*# SVC Classification*

*# Importing the libraries*

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** matplotlib.image **as** mpimg

**import** pandas **as** pd

*# Importing the dataset*

dataset **=** pd**.**read\_csv('iris.csv')

*#looking at the first 5 values of the dataset*

dataset**.**head()

%matplotlib inline

img=mpimg.imread('iristype.jpeg')

plt.figure(figsize=(20,40))

plt.axis('off')

plt.imshow(img)

*#Spliting the dataset in independent and dependent variables*

X **=** dataset**.**iloc[:,:4]**.**values

y **=** dataset['species']**.**values

*# Splitting the dataset into the Training set and Test set*

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

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size **=** 0.20, random\_state **=** 82)

*# Feature Scaling to bring the variable in a single scale*

*# StandardScaler is that it will transform your data such that its distribution will have a mean value 0 and standard deviation of 1.*

**from** sklearn.preprocessing **import** StandardScaler

sc **=** StandardScaler()

X\_train **=** sc**.**fit\_transform(X\_train)

X\_test **=** sc**.**transform(X\_test)

**Using Linear Kernel.**

*# Fitting SVC Classification to the Training set with linear kernel*

**from** sklearn.svm **import** SVC

svcclassifier **=** SVC(kernel **=** 'linear', random\_state **=** 0)

svcclassifier**.**fit(X\_train, y\_train)

*# Predicting the Test set results*

y\_pred **=** svcclassifier**.**predict(X\_test)

print(y\_pred)

*#lets see the actual and predicted value side by side*

*#* *vstack() function is used to stack the sequence of input arrays vertically to make a single array.*

y\_compare **=** np**.**vstack((y\_test,y\_pred))

*#actual value on the left side and predicted value on the right hand side*

*#printing the top 5 values*

y\_compare[:5,:]

*# Making the Confusion Matrix*

**from** sklearn.metrics **import** confusion\_matrix

cm **=** confusion\_matrix(y\_test, y\_pred)

print(cm)

*#finding accuracy from the confusion matrix.*

*# The shape of an array can be defined as the number of elements in each dimension*

a **=** cm**.**shape

corrPred **=** 0

falsePred **=** 0

**for** row **in** range(a[0]):

**for** c **in** range(a[1]):

**if** row **==** c:

corrPred **+=**cm[row,c]

**else**:

falsePred **+=** cm[row,c]

print('Correct predictions: ', corrPred)

print('False predictions', falsePred)

kernelLinearAccuracy **=** corrPred**/**(cm**.**sum())

print ('Accuracy of the SVC Clasification is: ', corrPred**/**(cm**.**sum()))

# Using polynomial kernel

*# Fitting SVC Classification to the Training set with linear kernel*

**from** sklearn.svm **import** SVC

svcclassifier **=** SVC(kernel **=** 'poly', random\_state **=** 0)

svcclassifier**.**fit(X\_train, y\_train)

*# Predicting the Test set results*

y\_pred **=** svcclassifier**.**predict(X\_test)

print(y\_pred)

*#lets see the actual and predicted value side by side*

y\_compare **=** np**.**vstack((y\_test,y\_pred))

*#actual value on the left side and predicted value on the right hand side*

*#printing the top 5 values*

y\_compare[:5,:]

*# Making the Confusion Matrix*

**from** sklearn.metrics **import** confusion\_matrix

cm **=** confusion\_matrix(y\_test, y\_pred)

print(cm)

*#finding accuracy from the confusion matrix.*

a **=** cm**.**shape

corrPred **=** 0

falsePred **=** 0

**for** row **in** range(a[0]):

**for** c **in** range(a[1]):

**if** row **==** c:

corrPred **+=**cm[row,c]

**else**:

falsePred **+=** cm[row,c]

print('Correct predictions: ', corrPred)

print('False predictions', falsePred)

kernelPolyAccuracy **=** corrPred**/**(cm**.**sum())

print ('Accuracy of the SVC Clasification is: ', corrPred**/**(cm**.**sum()))

