Data Analysis and Machine-Learning

Chapter 6.3:

Support Vector Machine (3)

Applications Part.1: Logistic Regression



Now that we have understood the (1) mathematical and algorithmic frameworks of the SVM and (2) specific parameters, let us apply SVM for modelling with actual datasets.

In this chapter, let us start with the application of logistic regression model for review and comparison, before getting into SVC for classifications.

1. Classification of Iris Sample Data

#Import Essential Libraries

import numpy as np

import pandas as pd

from sklearn import datasets

#Load sample data

iris = datasets.load\_iris()

iris.keys()

OUTPUT:

dict\_keys(['data', 'target', 'frame', 'target\_names', 'DESCR', 'feature\_names', 'filename'])

iris['feature\_names']

OUTPUT:

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

iris['target\_names']

OUTPUT:

array(['setosa', 'versicolor', 'virginica'], dtype='<U10')

iris['target']

OUTPUT:

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])

#Define features(x) and target(y)

features = pd.DataFrame(data=iris['data'], columns=iris['feature\_names'])

target = pd.DataFrame(data=iris['target'], columns=['target'])

#Divide train and test data (e.g., 80:20)

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, shuffle=True,random\_state=123)

#Logistic regression

from sklearn.linear\_model import LogisticRegression

logitModel = LogisticRegression(max\_iter=100000).fit(x\_train, y\_train.values.ravel())

yhat = logitModel.predict(x\_train)

#Evaluation

from sklearn.metrics import f1\_score, accuracy\_score, roc\_auc\_score, classification\_report, ConfusionMatrixDisplay, \

    confusion\_matrix

confusion\_matrix(y\_train, yhat)

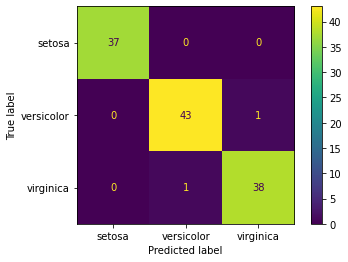
OUTPUT:

array([[37, 0, 0], [ 0, 43, 1], [ 0, 1, 38]], dtype=int64)

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y\_train, yhat), display\_labels=iris['target\_names'])

disp.plot()

OUTPUT:



#F1-score and accuracy score

display(f1\_score(y\_train, yhat, average = None))

display(accuracy\_score(y\_train, yhat))

OUTPUT:

array([1. , 0.97727273, 0.97435897])

0.9833333333333333

f1\_score(y\_train, yhat, average='weighted')

OUTPUT:

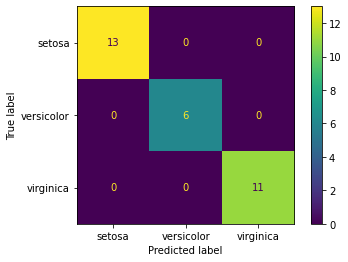
0.9833333333333333

#Test Scores and Evaluation

yhat = logitModel.predict(x\_test)

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y\_test, yhat), display\_labels=iris['target\_names'])

disp.plot()



f1\_score(y\_test, yhat, average=None)

OUTPUT:

array([1., 1., 1.])

f1\_score(y\_test, yhat, average='weighted')

OUTPUT:

1.0

accuracy\_score(y\_test, yhat)

OUTPUT:

1.0

2. Classification of Breast Cancer Sample Data

#Import Essential Libraries

import numpy as np

import pandas as pd

from sklearn.metrics import accuracy\_score, roc\_auc\_score, f1\_score, classification\_report, \

    confusion\_matrix, ConfusionMatrixDisplay

from sklearn import datasets

from sklearn.linear\_model import LinearRegression, LogisticRegression

from sklearn.svm import SVC, SVR

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

#Load sample dataset

cancer = datasets.load\_breast\_cancer()

cancer.keys()

OUTPUT:

dict\_keys(['data', 'target', 'frame', 'target\_names', 'DESCR', 'feature\_names', 'filename'])

cancer['feature\_names']

OUTPUT:

array(['mean radius', 'mean texture', 'mean perimeter', 'mean area', 'mean smoothness', 'mean compactness', 'mean concavity', 'mean concave points', 'mean symmetry', 'mean fractal dimension', 'radius error', 'texture error', 'perimeter error', 'area error', 'smoothness error', 'compactness error', 'concavity error', 'concave points error', 'symmetry error', 'fractal dimension error', 'worst radius', 'worst texture', 'worst perimeter', 'worst area', 'worst smoothness', 'worst compactness', 'worst concavity', 'worst concave points', 'worst symmetry', 'worst fractal dimension'], dtype='<U23')

cancer['target\_names']

OUTPUT:

array(['malignant', 'benign'], dtype='<U9')

cancer['target']

OUTPUT:

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])

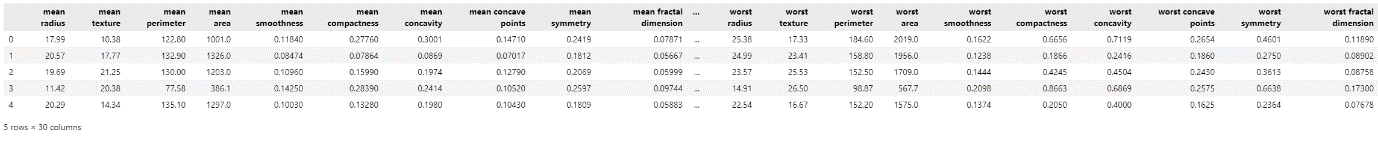
#Define Features and Target

features = pd.DataFrame(data = cancer['data'], columns=cancer['feature\_names'])

target = pd.DataFrame(data=cancer['target'], columns=['target'])

features.head()

OUTPUT:



target['target'].unique()

OUTPUT:

array([0, 1])

target['target'].value\_counts()

OUTPUT:

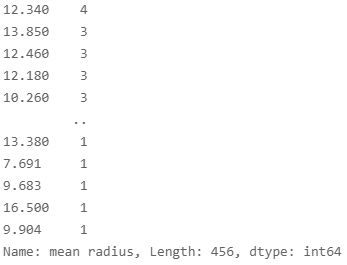
1 357

0 212

Name: target, dtype: int64

features['mean radius'].value\_counts()

OUTPUT:



#Train-test split (80:20)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, shuffle=True, random\_state=123)

#Logistic Regression

logitModel = LogisticRegression(max\_iter=100000).fit(x\_train, y\_train.values.ravel())

yhat = logitModel.predict(x\_train)

#Evaluation

display(accuracy\_score(y\_train, yhat))

display(f1\_score(y\_train, yhat))

OUTPUT:

0.9560439560439561

0.965034965034965

#Confusion Matrix

cm = confusion\_matrix(y\_train, yhat)

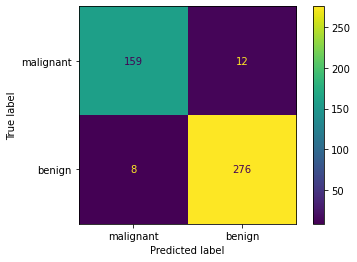
cm

array([[159, 12],

[ 8, 276]], dtype=int64)

disp = ConfusionMatrixDisplay(cm, display\_labels=cancer['target\_names'])

disp.plot()



#Test Data Evaluation

yhat = logitModel.predict(x\_test)

display(accuracy\_score(y\_test, yhat))

display(f1\_score(y\_test, yhat))

OUTPUT:

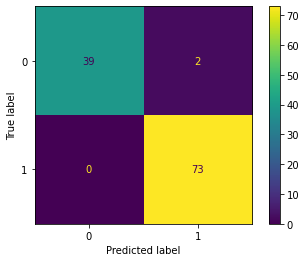
0.9824561403508771

0.9864864864864865

cm = confusion\_matrix(y\_test, yhat)

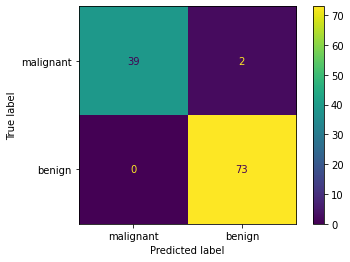
disp = ConfusionMatrixDisplay(cm)

disp.plot()



disp = ConfusionMatrixDisplay(cm, display\_labels=cancer['target\_names'])

disp.plot()



