## **Project Info** Creat the deciion tree classifier and visualize it graphically. • The purpose is if we feed any new data to this classifier, it would be able to predict the right class accordingly. Attribute Information: 1. Sepal Length in cm 2. Sepal Width in cm 3. Petal Length in cm 4. Petal Width in cm 5. Species: (Iris-setosa, Iris-versicolor, Iris-virginica) Import necessary libraries In [21]: **import** pandas **as** pd import numpy as np import seaborn as sns import matplotlib.pyplot as plt from sklearn.cluster import KMeans from sklearn import tree from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeClassifier import warnings warnings.filterwarnings('ignore') Load & Explore data data=pd.read\_csv('D:/Projects/GRIP/Iris.csv') In [2]: data.head() Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm **Species** 0 1 5.1 3.5 1.4 0.2 Iris-setosa 4.9 1.4 0.2 Iris-setosa **2** 3 4.7 3.2 1.3 0.2 Iris-setosa **3** 4 4.6 1.5 0.2 Iris-setosa **4** 5 5.0 3.6 1.4 0.2 Iris-setosa data.drop('Id',axis=1,inplace=True) In [4]: # To show basic info about datatype data.info() shape = data.shapeprint(f'\n Number of Rows = {shape[0]}\n Number of columns = {shape[1]} ') <class 'pandas.core.frame.DataFrame'> RangeIndex: 150 entries, 0 to 149 Data columns (total 5 columns): Non-Null Count Dtype Column # -----SepalLengthCm 150 non-null float64 SepalWidthCm 150 non-null float64 PetalLengthCm 150 non-null float64 3 PetalWidthCm 150 non-null 4 Species 150 non-null dtypes: float64(4), object(1) memory usage: 6.0+ KB Number of Rows = 150Number of columns = 5In [5]: # To display stats about the data data.describe() SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Out[5]: 150.000000 150.000000 150.000000 150.000000 count mean 5.843333 3.054000 3.758667 1.198667 0.828066 0.433594 0.763161 1.764420 std min 4.300000 2.000000 1.000000 0.100000 25% 5.100000 2.800000 1.600000 0.300000 5.800000 3.000000 4.350000 1.300000 6.400000 3.300000 5.100000 1.800000 75% 7.900000 4.400000 6.900000 2.500000 In [6]: # To display no. of samples on each class data['Species'].value\_counts() Iris-setosa Out[6]: Iris-versicolor Iris-virginica Name: Species, dtype: int64 In [7]: # Check for null values data.isnull().sum() SepalLengthCm Out[7]: SepalWidthCm 0 PetalLengthCm 0 PetalWidthCm 0 Species 0 dtype: int64 In [8]: # drop ID and Species columns iris = data.drop(['Species'], axis=1) iris.head() SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Out[8]: 1.4 0.2 4.9 3.0 1.4 0.2 1 2 4.7 3.2 1.3 0.2 3 4.6 3.1 1.5 0.2 3.6 1.4 0.2 Find the optimum number of clusters for K Means In [9]: x = iris.iloc[:, :].values WCSS = []# WCSS means >> Within cluster sum of squares **for** i **in** range(1, 21): kmean = KMeans(n\_clusters = i, init = 'k-means++', $max_iter = 200$ , $n_init = 15$ , $random_state = 0$ ) kmean.fit(x)wcss.append(kmean.inertia\_) In [10]: # Plotting the results onto a line graph, # `allowing us to observe 'The elbow' plt.plot(range(1, 21), wcss, color = "blue") plt.title('The elbow method') plt.xlabel('Number of clusters') plt.ylabel('WCSS') plt.annotate('ELBOW', xytext=(5, 200), xy=(3, 100), arrowprops={'facecolor':'green'}) plt.grid() plt.show() The elbow