Guided Project : Feature Engineering - LDA

Name	Lakshmi Thirunavukkarasu
Course	AI and ML (Batch 5)
Problem Statement	Reduce the number of features from 4 to 2 for IRIS dataset to make the data linearly separable and build a classification model on the LDA dataset

Prerequisites

Software Requirements:

1. Anaconda

2. Python 3.8

3. Python Packages

- scikit-learn

- numpy

- matplotlib

Dataset Used	IRIS Dataset
Method for Dimensionality reduction	LDA
Algorithm used Classifier	Decision Tree Classifier

Step 1: Load the Dataset

In [1]:
```
import numpy as np
from sklearn import datasets
```

Load the Dataset

In [4]:
```
iris  = datasets.load_iris()
X = iris.data
y = iris.target
print("Size of the dataset", X.shape)
```

```
Size of the dataset (150, 4)
```

Display the feature names and Target class

In [ ]:
```
# input_features = iris.feature_names
print("Input Features ", input_features)
target_class = iris.target_names
print("Output Class", target_class)
```

Step 2: Visualize the data set

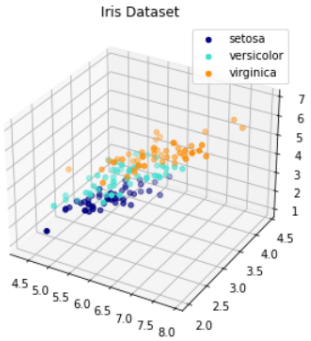
Apply LDA to reduce the number of features from 4 to 2

In [21]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components  = 2)
lda.fit(X_train, y_train)
x_transform = lda.transform(X_train)
print("Number of Features before LDA: ", X_train.shape[1])
print("Number of Features after LDA : ", x_transform.shape[1])
```

```
Number of Features before LDA:  4
Number of Features after LDA :  2
```

Plot the data

In [11]:
```
import matplotlib.pyplot as plt
colors = ['navy','turquoise','darkorange']
fig = plt.figure(figsize=(5,5))
ax  =  fig.add_subplot(111,projection= '3d')
plt.title("Iris Dataset")
for color , i, target_name in zip(colors,[0,1,2],target_class):
    ax.scatter(X[y == i, 0],X[y == i, 1],X[y == i, 2], color=color, label = target_name )
plt.legend(loc='best')
plt.show()
```



Step 3: Apply LDA to reduce the features from 4 to 2

Apply LDA to reduce the number of features from 4 to 2

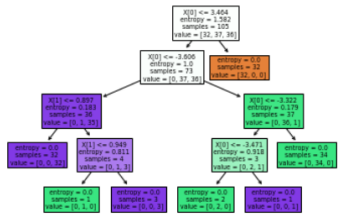
In [21]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components  = 2)
lda.fit(X_train, y_train)
x_transform = lda.transform(X_train)
print("Number of Features before LDA: ", X_train.shape[1])
print("Number of Features after LDA : ", x_transform.shape[1])
```

```
Number of Features before LDA:  4
Number of Features after LDA :  2
```

Step 4: Build a classification Model

Input LDA feature set into the Decision Tree Classifier

In [30]:
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
tree = DecisionTreeClassifier(criterion = 'entropy')
clf = tree.fit(x_transform, y_train)
plot_tree(clf, filled=True)
plt.figure(figsize = (500,500))
plt.show()
```



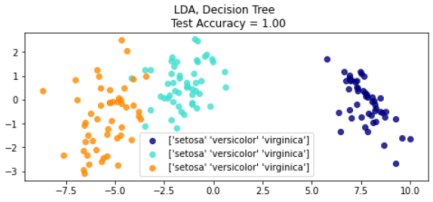
```
<Figure size 36000x36000 with 0 Axes>
```

Step 5: Validate the accuracy metrics

Plot the predicted target class using Decision Tree Algorithm

In [42]:
```
X_full_transform = lda.transform(X)
plt.figure(figsize = (8,3))
for color , i, target_name  in zip(colors, [0,1,2], target_class):
    plt.scatter(X_full_transform[y == i, 0], X_full_transform[y == i, 1], alpha = 0.8, color = color, label = target_class)

plt.legend(loc='best')
plt.title("LDA, Decision Tree \n Test Accuracy = {:.2f}".format(acc))
plt.show()
```



Build Classification Report

In [38]:
```
# compute the Decision Tree Accurancy
from sklearn.metrics import accuracy_score,confusion_matrix, classification_report
y_pred  = clf.predict(lda.transform(X_test))
accuracy = accuracy_score(y_test,y_pred)
report = classification_report(y_pred, y_test)
cm = confusion_matrix(y_test, y_pred)

print("Classification report:")
print("Accuracy: ", accuracy)
print(report)
print("Confusion matrix:")
print(cm)
```

```
Classification report:
Accuracy:  1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        14

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45

Confusion matrix:
[[18  0  0]
 [ 0 13  0]
 [ 0  0 14]]
```