3. Classification of Spam Mail Sample Data

#Import Libraries

import numpy as np

import pandas as pd

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

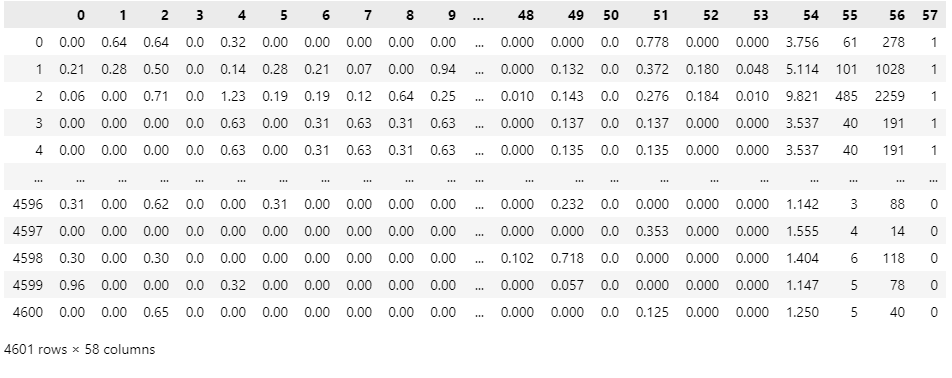
from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import f1\_score, accuracy\_score, roc\_auc\_score, confusion\_matrix, ConfusionMatrixDisplay

#Import Data

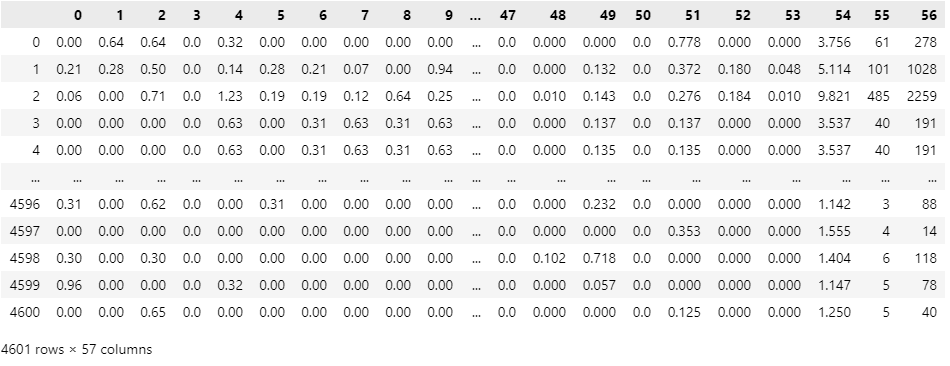
spam = pd.read\_csv('YOUR FILE PATH\\spam.data', sep=' ', header=None)

spam



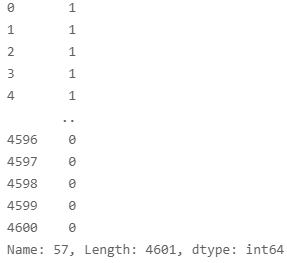
features = spam.loc[:, 0:56]

features



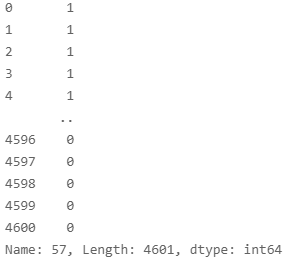
target = spam.loc[:, 57]

target



target = spam[57]

target



#Train test split (80:20)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, shuffle=True, random\_state=123)

#Logistic Regression

logitModel = LogisticRegression(max\_iter=10000).fit(x\_train, y\_train)

yhat = logitModel.predict(x\_train)

#Evaluation

display(accuracy\_score(y\_train, yhat))

display(f1\_score(y\_train, yhat))

display(roc\_auc\_score(y\_train, yhat))

OUTPUT:

0.9277173913043478

0.906993006993007

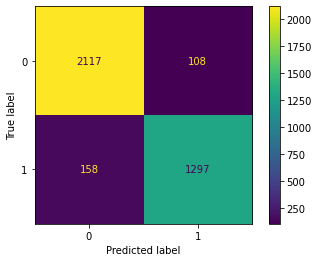
0.9214348044326036

cm = confusion\_matrix(y\_train, yhat)

disp = ConfusionMatrixDisplay(cm)

disp.plot()

OUTPUT:



#Test Evaluation

yhat = logitModel.predict(x\_test)

display(accuracy\_score(y\_test, yhat))

display(f1\_score(y\_test, yhat))

display(roc\_auc\_score(y\_test, yhat))

OUTPUT:

0.9370249728555917

0.9185393258426967

0.9327252250017365

cm = confusion\_matrix(y\_test, yhat)

disp = ConfusionMatrixDisplay(cm)

disp.plot()

