Author : Pashanki Pandit

Data Science & Business Analytics Tasks

Task # 6 - Prediction using Decision Tree Algorithm

Graduate Rotational Internship Program(GRIP) The Sparks Foundation

In this task I Create the Decision Tree classifier and visualize it graphically.

Load the libraries in Python

```
In [162]: import pandas as pd
          import seaborn as sb
          import matplotlib.pyplot as plt
          import missingno as mn
          from sklearn.utils import resample
```

Load the data

```
In [163]: df=pd.read_csv("C:/Users/Abhijeet/Downloads/Iris (2).csv")
```

```
In [164]: df.head()
```

Out[164]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
In [165]: df.shape
```

```
Out[165]: (150, 6)
```

Preprocessing the Data

```
In [166]: from sklearn.preprocessing import LabelEncoder
          le=LabelEncoder()
          df["Species"]=le.fit_transform(df.Species)
```

```
In [167]: df.head()
```

Out[167]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

Delete the unwanted coumbs

```
In [168]: df.columns
```

```
Out[168]: Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
                 'Species'],
                dtype='object')
```

```
In [169]: df = df.drop(['Id'],axis=1)
```

```
In [170]: df.head()
```

Out[170]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
In [171]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    int64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

```
In [172]: df.describe()
```

Out[172]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667	1.000000
std	0.828066	0.433594	1.764420	0.763161	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

Insights

- from above we get that the difference between mean and medium if it is more than (>5%) it means it is not normally distributed.
- from above there is all are normally distributed.

Check the correletion

```
In [173]: df.corr()
```

Out[173]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954	0.782561
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757	0.949043
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000	0.956464
Species	0.782561	-0.419446	0.949043	0.956464	1.000000

We check the Missing values

```
In [174]: df.isna().sum()  ### all are zero means no missing values
```

```
Out[174]: SepalLengthCm    0
          SepalWidthCm     0
          PetalLengthCm    0
          PetalWidthCm     0
          Species          0
          dtype: int64
```

Divide categorical and continuos variables in 2 lists

```
In [175]: cat=[]
          con=[]
          for i in df.columns:
              if (df[i].dtype=="object"):
                  cat.append(i)
              else:
                  con.append(i)
```

```
In [176]: cat
```

```
Out[176]: []
```

```
In [177]: con
```

```
Out[177]: ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Species']
```

```
In [178]: #EDA
          import seaborn as sb
          import matplotlib.pyplot as plt
          for i in df.columns:
              if (df[i].dtype=="object"):
                  print(pd.crosstab(df.Species,df[i]))
              else:
                  sb.boxplot(df.Species,df[i])
                  plt.show()
```

Insights

-We confused from above it is the linear data

We create the DecisionTreeClassifier

```
In [128]: from sklearn.tree import DecisionTreeClassifier
          dtc=DecisionTreeClassifier()
          model=dtc.fit(strain,ytrain)
          pred=model.predict(xtest)
```

```
In [129]: from sklearn.tree import export_graphviz
          export_graphviz(dtc,out_file="C:/Users/Abhijeet/Desktop/dummy_work/e2")

          import pydotplus as pdp
          graph=pdp.graph_from_dot_file("C:/Users/Abhijeet/Desktop/dummy_work/e2")
          from IPython.display import Image
          Image(graph.create_jpg())
```

Out[129]:

You can now feed any new/test data to this classifer and it would be able to predict the right class accordingly. We shown h...

```
In [130]: X=df[['Species']]
          X=df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]

          from sklearn.model_selection import train_test_split
          xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=0.2,random_state=30)

          from sklearn.tree import DecisionTreeClassifier
          dtc=DecisionTreeClassifier()
          model=dtc.fit(xtrain,ytrain)
          pred=model.predict(xtest)

          from sklearn.metrics import accuracy_score
          print(accuracy_score(ytest,pred))
```

```
0.9666666666666667
```

Insights

- we get the very good accuracy at the randome state=30

```
In [131]: from sklearn.tree import export_graphviz
          export_graphviz(dtc,out_file="C:/Users/Abhijeet/Desktop/dummy_work/e2")

          import pydotplus as pdp
          graph=pdp.graph_from_dot_file("C:/Users/Abhijeet/Desktop/dummy_work/e2")
          from IPython.display import Image
          Image(graph.create_jpg())
```

Out[131]: