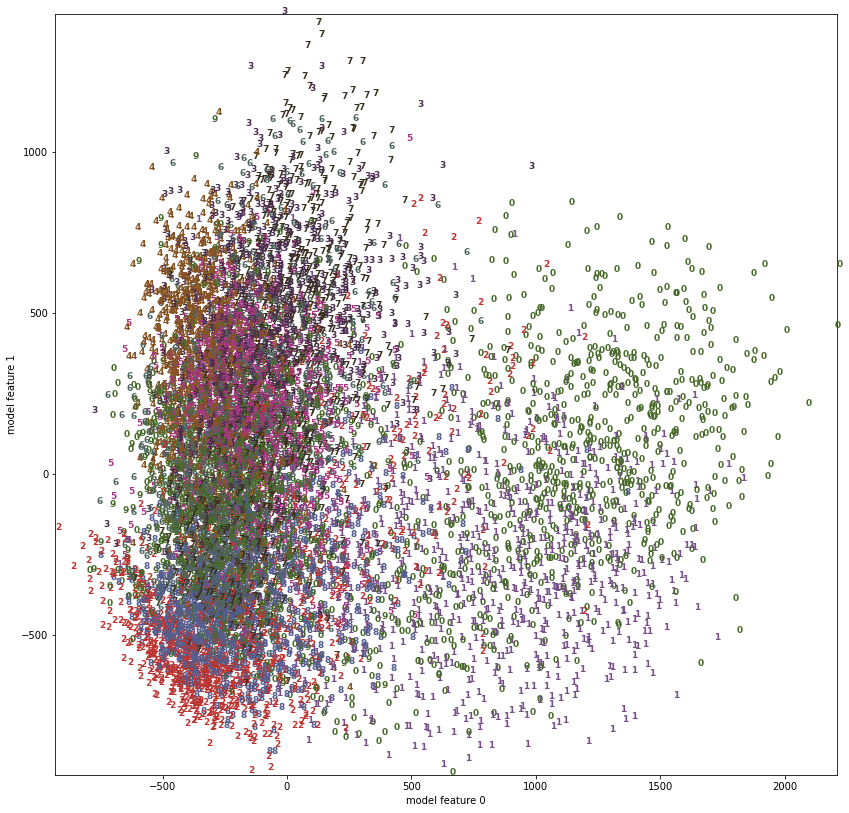
0.84814453125

### ****Dimensionality Reduction****

from sklearn.decomposition import PCA  
from sklearn.manifold import TSNE  
from sklearn.manifold import Isomap

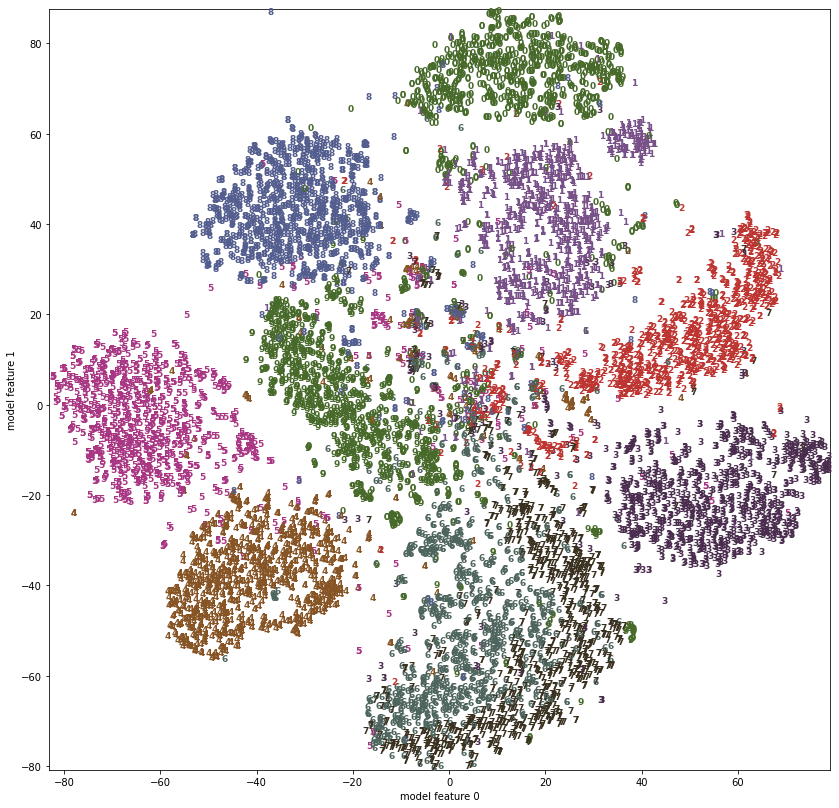
def display\_2d\_component\_names(model,selected,dataobj):  
 colors=["#476A2A",'#785188','#BD3430','#4A2D4E','#875525','#A83683','#4E655E',  
 '#3A3120','#535D8E']  
 plt.figure(figsize=(14,14))  
 plt.xlim(model[:,0].min(),model[:,0].max()+1)  
 plt.ylim(model[:,1].min(),model[:,1].max()+1)  
 for i in range(len(dataobj)):  
 cindex=dataobj[i]%len(selected)  
 if dataobj[i] not in selected:  
 continue   
 plt.text(model[i,0],model[i,1],str(dataobj[i]),  
 color=colors[cindex%9],fontdict={'weight':'bold','size':9})  
 plt.xlabel("model feature 0")  
 plt.ylabel("model feature 1")

pca=PCA(n\_components=2)  
pca.fit(X)  
digits\_pca=pca.transform(X)  
display\_2d\_component\_names(digits\_pca,(0,1,2,3,4,5,6,7,8,9),y)



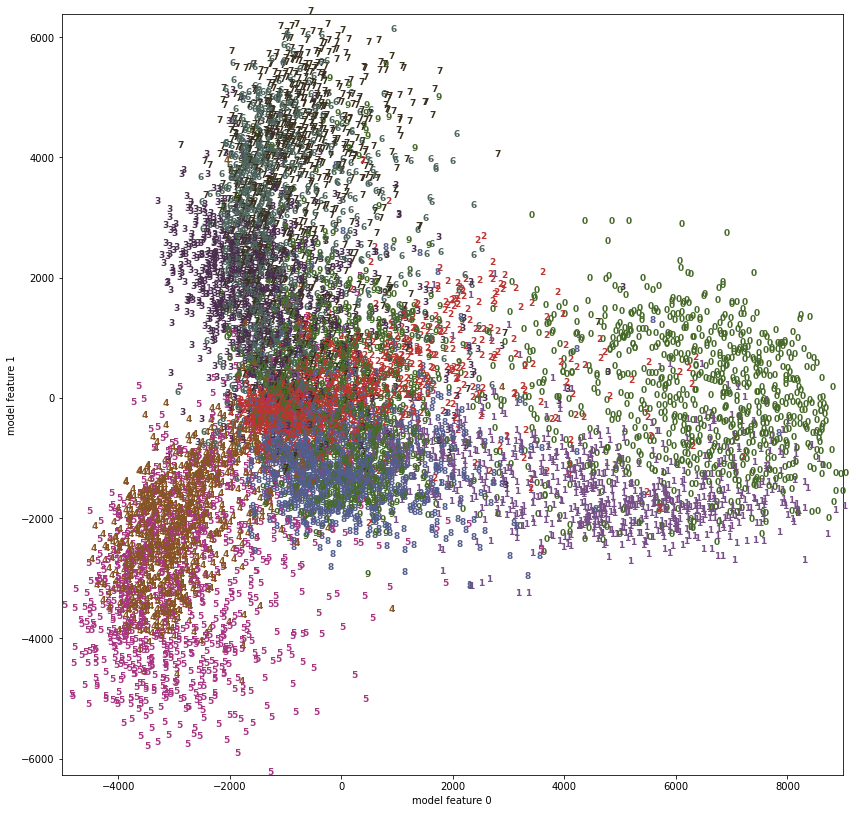
tsne=TSNE(random\_state=42)  
digits\_tsne=tsne.fit\_transform(X)

display\_2d\_component\_names(digits\_tsne,(0,1,2,3,4,5,6,7,8,9),y)



iso=Isomap(n\_neighbors=10,n\_components=2)  
digits\_iso=iso.fit(X).transform(X)

display\_2d\_component\_names(digits\_iso,(0,1,2,3,4,5,6,7,8,9),y)



Out of three (PCA,Isomap,tsne),tsne seems to be best in isolating groups

### ****Support Vector Machine****[¶](https://render.githubusercontent.com/view/ipynb?color_mode=auto&commit=1d83a40b8293b2a78ac362682fe542724c252490&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f6d6f6b6b616c616d6f756e696b612f4d4c50726f6a656374322f316438336134306238323933623261373861633336323638326665353432373234633235323439302f4d4c50726f6a656374322532302831292e6970796e62&nwo=mokkalamounika%2FMLProject2&path=MLProject2+%281%29.ipynb&repository_id=369213240&repository_type=Repository#Support-Vector-Machine)

**Without scaling**

X\_train,X\_test,y\_train,y\_test=train\_test\_split(digits\_tsne,y,stratify=y,shuffle=True,random\_state=144)

from sklearn import svm  
svm\_model=svm.LinearSVC(C=100)  
#change C=10 and C=100 to see different results  
svm\_model.fit(X\_train,y\_train)

/usr/local/lib/python3.7/dist-packages/sklearn/svm/\_base.py:947: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.  
 "the number of iterations.", ConvergenceWarning)

LinearSVC(C=100, class\_weight=None, dual=True, fit\_intercept=True,  
 intercept\_scaling=1, loss='squared\_hinge', max\_iter=1000,  
 multi\_class='ovr', penalty='l2', random\_state=None, tol=0.0001,  
 verbose=0)

svm\_model.score(X\_train,y\_train)

0.22630208333333332

svm\_model.score(X\_test,y\_test)

0.227734375

**With scaling**

from sklearn.preprocessing import StandardScaler  
scaler=StandardScaler().fit(X\_train)  
X\_train\_scaled=scaler.transform(X\_train)  
X\_test\_scaled=scaler.transform(X\_test)

from sklearn import svm  
svm\_model=svm.LinearSVC(C=10)  
#change C=10 and C=100 to see different results  
svm\_model.fit(X\_train\_scaled,y\_train)

/usr/local/lib/python3.7/dist-packages/sklearn/svm/\_base.py:947: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.  
 "the number of iterations.", ConvergenceWarning)

LinearSVC(C=10, class\_weight=None, dual=True, fit\_intercept=True,  
 intercept\_scaling=1, loss='squared\_hinge', max\_iter=1000,  
 multi\_class='ovr', penalty='l2', random\_state=None, tol=0.0001,  
 verbose=0)

svm\_model.score(X\_train\_scaled,y\_train)

0.6908854166666667

svm\_model.score(X\_test\_scaled,y\_test)

0.6875

### ****Neural Networks****

**without scaling**

from sklearn.neural\_network import MLPClassifier

model=MLPClassifier(hidden\_layer\_sizes=(200,200,200,200,200,200))

model.fit(X\_train,y\_train)

/usr/local/lib/python3.7/dist-packages/sklearn/neural\_network/\_multilayer\_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.  
 % self.max\_iter, ConvergenceWarning)

MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,  
 beta\_2=0.999, early\_stopping=False, epsilon=1e-08,  
 hidden\_layer\_sizes=(200, 200, 200, 200, 200, 200),  
 learning\_rate='constant', learning\_rate\_init=0.001, max\_fun=15000,  
 max\_iter=200, momentum=0.9, n\_iter\_no\_change=10,  
 nesterovs\_momentum=True, power\_t=0.5, random\_state=None,  
 shuffle=True, solver='adam', tol=0.0001, validation\_fraction=0.1,  
 verbose=False, warm\_start=False)

model.score(X\_test,y\_test)

0.883984375

**with scaling**

model=MLPClassifier(hidden\_layer\_sizes=(200,200,200,200,200,200))

model.fit(X\_train\_scaled,y\_train)

/usr/local/lib/python3.7/dist-packages/sklearn/neural\_network/\_multilayer\_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.  
 % self.max\_iter, ConvergenceWarning)

MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,  
 beta\_2=0.999, early\_stopping=False, epsilon=1e-08,  
 hidden\_layer\_sizes=(200, 200, 200, 200, 200, 200),  
 learning\_rate='constant', learning\_rate\_init=0.001, max\_fun=15000,  
 max\_iter=200, momentum=0.9, n\_iter\_no\_change=10,  
 nesterovs\_momentum=True, power\_t=0.5, random\_state=None,  
 shuffle=True, solver='adam', tol=0.0001, validation\_fraction=0.1,  
 verbose=False, warm\_start=False)

model.score(X\_test\_scaled,y\_test)

0.879296875

model.score(X\_train\_scaled,y\_train)

0.9092447916666667

### ****Fine Tuning Parameters for SVM****

param\_grid = [{'kernel': ['rbf'],  
 'C': [0.001, 0.01, 0.1, 1, 10, 100],  
 'gamma': [0.001, 0.01, 0.1, 1, 10, 100]},  
 {'kernel': ['linear'],  
 'C': [0.001, 0.01, 0.1, 1, 10, 100]}]

from sklearn.model\_selection import GridSearchCV  
from sklearn.svm import SVC  
grid\_search = GridSearchCV(SVC(), param\_grid, cv=5)

grid\_search.fit(X\_train\_scaled, y\_train)

GridSearchCV(cv=5, error\_score=nan,  
 estimator=SVC(C=1.0, break\_ties=False, cache\_size=200,  
 class\_weight=None, coef0=0.0,  
 decision\_function\_shape='ovr', degree=3,  
 gamma='scale', kernel='rbf', max\_iter=-1,  
 probability=False, random\_state=None, shrinking=True,  
 tol=0.001, verbose=False),  
 iid='deprecated', n\_jobs=None,  
 param\_grid=[{'C': [0.001, 0.01, 0.1, 1, 10, 100],  
 'gamma': [0.001, 0.01, 0.1, 1, 10, 100],  
 'kernel': ['rbf']},  
 {'C': [0.001, 0.01, 0.1, 1, 10, 100],  
 'kernel': ['linear']}],  
 pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score=False,  
 scoring=None, verbose=0)

print("Best parameters: {}".format(grid\_search.best\_params\_))  
print("Best cross-validation score: {:.2f}".format(grid\_search.best\_score\_))

Best parameters: {'C': 10, 'gamma': 100, 'kernel': 'rbf'}  
Best cross-validation score: 0.91

print("Best estimator:\n{}".format(grid\_search.best\_estimator\_))

Best estimator:  
SVC(C=10, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,  
 decision\_function\_shape='ovr', degree=3, gamma=100, kernel='rbf',  
 max\_iter=-1, probability=False, random\_state=None, shrinking=True,  
 tol=0.001, verbose=False)

print("Test set score: {:.2f}".format(grid\_search.score(X\_test\_scaled, y\_test)))

Test set score: 0.90

**Best parameters: {'C': 10, 'gamma': 100, 'kernel': 'rbf'}**