IT581 Adversarial Machine Learning

Lab Assignment - 01

Group 05

2021 17013: Shaikh Faizan Ahmed
2021 17014: Sonam Bharti

Question 1.

Write a function that plots for a given norm p. Use this function to plot for p = 1,2,3,inf.

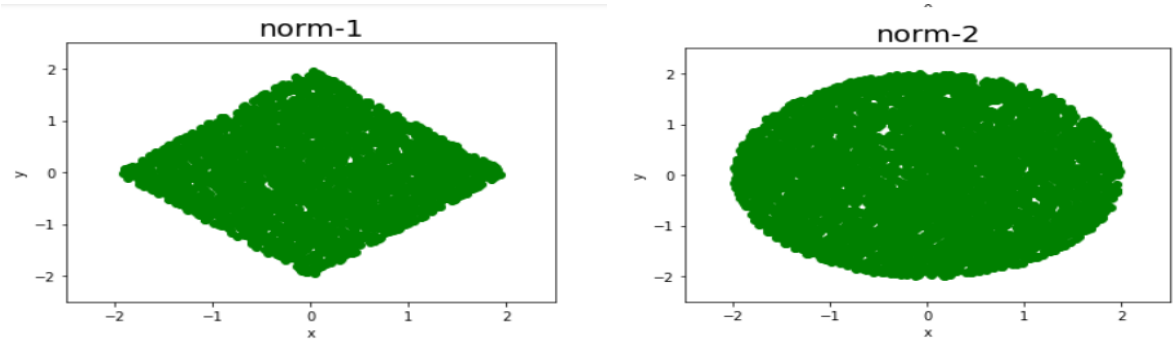
Answer

code

```
#import required libraries
import numpy as np
import pylab

#define function to find norm
def plot_p_norm(p):
    for i in range(6000):
        x = np.array([np.random.rand()*4-2, np.random.rand()*4-2])

        if np.linalg.norm(x,p) < 2:
            pylab.plot(x[0], x[1], 'go')
    pylab.title(label=f"norm-%f"%p,fontsize=20)
    pylab.xlabel('x')
    pylab.ylabel('y')
    pylab.axis([-2.5, 2.5, -2.5, 2.5])
    pylab.show()



p = np.array([1,2,3,np.inf])
for i in p:
    plot_p_norm(i)
```

Listing 1: Question 1

Output



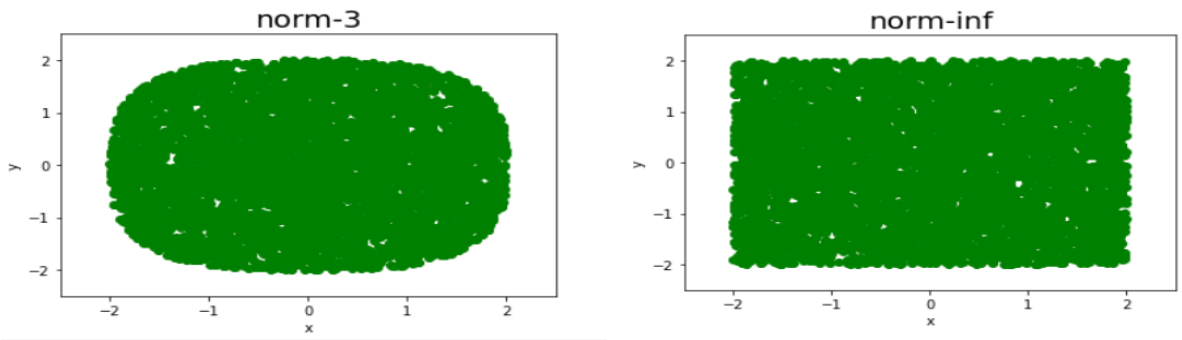


Figure 1: norm Plots

Observation/ Justification

Norms are a way of measuring the "length" of vectors, matrices, etc.
Mathematically, a norm is any function `f that satisfies....`

- $f(x) = 0 \Rightarrow x = 0$
- $f(x + y) \leq f(x) + f(y) \quad (TriangleInequality)$
- $\forall a \in, f(ax) = |a|f(x) \quad (Linearity)$

`The most commonly used norms are clubbed under p-norms (or l-norms) family where p is any number greater or equal to 1.`
`Formula to calculate the p-form of vector x...`

$$||x||_p = (x_1^p + x_2^p + ... + x_n^p)^{1/p}$$

`It can be also written as.....`

$$||x||_p = (\sum_{i=1}^{n} x_i^p)1/p$$

`where,`
`p = 1 for` $L^1 norm$
`p = 2 for` $L^2 norm$
`p = 3 for` $L^3 norm$
`p = inf for` $L^\infty norm$
`Conclusion: The function we have written in the above code follows this norms equation and returns a norm value for each vector and plot a graph upto 6000 vectors for each required value of p.`

Question 2.

`You will work with a widely used Iris dataset. The Iris Dataset contains four features (sepal length, sepal width, petal length, and petal width) of 50 samples of three species of Iris (Iris setosa, Iris virginica, and Iris versicolor). For this lab we'll use only two species. Visualize the data in the Iris Dataset by considering maximum combinations of two features in a 2D plot. Use red and green colors for labeling the two classes: Iris setosa and Iris versicolor, respectively. Comment on whether any two classes among the three can be separated by a line? Plot its linear discriminant function as line using two features. And also plot hyperplane using 3 features....`

Answer

code

```
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder as le
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis


iris_dat = pd.read_csv("Iris.csv")

new_iris = iris_dat[(iris_dat.Species == 'Iris-setosa') | (iris_dat.Species == 'Iris-
    versicolor')]
iris_df = new_iris.iloc[:,1:6]
iris_data = new_iris.iloc[:,1:5]

fig1 = px.scatter(iris_df, x=iris_df['SepalLengthCm'],
                  y=iris_df['SepalWidthCm'],
                  labels={"x":"Sepal Length","y":"Sepal Width"},
                  opacity=1,
                  color=iris_df['Species'],
                  color_discrete_sequence=["red", "green"])
fig1.update_layout(title_text="Sepal Length VS Sepal Width",
                  xaxis_title="Sepal Length in cm",
                  yaxis_title="Sepal Width in cm")
fig1.show()

fig2 = px.scatter(iris_df, x=iris_df['PetalLengthCm'],
                  y=iris_df['PetalWidthCm'],
                  labels={"x":"Petal Length","y":"Petal Width"},
                  opacity=1,
                  color=iris_df['Species'],
                  color_discrete_sequence=["red", "green"])
fig2.update_layout(title_text="Petal Length VS Petal Width",
                  xaxis_title="Petal Length in cm",
                  yaxis_title="Petal Width in cm")
fig2.show()


fig3 = px.scatter(iris_df, x=iris_df['SepalLengthCm'],
                  y=iris_df['PetalLengthCm'],
                  labels={"x":"Sepal Length","y":"Petal Length"},
                  opacity=1,
                  color=iris_df['Species'],
                  color_discrete_sequence=["red", "green"])
fig3.update_layout(title_text="Sepal Length VS Petal Length",
                  xaxis_title="Sepal Length in cm",
                  yaxis_title="Petal Length in cm")
fig3.show()


fig4 = px.scatter(iris_df, x=iris_df['SepalLengthCm'],
                  y=iris_df['PetalWidthCm'],
                  labels={"x":"Sepal Length","y":"Petal Width"},
                  opacity=1,
                  color=iris_df['Species'],
                  color_discrete_sequence=["red", "green"])
fig4.update_layout(title_text="Sepal Length VS Petal Width",
                  xaxis_title="Sepal Length in cm",
                  yaxis_title="Petal Width in cm")
fig4.show()


fig5 = px.scatter(iris_df, x=iris_df['SepalWidthCm'],
                  y=iris_df['PetalLengthCm'],
                  labels={"x":"Sepal Width","y":"Petal Length"},
                  opacity=1,
                  color=iris_df['Species'],
                  color_discrete_sequence=["red", "green"])
fig5.update_layout(title_text="Sepal Width VS Petal Length",
                  xaxis_title="Sepal Width  in cm",
```

```
                    yaxis_title="Petal Length in cm")
fig5.show()

fig6 = px.scatter(iris_df, x=iris_df['SepalWidthCm'],
                    y=iris_df['PetalWidthCm'],
                    labels={"x":"Sepal Width","y":"Petal Width"},
                    opacity=1,
                    color=iris_df['Species'],
                    color_discrete_sequence=["red", "green"])

fig6.update_layout(title_text="Sepal Width VS Petal Width",
                    xaxis_title="Sepal Width in cm",
                    yaxis_title="Petal Width in cm")
fig6.show()


A = iris_data.to_numpy()
X = A[:,:2]
y = iris_df.iloc[:,-1]
y = le().fit_transform(y)

lda = LinearDiscriminantAnalysis()
lda.fit_transform(X, y)

i, w0, w1 = lda.intercept_[0], lda.coef_[0][0], lda.coef_[0][1]
x0 = np.array([X[:,0].min(), X[:,0].max()])
y0 = -(i+x0*w0)/w1

fig7  = px.scatter(iris_df, x=iris_df['SepalLengthCm'], y=iris_df['SepalWidthCm'],
                     labels={"x":"Sepal Length","y":"Sepal Width"},
                     opacity=1,
                     color=iris_df['Species'],
                     color_discrete_sequence=["red", "green"])
fig7.add_traces(go.Scatter(x=x0, y=y0, name="LDA"))
fig7.update_layout(title_text="Linear Discriminant Function as Line using Two Features",
                    xaxis_title="Sepal Length in cm",
                    yaxis_title="Sepal Width in cm")
fig7.show()


X1 = A[:,:3]
lda1 = LinearDiscriminantAnalysis()
lda1.fit_transform(X1, y)


i, w0, w1, w2 = lda1.intercept_[0], lda1.coef_[0][0], lda1.coef_[0][1], lda1.coef_[0][2]
x0 = np.array([X[:,0].min(), X[:,0].max()])
x1 = np.array([X[:,1].min(), X[:,1].max()])

x0,x1= np.meshgrid(x0,x1)
y0 = -(i+x0*w0+x1*w1)/w2

fig8 = px.scatter_3d(iris_df,
                      x='SepalLengthCm', y='SepalWidthCm', z='PetalLengthCm',
                      color='Species',
                      color_discrete_sequence=["red", "green"],
                      height=900, width=900)
fig8.add_traces(go.Surface(x=x0, y=x1, z=y0, name="LDA", colorscale='blues'))
fig8.update_layout(title_text="Linear Discriminant Function as Line using Three Features",
                    showlegend=True,
                    legend=dict(orientation="h", yanchor="top", y=0, xanchor="center", x=0.5))
fig8.show()
```

Listing 2: Question 2

Output

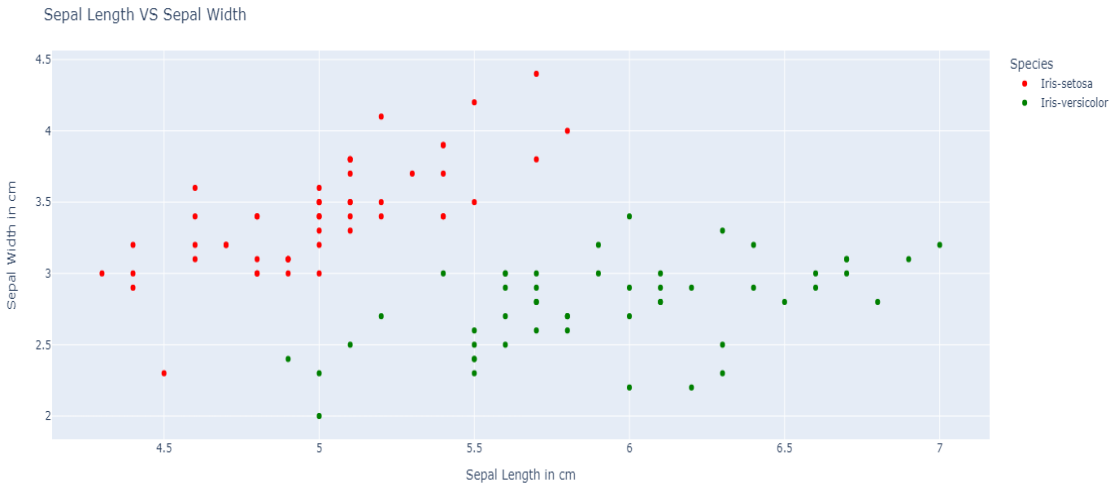


Figure 2: Sepal-Sepal-features

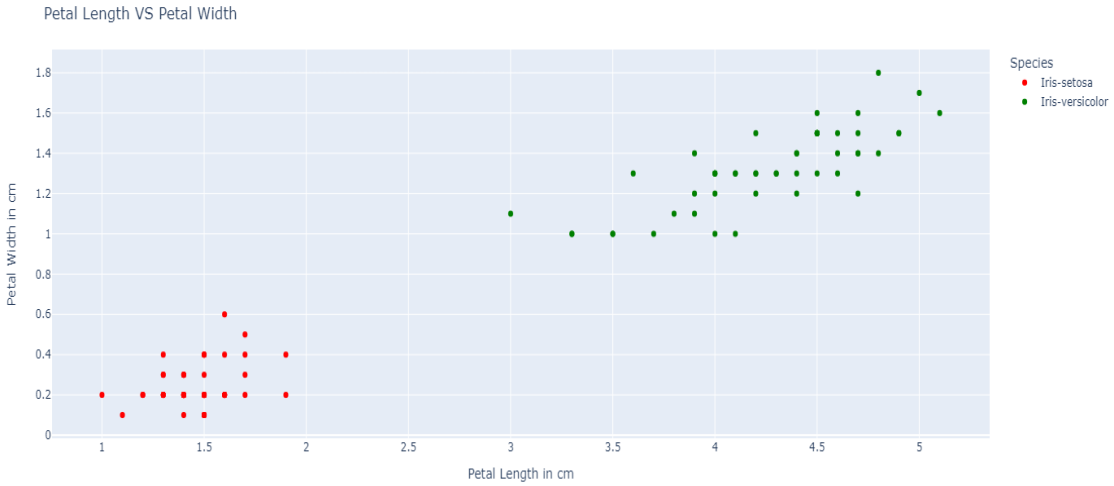


Figure 3: Petal-Petal-features

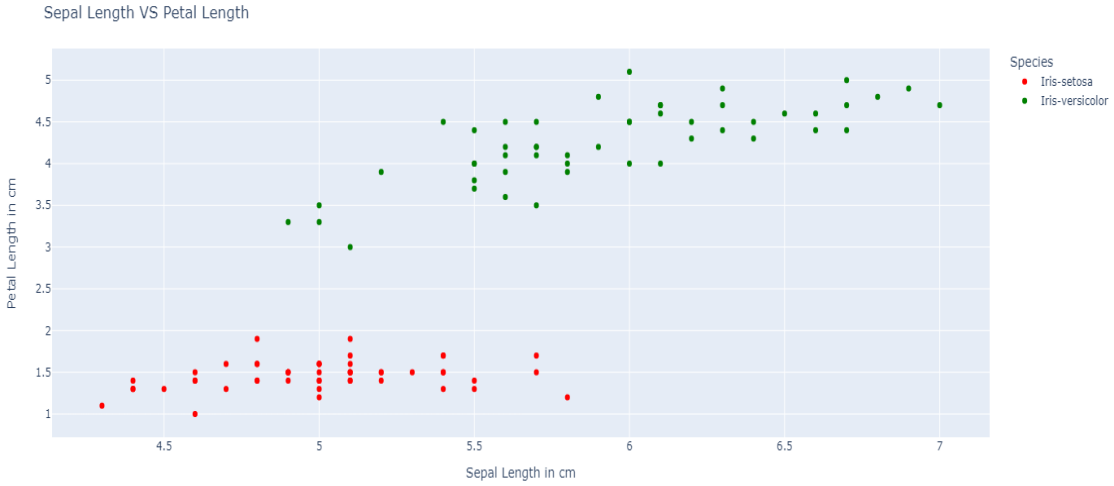


Figure 4: Petal-Sepal-length

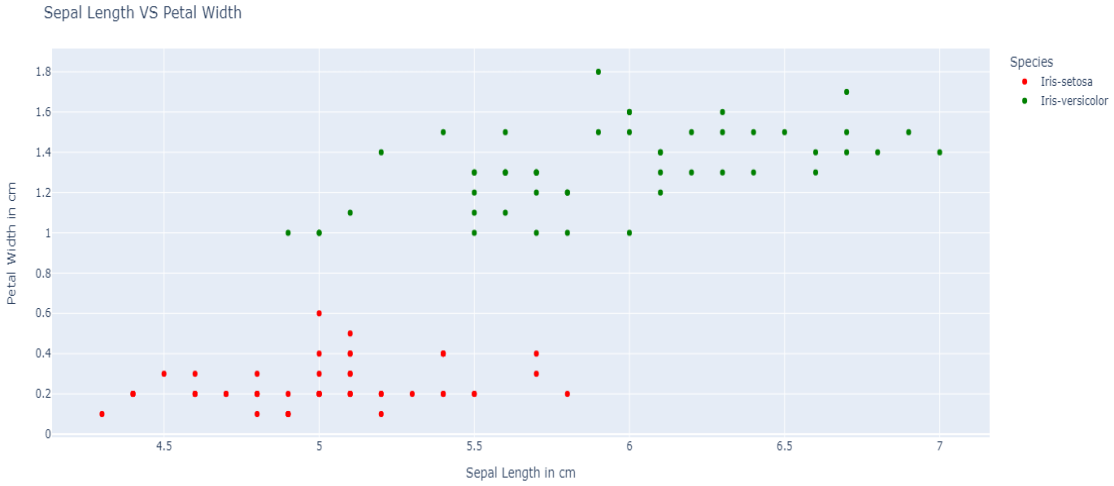


Figure 5: SepalLen-PetalWid-features

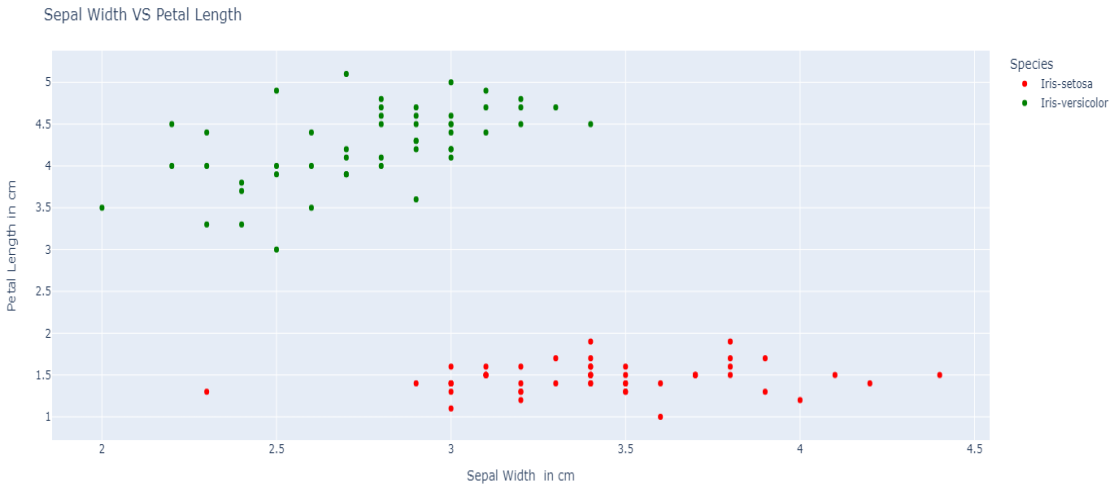