# Using rbf kernel

*# Fitting SVC Classification to the Training set with linear kernel*

**from** sklearn.svm **import** SVC

svcclassifier **=** SVC(kernel **=** 'rbf', random\_state **=** 0)

svcclassifier**.**fit(X\_train, y\_train)

*# Predicting the Test set results*

y\_pred **=** svcclassifier**.**predict(X\_test)

print(y\_pred)

*#lets see the actual and predicted value side by side*

y\_compare **=** np**.**vstack((y\_test,y\_pred))

*#actual value on the left side and predicted value on the right hand side*

*#printing the top 5 values*

y\_compare[:5,:]

*# Making the Confusion Matrix*

**from** sklearn.metrics **import** confusion\_matrix

cm **=** confusion\_matrix(y\_test, y\_pred)

print(cm)

*#finding accuracy from the confusion matrix.*

a **=** cm**.**shape

corrPred **=** 0

falsePred **=** 0

**for** row **in** range(a[0]):

**for** c **in** range(a[1]):

**if** row **==** c:

corrPred **+=**cm[row,c]

**else**:

falsePred **+=** cm[row,c]

print('Correct predictions: ', corrPred)

print('False predictions', falsePred)

kernelRbfAccuracy **=** corrPred**/**(cm**.**sum())

print ('Accuracy of the SVC Clasification is: ', corrPred**/**(cm**.**sum()))

# Using sigmoid kernel

*# Fitting SVC Classification to the Training set with linear kernel*

**from** sklearn.svm **import** SVC

svcclassifier **=** SVC(kernel **=** 'sigmoid', random\_state **=** 0)

svcclassifier**.**fit(X\_train, y\_train)

*# Predicting the Test set results*

y\_pred **=** svcclassifier**.**predict(X\_test)

print(y\_pred)

*#lets see the actual and predicted value side by side*

y\_compare **=** np**.**vstack((y\_test,y\_pred))

*#actual value on the left side and predicted value on the right hand side*

*#printing the top 5 values*

y\_compare[:5,:]

*# Making the Confusion Matrix*

**from** sklearn.metrics **import** confusion\_matrix

cm **=** confusion\_matrix(y\_test, y\_pred)

print(cm)

*#finding accuracy from the confusion matrix.*

a **=** cm**.**shape

corrPred **=** 0

falsePred **=** 0

**for** row **in** range(a[0]):

**for** c **in** range(a[1]):

**if** row **==** c:

corrPred **+=**cm[row,c]

**else**:

falsePred **+=** cm[row,c]

print('Correct predictions: ', corrPred)

print('False predictions', falsePred)

kernelSigmoidAccuracy **=** corrPred**/**(cm**.**sum())

print('\n\nAccuracy of SVC classfication with different kernels are :\n\n\n')

print ('Accuracy of the SVC Clasification with Linear kernel and no other adjust is: \t\t\t', kernelLinearAccuracy)

print ('Accuracy of the SVC Clasification with Polynomial kernel and no other adjust is: \t\t', kernelPolyAccuracy)

print ('Accuracy of the SVC Clasification with Radial Basis Function kernel and no other adjust is: \t', kernelRbfAccuracy)

print ('Accuracy of the SVC Clasification with Sigmoid kernel and no other adjust is: \t\t\t', kernelSigmoidAccuracy)

# Output:

[[11 0 0]

[ 0 4 5]

[ 0 0 10]]

Correct predictions: 25

False predictions 5

Accuracy of SVC classfication with different kernels are :

Accuracy of the SVC Clasification with Linear kernel and no other adjust is: 0.966666666667

0.966666666667

Accuracy of the SVC Clasification with Polynomial kernel and no other adjust is:

0.933333333333

Accuracy of the SVC Clasification with Radial Basis Function kernel and no other adjust is:

0.833333333333

Accuracy of the SVC Clasification with Sigmoid kernel and no other adjust is: 0.833333333333