**LAB - 5**

**PRINCIPAL COMPONENT ANALYSIS**

**PRIYANSHU SHARMA**

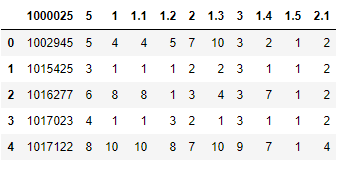
**15BCE1282**

**CODE**

import pandas as pd

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/breast.csv')

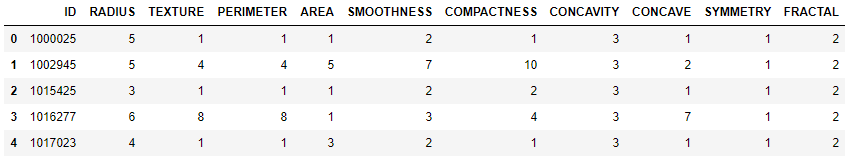
data.head()



colnames=['ID', 'RADIUS', 'TEXTURE', 'PERIMETER', 'AREA', 'SMOOTHNESS', 'COMPACTNESS', 'CONCAVITY', 'CONCAVE', 'SYMMETRY', 'FRACTAL']

data = pd.read\_csv('C:/Users/PRIYANSHU SHARMA/Desktop/PRIYANSHU/6 STUDY/6 SEMSTER/MACHINE LEARNING/LAB/breast.csv', names=colnames, header=None)

data.head()



from sklearn.cross\_validation import train\_test\_split

from sklearn.preprocessing import StandardScaler

X = data.iloc[0:, 1:10].values

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,data['FRACTAL'], test\_size=0.3, random\_state=0)

sc = StandardScaler()

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

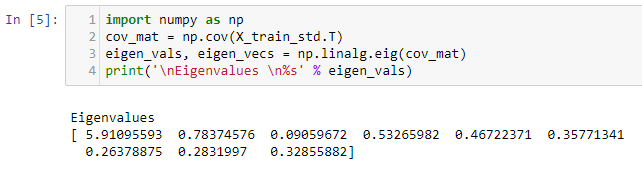
X\_test\_std = sc.fit\_transform(X\_test)

import numpy as np

cov\_mat = np.cov(X\_train\_std.T)

eigen\_vals, eigen\_vecs = np.linalg.eig(cov\_mat)

print('\nEigenvalues \n%s' % eigen\_vals)



tot = sum(eigen\_vals)

var\_exp = [(i / tot) for i in

sorted(eigen\_vals, reverse=True)]

cum\_var\_exp = np.cumsum(var\_exp)

import matplotlib.pyplot as plt

plt.bar(range(1,10), var\_exp, alpha=0.5, align='center',

label='individual explained variance')

plt.step(range(1,10), cum\_var\_exp, where='mid',

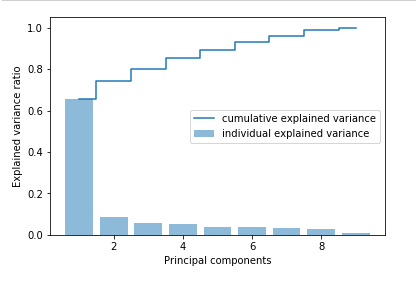
label='cumulative explained variance')

plt.ylabel('Explained variance ratio')

plt.xlabel('Principal components')

plt.legend(loc='best')

plt.show()



eigen\_pairs =[(np.abs(eigen\_vals[i]),eigen\_vecs[:,i])

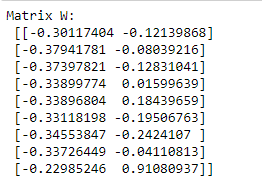
for i in range(len(eigen\_vals))]

eigen\_pairs.sort(reverse=True)

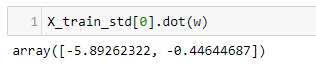
w= np.hstack((eigen\_pairs[0][1][:, np.newaxis],

eigen\_pairs[1][1][:, np.newaxis]))

print('Matrix W:\n',w)



X\_train\_std[0].dot(w)



X\_train\_pca = X\_train\_std.dot(w)

colors = ['r', 'b', 'green','orange']

markers = ['s', 'x', 'o','^']

for l, c, m in zip(np.unique(y\_train), colors, markers):

plt.scatter(X\_train\_pca[y\_train==l, 0],

X\_train\_pca[y\_train==l, 1],

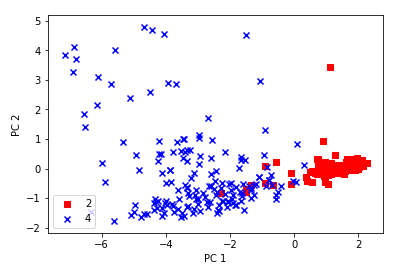
c=c, label=l, marker=m)

plt.xlabel('PC 1')

plt.ylabel('PC 2')

plt.legend(loc='lower left')

plt.show()



from matplotlib.colors import ListedColormap

def plot\_decision\_regions(X, y, classifier, resolution=0.02):

# setup marker generator and color map

markers = ('s', 'x', 'o', '^', 'v')

colors = ('red', 'blue', 'green', 'orange', 'cyan')

cmap = ListedColormap(colors[:len(np.unique(y))])

# plot the decision surface

x1\_min, x1\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

x2\_min, x2\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx1, xx2 = np.meshgrid(np.arange(x1\_min, x1\_max, resolution),

np.arange(x2\_min, x2\_max, resolution))

Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)

Z = Z.reshape(xx1.shape)

plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)

plt.xlim(xx1.min(), xx1.max())

plt.ylim(xx2.min(), xx2.max())

# plot class samples

for idx, cl in enumerate(np.unique(y)):

plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],

alpha=0.8, c=cmap(idx),

marker=markers[idx], label=cl)

from sklearn.linear\_model import LogisticRegression

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

lr = LogisticRegression()

X\_train\_pca = pca.fit\_transform(X\_train\_std)

X\_test\_pca = pca.transform(X\_test\_std)

lr.fit(X\_train\_pca, y\_train)

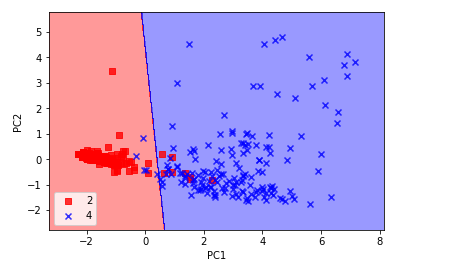
plot\_decision\_regions(X\_train\_pca, y\_train, classifier=lr)

plt.xlabel('PC1')

plt.ylabel('PC2')

plt.legend(loc='lower left')

plt.show()



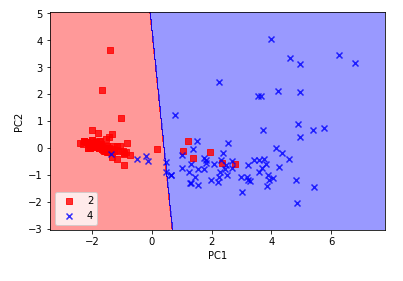
plot\_decision\_regions(X\_test\_pca, y\_test, classifier=lr)

plt.xlabel('PC1')

plt.ylabel('PC2')

plt.legend(loc='lower left')

plt.show()



pca = PCA(n\_components=None)

X\_train\_pca = pca.fit\_transform(X\_train\_std)

pca.explained\_variance\_ratio\_