Figure 6: PetalLen-SepalWid-length

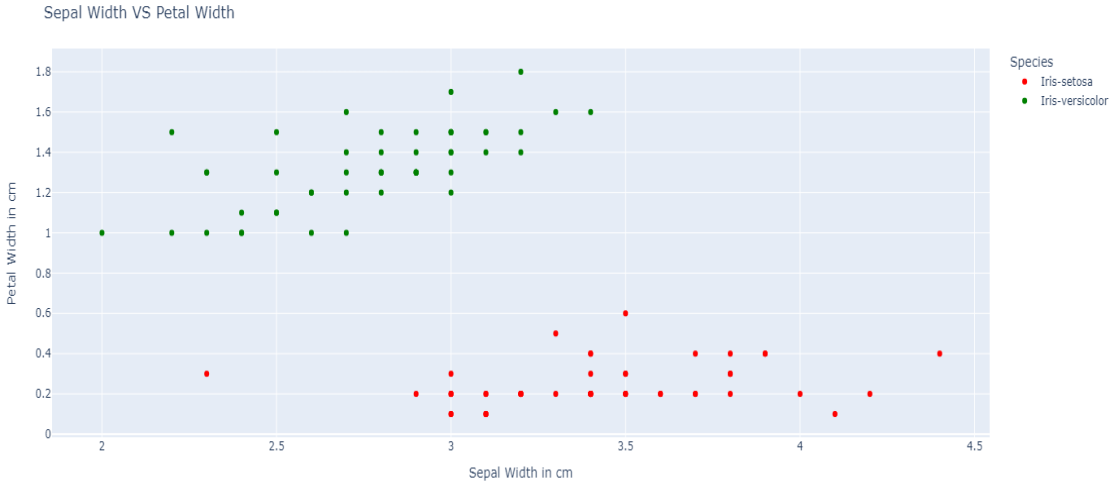


Figure 7: SepalWid-PetalWid-features

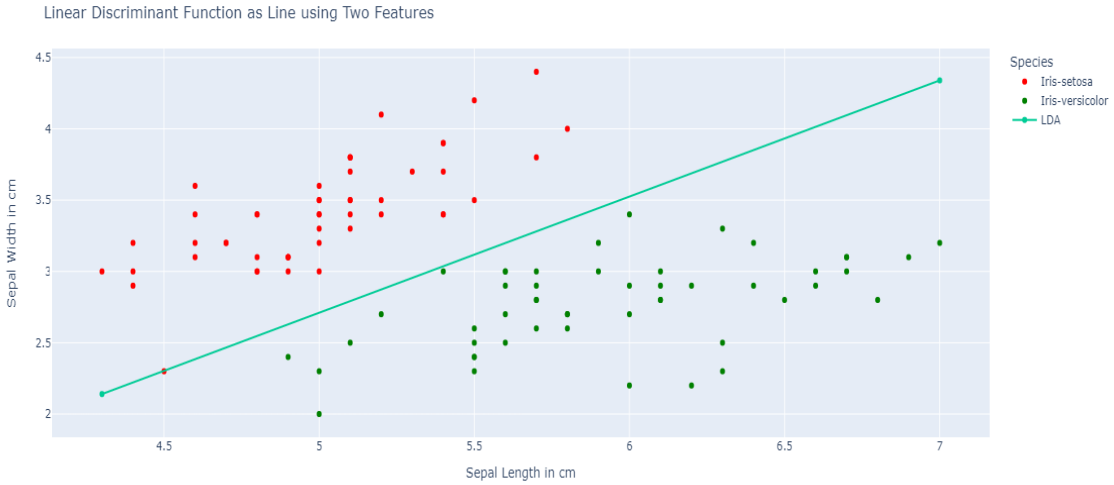


Figure 8: Scatter plot: Sepal-features

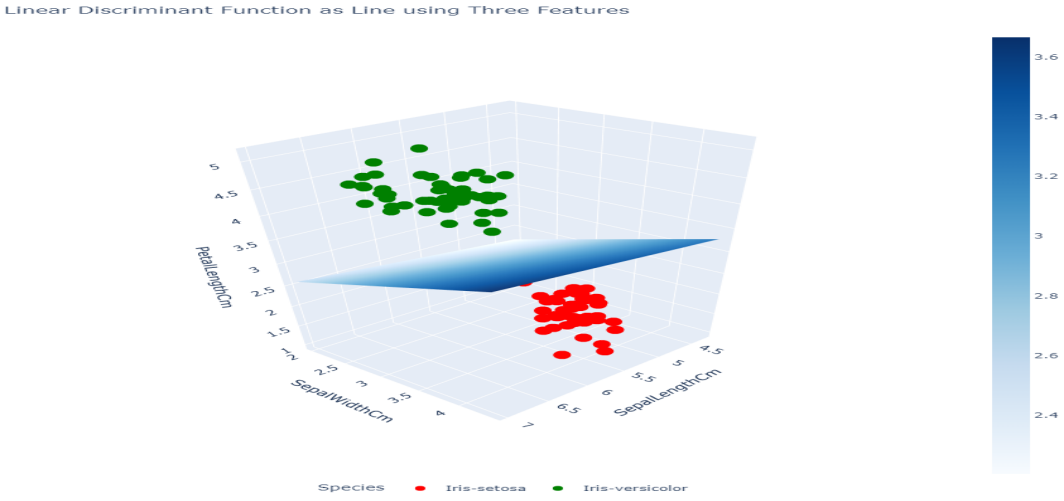


Figure 9: 3D-Plot:90

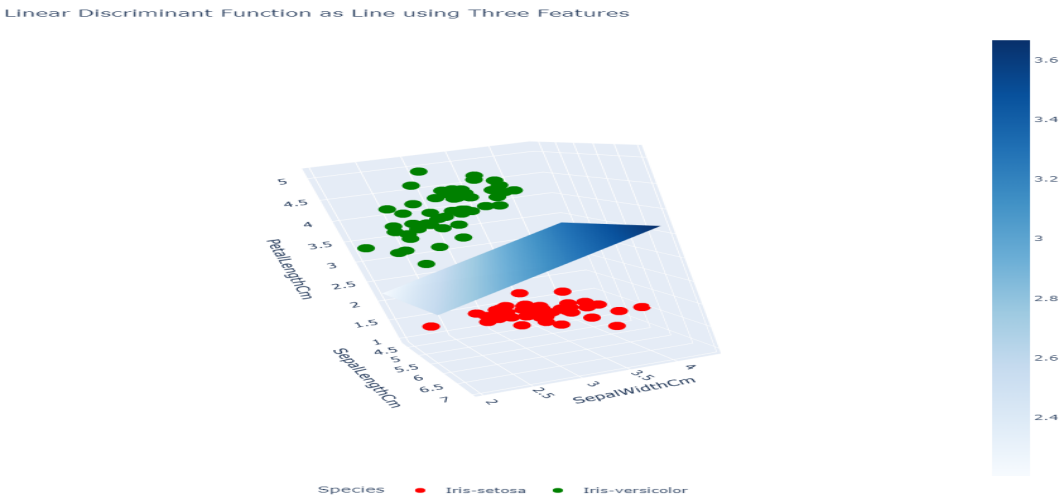


Figure 10: 3D-Plot:25

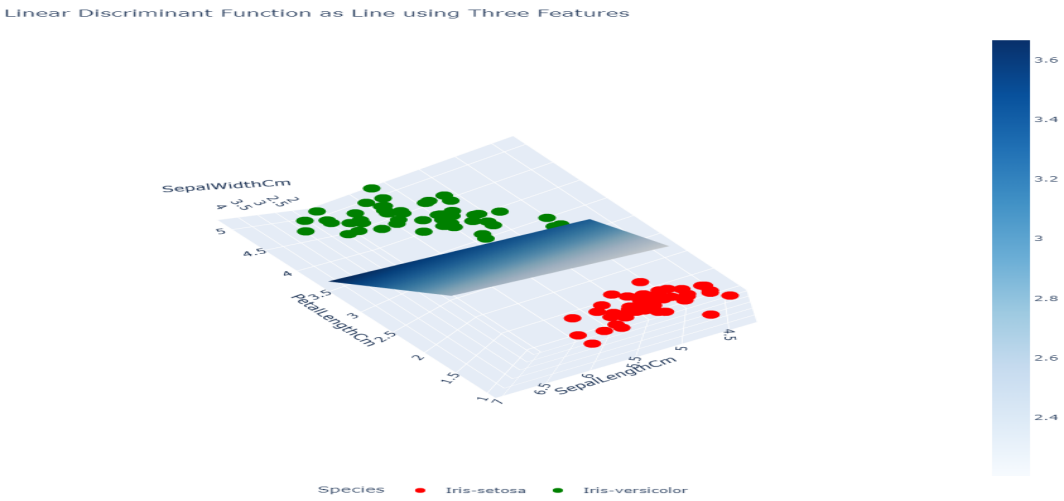


Figure 11: 3D-Plot:135

