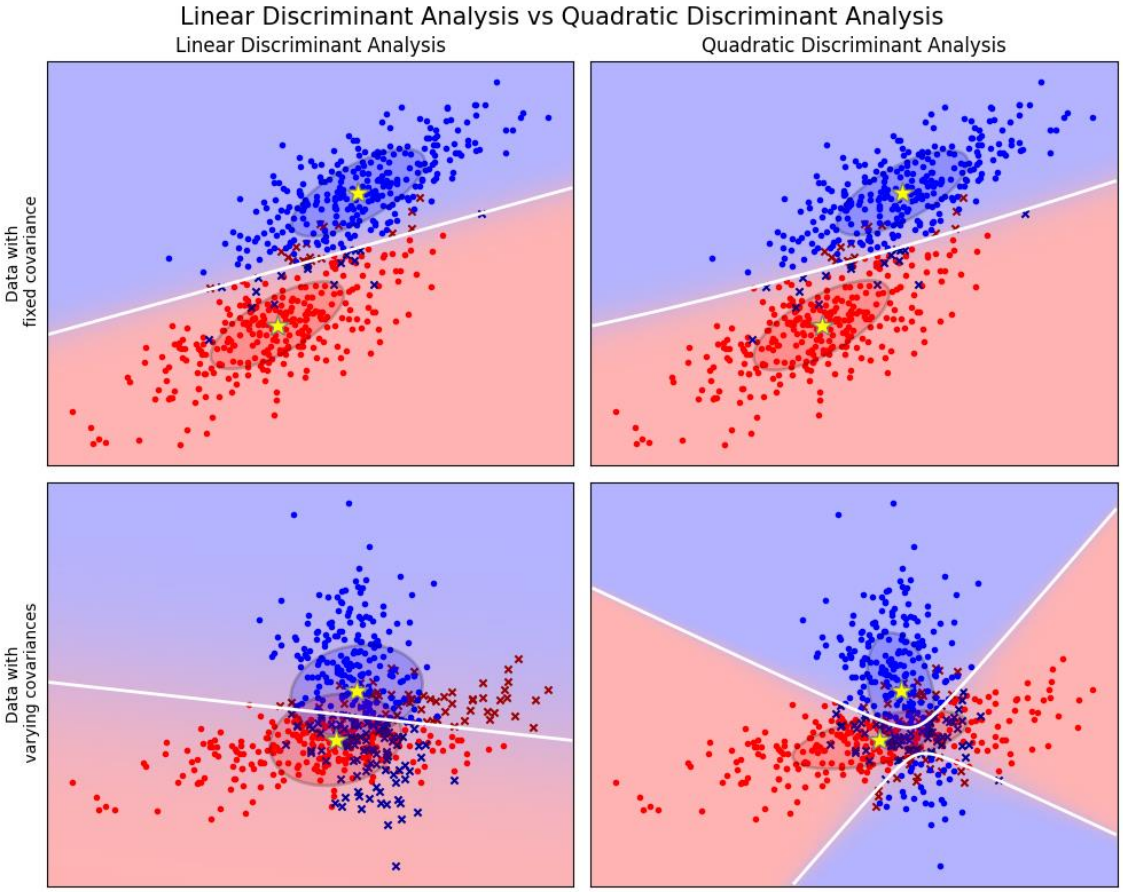
Lab 3 – Dimensionality reduction.Linear discriminant analysis and principal componente analysis

Author: Gonçalo Aguiar 904475

1. **Open the analysis_LDA_i_QDA.py program. Examine the code, in particular lines 155-156 and 162-163. Run the program and view the results, read the comment below.**

   After running the given code the result is the following plots:



   This plot compares two methods of classification: Linear Discriminant Analysis and Quadratic Discriminant Analysis. The bottom row of the plot shows that while Linear Discriminant Analysis is limited to learning only linear decision boundaries, Quadratic Discriminant Analysis can learn both linear and curved boundaries, making it a more versatile method.