method 700 600 500 400 300 ELBOW 200 100 12.5 2.5 5.0 10.0 15.0 17.5 20.0 Number of clusters • You can clearly see why it is called 'The elbow method' from the above graph, the optimum clusters is where the elbow occurs. • This is when the within cluster sum of squares (WCSS) doesn't decrease significantly with every iteration. • From this we choose the number of clusters as 3 Clusters In [11]: # Applying kmeans to the dataset kmean = KMeans(n\_clusters = 3, init = 'k-means++', $max_iter = 200$ , $n_init = 15$ , $random_state = 0$ ) $y = kmean.fit_predict(x)$ In [12]: fig, ax = plt.subplots(1, 2, figsize=(12, 4))# Visualising the clusters - On the first two columns (sepal length, sepal width) ax[0].scatter(x[y == 0, 0], x[y == 0, 1],s=50, c='blue', label='Iris-setosa') ax[0].scatter(x[y == 1, 0], x[y == 1, 1],s=50, c='green', label='Iris-versicolour') ax[0].scatter(x[y == 2, 0], x[y == 2, 1],s=50, c='red', label='Iris-virginica') ax[0].scatter(kmean.cluster\_centers\_[:, 0], kmean.cluster\_centers\_[:, 1], s=80, c='black', label='Centroids') ax[0].set\_xlabel('sepal\_length') ax[0].set\_ylabel('sepal\_width') ax[0].set\_title('Sepal Length and Width') ax[0].grid() ax[0].legend() # Visualising the clusters - On the second two columns (petal length, petal width) ax[1].scatter(x[y == 0, 2], x[y == 0, 3],s=50, c='blue', label='Iris-setosa') ax[1].scatter(x[y == 1, 2], x[y == 1, 3],s=50, c='green', label='Iris-versicolour') ax[1].scatter(x[y == 2, 2], x[y == 2, 3],s=50, c='red', label='Iris-virginica') ax[1].scatter(kmean.cluster\_centers\_[:, 2], kmean.cluster\_centers\_[:, 3], s=80, c='black', label='Centroids') ax[1].set\_xlabel('petal\_length') ax[1].set\_ylabel('petal\_width') ax[1].set\_title('Petal Length and Width') ax[1].grid() ax[1].legend() plt.show() Sepal Length and Width Petal Length and Width 4.5 Iris-setosa Iris-setosa Iris-versicolour Iris-versicolour 4.0 Iris-virginica Iris-virginica 2.0 Centroids Centroids | | sepal\_width o.s petal\_width 2.5 0.5 2.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 sepal\_length petal\_length The Original data In [13]: colors = ['green', 'blue', 'red'] species = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'] In [14]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))for i in range(3): x\_sepal = data[data['Species']==species[i]] ax1.scatter(x\_sepal['SepalLengthCm'], x\_sepal['SepalWidthCm'], c = colors[i], label = species[i]) ax1.set\_xlabel('sepal\_length') ax1.set\_ylabel('sepal\_width') ax1.legend() for i in range(3): x\_petal = data[data['Species']==species[i]] ax2.scatter(x\_petal['PetalLengthCm'], x\_petal['PetalWidthCm'], c = colors[i], label = species[i]) ax2.set\_xlabel('petal\_length') ax2.set\_ylabel('petal\_width') ax2.legend() plt.show() 4.5 Iris-setosa Iris-setosa 2.5 Iris-versicolor Iris-versicolor Iris-virginica Iris-virginica 4.0 2.0 3.5 sepal\_width petal\_width 1.0 2.5 0.5 2.0 4.5 5.5 6.0 7.0 7.5 8.0 5.0 6.5 petal\_length sepal\_length In [15]: fig, (ax1, ax2) = plt.subplots(1,2, figsize=(15,6)) for i in range(3): x = data[data['Species']==species[i]] ax1.scatter(x['SepalLengthCm'], x['PetalLengthCm'], c = colors[i], label = species[i]) ax1.set\_xlabel('Petal\_length') ax1.set\_ylabel('petal\_length') ax1.legend() for i in range(3): x = data[data['Species']==species[i]] ax2.scatter(x['SepalWidthCm'], x['PetalWidthCm'], c = colors[i], label = species[i]) ax2.set\_xlabel('petal\_width') ax2.set\_ylabel('petal\_width') ax2.legend() plt.show() Iris-setosa 2.5 Iris-versicolor Iris-virginica 2.0 5 petal\_width petal\_length Iris-setosa Iris-versicolor Iris-virginica 1.0 3 2 · 0.5 7.0 7.5 4.5 5.0 6.0 6.5 8.0 2.0 2.5 3.0 3.5 4.0 Petal\_length petal\_width Coorelation Matrix data.corr() In [16]: SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Out[16]: 1.000000 0.871754 0.817954 SepalLengthCm -0.109369 SepalWidthCm -0.109369 1.000000 -0.420516 -0.356544 PetalLengthCm 0.871754 -0.420516 1.000000 0.962757 **PetalWidthCm** 0.817954 -0.356544 0.962757 1.000000 In [17]: corr = data.corr() fig, ax = plt.subplots(figsize=(10,6)) sns.heatmap(corr,annot=True, ax=ax, cmap ='coolwarm') plt.show() 1.0 SepalLengthCm 1 0.87 0.82 - 0.8 SepalWidthCm - 0.6 1 -0.42 -0.36 - 0.4 PetalWidthCm PetalLengthCm - 0.2 0.87 -0.42 0.96 - 0.0 -0.20.82 -0.36 0.96 SepalWidthCm PetalLengthCm PetalWidthCm SepalLengthCm **Model Training** In [18]: x = data.drop('Species', axis =1) y = data['Species'] In [19]: x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.3, random\_state=42) In [22]: # Decision Tree Model dtree = DecisionTreeClassifier() # model training dtree.fit(x\_train, y\_train) # print matric to get performance print('Accuracy: ', dtree.score(x\_test, y\_test)\*100,"%") Accuracy: 100.0 % features = x.columns.tolist() target = y.value\_counts().index In [24]: plt.figure(figsize=(15,10)) tree.plot\_tree(dtree, feature\_names = features, class\_names = target, filled = True) plt.title('Decision Tree Graph') plt.show() Decision Tree Graph PetalWidthCm <= 0.8 gini = 0.664samples = 105value = [31, 37, 37] class = Iris-versicolor PetalLengthCm <= 4.75 gini = 0.0 gini = 0.5samples = 31samples = 74 value = [31, 0, 0]value = [0, 37, 37]class = Iris-setosa class = Iris-versicolor PetalWidthCm <= 1.6 PetalWidthCm <= 1.75 gini = 0.059gini = 0.214 samples = 33samples = 41 value = [0, 32, 1] value = [0, 5, 36]class = Iris-virginica class = Iris-versicolor PetalLengthCm <= 4.95 PetalLengthCm <= 4.85 gini = 0.0gini = 0.0gini = 0.5 gini = 0.059samples = 32 samples = 1samples = 8samples = 33value = [0, 32, 0]value = [0, 0, 1]value = [0, 4, 4]value = [0, 1, 32]class = Iris-versicolor class = Iris-virginica class = Iris-versicolor class = Iris-virginica SepalWidthCm <= 3.1 PetalWidthCm <= 1.55 gini = 0.0 gini = 0.0gini = 0.444 gini = 0.444 samples = 2samples = 30 samples = 6 samples = 3value = [0, 2, 0]value = [0, 0, 30]value = [0, 2, 4]value = [0, 1, 2]class = Iris-versicolor class = Iris-virginica class = Iris-virginica class = Iris-virginica SepalLengthCm <= 6.95 gini = 0.0gini = 0.0 gini = 0.0gini = 0.444samples = 1samples = 3samples = 2samples = 3value = [0, 0, 3]value = [0, 0, 2]value = [0, 1, 0]value = [0, 2, 1]class = Iris-virginica class = Iris-virginica class = Iris-versicolor class = Iris-versicolor gini = 0.0gini = 0.0samples = 2 samples = 1value = [0, 2, 0]value = [0, 0, 1]class = Iris-versicolor :lass = Iris-virginica