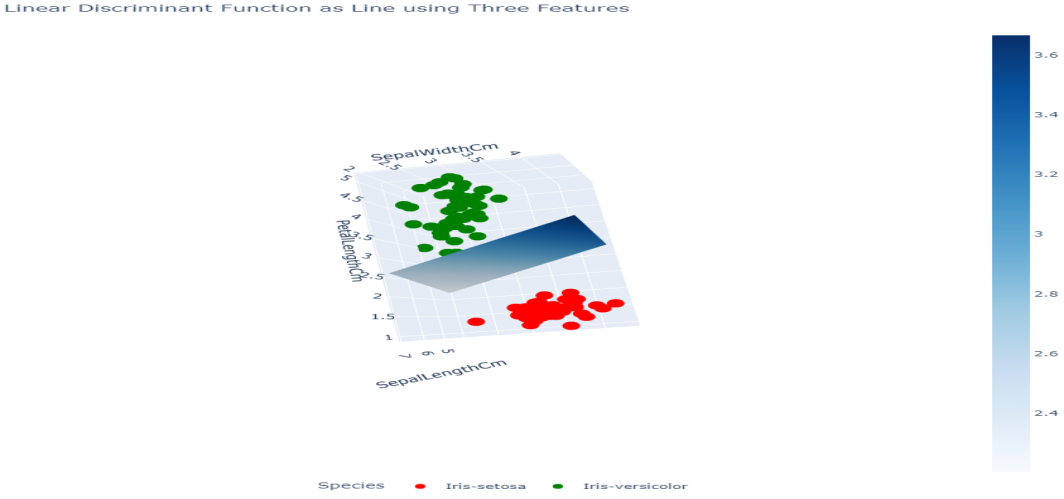


Figure 12: 3D-Plot:285

Definition

A discriminant function that is a linear combination of the components of x can be written as

$$g(x) = w^T x + w_0$$

where w is the weight vector and w_0 the basis or threshold weight.
A two-category classifier with a discriminant function of the form uses following rule:

- decide w1 if g(x) $>$ 0 and w2 if g(x) $<$ 0.
- If g(x) = 0 $\Rightarrow$ x is assigned to either class.
- The equation g(x) = 0 defines the decision surface that separates points assigned to the category w_1 from points assigned to the category w_2.
- When g(x) is linear, the decision surface is a hyperplane.
- If x_1 and x_2 are both on the decision surface, then:

$$w^T x_1 + w_0 = w^T x_2 + w_0$$
$$w^t(x_1 + x_2) = 0$$

- w is normal to any vector lying on the hyperplane.

Observation/ Justification

After visualizing the data by considering the maximum combination of two-features in a 2D-plot we can clearly see that it can be divided into 2-clusters using LDA (Ref. fig.2-fig.7).
Also we can generate a line using LDF to clearly visualise the separation.(Ref. fig.8) The 3D-plot shows the hyperplane using three features.