2. **Complete the lda.py program at the locations marked with a # !!! comment, in order to implement the LDA algorithm given below. Calculate the covariance matrix using the np.cov() function with the parameter rowvar = False. Use the np.linalg.inv() function to calculate the inverse of the matrix. Use the .dot() method on numpy arrays or the np.dot() function to calculate dot products. Use the np.linalg.eigh() function to calculate eigenvectors and eigenvalues. Use the .T method to transpose matrices.**

   In order to fullfil the task the following code was completed:

```
from sklearn import datasets
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

iris = datasets.load_iris()
X = iris.data
y = iris.target

# Visualising the original data (first three features)

fig = plt.figure(1)
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

plt.cla()

for name, label in [('Setosa', 0), ('Versicolour', 1), ('Virginica', 2)]:
    ax.text3D(X[y == label, 0].mean(),
              1.1*X[y == label, 1].mean(),
              X[y == label, 2].mean(), name,
              horizontalalignment='center',
              bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))

# Reorder the labels to have colors matching the cluster results
y = np.choose(y, [1, 2, 0]).astype(float)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y, cmap=plt.cm.Set1,
           edgecolor='k')

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])

plt.show()


# set the target number of features (dimensions)
# !!! k =
k=2
```

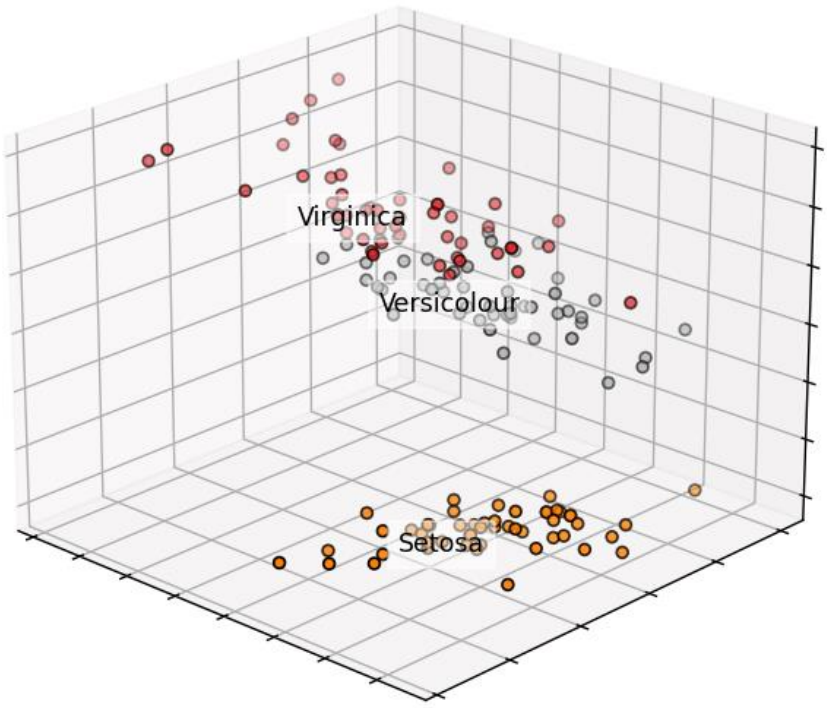
```
n_features = X.shape[1]
labels = np.unique(y)

# within-class scatter matrix
S_W = np.zeros((n_features, n_features))
for label in labels:
    xi = X[y == label]

    # increase S_W by a component for class i
    S_W = S_W + (len(xi)-1)*np.cov(xi,rowvar=False)

# between-class scatter matrix

# calculate the mean value of the dataset, total_mean
total_mean = np.mean(X,axis=0)

# create an empty array S_B with a proper shape
S_B = np.zeros((n_features,n_features))

for label in labels:
    xi = X[y == label]
    # calculate the mean mi of examples in class i
    mi= np.mean(X,axis=0)

    S_B += len(xi) * (mi - total_mean).dot((mi - total_mean).T)

# compute the matrix A = S_W^(-1) * S_B
# !!!
A = np.dot(np.linalg.inv(S_W),S_B)

# compute the eigenvectors and eigenvalues of A = S_W^(-1) * S_B
eigenvalues, eigenvectors = np.linalg.eigh(A)

# search for indices idx of the eigenvalues sorted in a decreasing order
idx = eigenvalues.argsort()[::-1]

# take the first k eigenvectors (so will be the number of features)
P = eigenvectors[:, idx][:, :k]
```

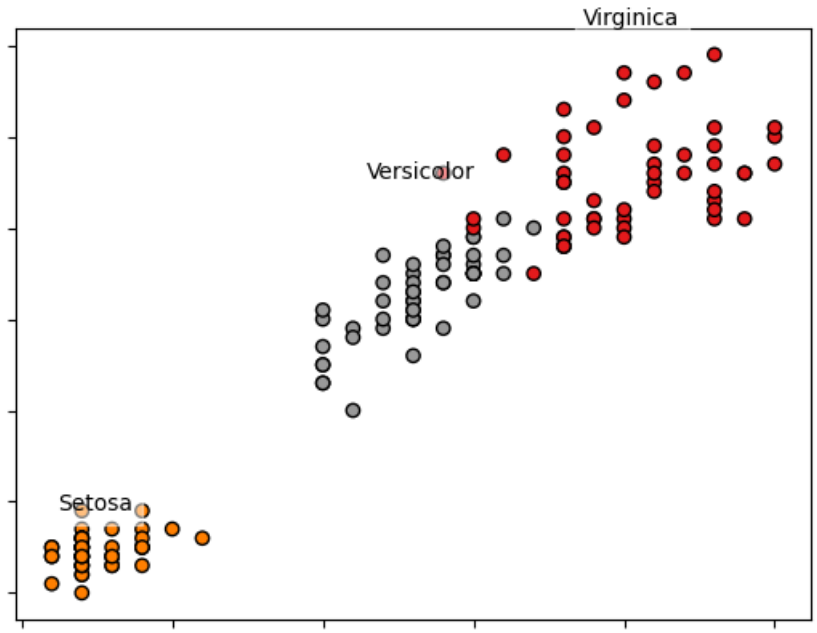
```
# scalar product X*P to get the projected space
new_space = np.dot(X,P)

if k == 3:
    y = iris.target

    fig = plt.figure(2)
    plt.clf()
    ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

    plt.cla()

    for name, label in [('Setosa', 0), ('Versicolor', 1), ('Virginica', 2)]:
        ax.text3D(new_space[y == label, 0].mean(),
                  1.1*new_space[y == label, 1].mean(),
                  new_space[y == label, 2].mean(), name,
                  horizontalalignment='center',
                  bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))

    # Reorder the labels to have colors matching the cluster results
    y = np.choose(y, [1, 2, 0]).astype(float)

    ax.scatter(new_space[:, 0], new_space[:, 1], new_space[:, 2], c=y, cmap=plt.cm.Set1,
               edgecolor='k')

    ax.w_xaxis.set_ticklabels([])
    ax.w_yaxis.set_ticklabels([])
    ax.w_zaxis.set_ticklabels([])

    plt.show()
elif k==2:
    y = iris.target

    fig, ax = plt.subplots()

    # plt.clf()
    # plt.cla()
```

```
for name, label in [('Setosa', 0), ('Versicolor', 1), ('Virginica', 2)]:
    ax.text(new_space[y == label, 0].mean(),
            1.3*new_space[y == label, 1].mean(),
            name,
            horizontalalignment='center',
            bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))

y = np.choose(y, [1, 2, 0]).astype(float)

ax.scatter(new_space[:, 0], new_space[:, 1], c=y, cmap=plt.cm.Set1,
           edgecolor='k')

ax.xaxis.set_ticklabels([])
ax.yaxis.set_ticklabels([])
plt.show()
```

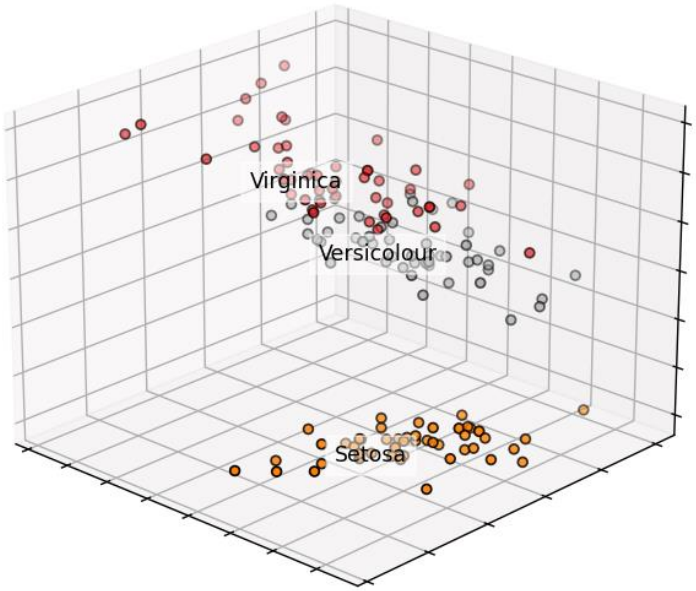
After running the code the result was the following plots:

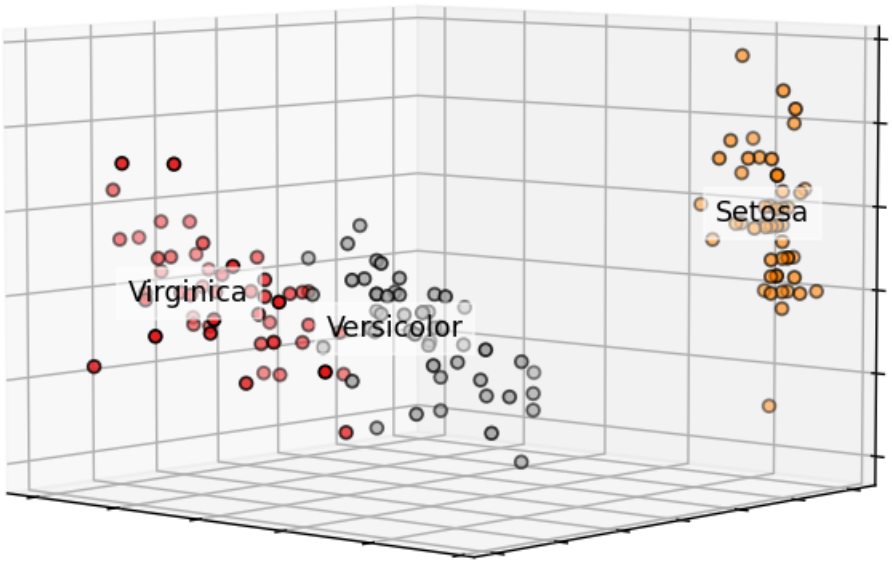
With K=2:



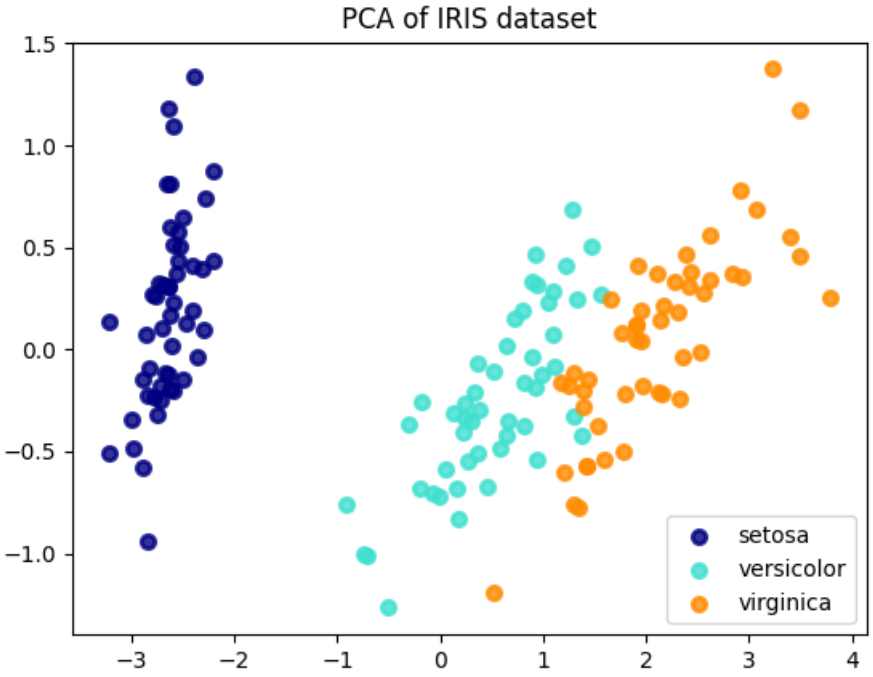


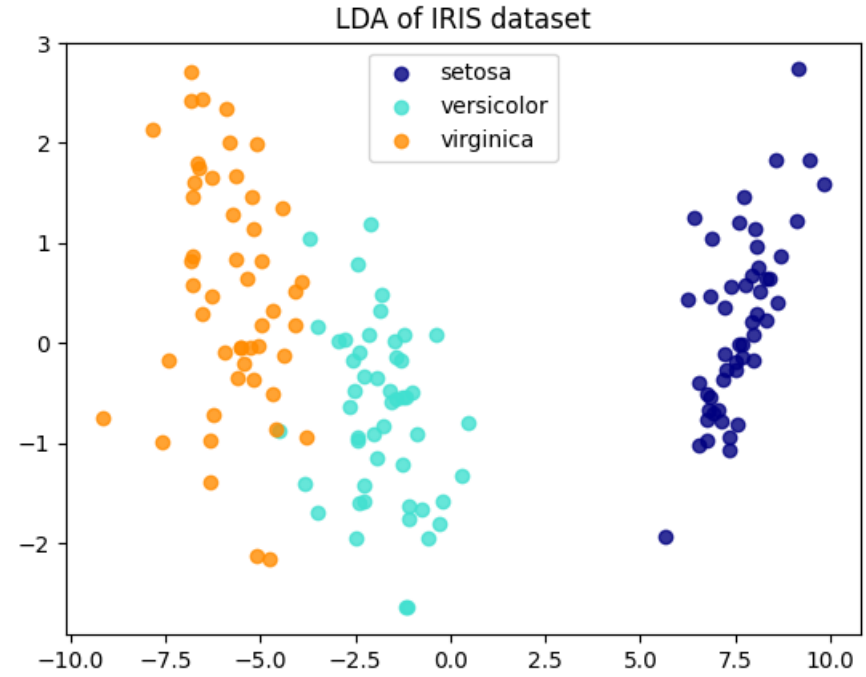
With K=3:


Virginica
Versicolour
Setosa


Setosa
Virginica
Versicolor

3. **Load, analyze and run the pca_sklearn.py program. Investigate the differences in LDA and PCA results.**





If we compare both plots we can see that the X and Y axes have different scales, and the LDA plot for the Iris dataset has an opposite appearance to the PCA plot.

4. **Write a program in which you perform classification on the Iris dataset transformed using LDA and PCA (reduce the dimensionality of the feature space to 2 as in the previous task). After reduction, select only samples from the versicolor and virginica classes for classification. Use logistic regression as the classifier. In both cases, count how many samples in the test set are misclassified.**

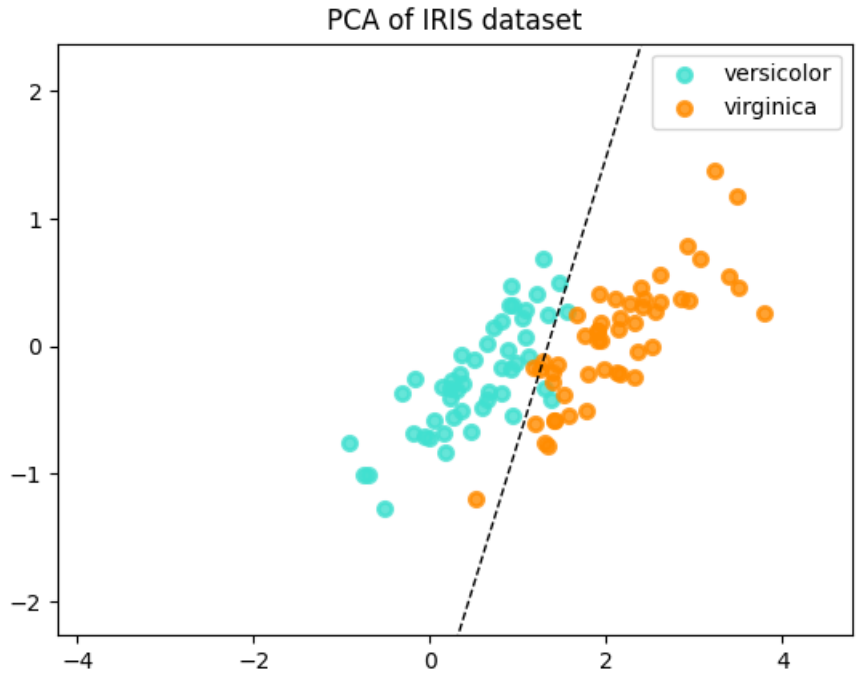
In order to complete the task the following code was written:

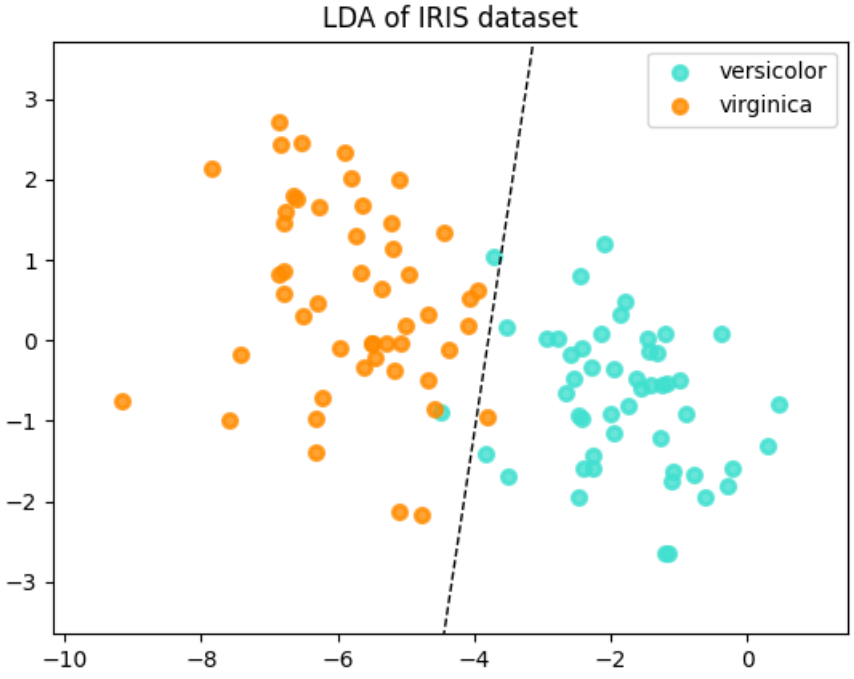
```
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names

pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)

lda = LinearDiscriminantAnalysis(n_components=2,solver="eigen")
X_r2 = lda.fit(X, y).transform(X)

# Percentage of variance explained for each components
print(
    "explained variance ratio (first two components): %s"
    % str(pca.explained_variance_ratio_)
)

plt.figure()
colors = ["navy", "turquoise", "darkorange"]
lw = 2

for color, i, target_name in zip(colors[1:], [1, 2], target_names[1:]):
    plt.scatter(
        X_r[y == i, 0], X_r[y == i, 1], color=color, alpha=0.8, lw=lw, label=target_name
    )
plt.legend(loc="best", shadow=False, scatterpoints=1)
plt.title("PCA of IRIS dataset")

from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0).fit(X_r[50:],y[50:])
import numpy as np

b = clf.intercept_[0]
```

```
w1,w2 = clf.coef_.T

c = -b/w2
m = -w1/w2

xmin, xmax = np.min(X_r,0)[0]-1, np.max(X_r,0)[0]+1
ymin, ymax = np.min(X_r,0)[1]-1, np.max(X_r,0)[1]+1

xd = np.array([xmin,xmax])
yd = m*xd +c

plt.plot(xd, yd, 'k', lw=1, ls='--')
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)
plt.figure()


for color, i, target_name in zip(colors[1:], [1, 2], target_names[1:]):
    plt.scatter(
        X_r2[y == i, 0], X_r2[y == i, 1], color=color, alpha=0.8, lw=lw, label=target_name
    )
plt.legend(loc="best", shadow=False, scatterpoints=1)
plt.title("LDA of IRIS dataset")

clf = LogisticRegression(random_state=0).fit(X_r2[50:],y[50:])

b = clf.intercept_[0]

w1,w2 = clf.coef_.T

c = -b/w2
m = -w1/w2

xmin, xmax = np.min(X_r2[50:],0)[0]-1, np.max(X_r2[50:],0)[0]+1
ymin, ymax = np.min(X_r2[50:],0)[1]-1, np.max(X_r2[50:],0)[1]+1

xd = np.array([xmin,xmax])
yd = m*xd +c
```

```
plt.plot(xd, yd, 'k', lw=1, ls='--')
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)

plt.show()
```

After running the code the result was the following:





LDA: We have 1 virginica and 2 versicolor miss classified.

PCA: We have 2 virginica and 3 versicolor miss classified.