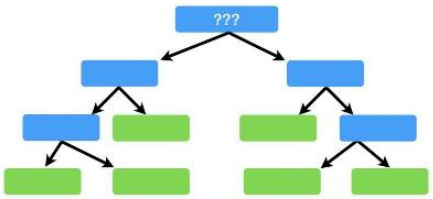
Objective : The objective of this TP is to explore the data using Python and build decision tree using Decision Tree Algorithms. This decision tree is a support tool that uses a tree-like graph or model of decisions and their possible consequences. It is one way to display an algorithm that contains only conditional control statements .



A decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems.

It works for both categorical and continuous input and output variables.

Types of Decision Trees

Types of the decision tree are based on the type of target variable we have. It can be of two types:

1. **Categorical Variable Decision Tree**
2. **Continuous Variable Decision Tree**

What is Categorical Variable Decision Tree?

A decision tree which has a categorical target variable is called categorical variable decision tree.

What is Continuous Variable Decision Tree?

A decision tree which has continuous target variable then it is called as the continuous variable decision tree.

In order to do so follow the steps below **:**

Running the Jupyter Notebook

- Files → Desktop → New → Python 3 → Folder
- Select directory → Rename Folder (change the name of the folder)
- Select the folder and create a python file.

Problem Statement:

To build a Decision Tree model for prediction of car quality given other attributes about the car.

DataSet

Car Evaluation Data Set

Data Set Characteristics:	Multivariate	**Number of Instances:**	1728	**Area:**	N/A
Attribute Characteristics:	Categorical	**Number of Attributes:**	6	**Date Donated**	1997-06-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1111870

Decision Tree Code: Implementation with Python

Importing relevant libraries:

First we will import certain libraries required for performing K-means Clustering. We will also look into these packages in details.
1. **Numpy**: It is a third party package that helps us to deal with multidimensional arrays.
2. **Pandas**: Allows us to organize data in tabular form.
3. **Matplotlib**: Helps in visualizing the numpy computation.
4. **Seaborn**: This also adds to the visualization of Matplotlib
5. **Scikit-learn**: This is the most widely used machine learning library. It has various functionality as in this case we are importing KMeans from it.

```
import os
import numpy as np
import pandas as pd
import numpy as np, pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree, metrics
```

Reading the data:

In this step we will load the data set into the variable data using pandas data frame. There are thirty observations with features satisfaction and Loyalty .The data here is in .csv format. Remember pandas helps us to organize data in tabular form.

```
data =
pd.read_csv('data/car_quality/car.data',names=['buying','maint','doors','persons
','lug_boot','safety','class'])
data.head()
```

Check a few information about the data set

```
data.info()
```

Identify the target variable

```
data['class'],class_names = pd.factorize(data['class'])
```

Let's check the encoded values now.

```
print(class_names)
print(data['class'].unique())
```

Identify the predictor variables and encode any string variables to equivalent integer codes

```
data['buying'],_ = pd.factorize(data['buying'])
data['maint'],_ = pd.factorize(data['maint'])
data['doors'],_ = pd.factorize(data['doors'])
data['persons'],_ = pd.factorize(data['persons'])
data['lug_boot'],_ = pd.factorize(data['lug_boot'])
data['safety'],_ = pd.factorize(data['safety'])
data.head()
```

Check the data types now:

```
data.info()
```

Select the predictor feature and the target variable

```
X = data.iloc[:,:-1]
y = data.iloc[:,-1]
```

Train test split:

```
# split data randomly into 70% training and 30% test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split.train_test_split(X, y,
test_size=0.3, random_state=0)
```

Training/model fitting

```
# train the decision tree
dtree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3,
random_state=0)
dtree.fit(X_train, y_train)
```

Model parameters study:

```
# use the model to make predictions with the test data
y_pred = dtree.predict(X_test)
# how did our model perform?
count_misclassified = (y_test != y_pred).sum()
print('Misclassified samples: {}'.format(count_misclassified))
accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: {:.2f}'.format(accuracy))
```

Visualization of the decision graph:

```
import graphviz
feature_names = X.columns

dot_data = tree.export_graphviz(dtree, out_file=None, filled=True, rounded=True,
                                feature_names=feature_names,
                                class_names=class_names)
graph = graphviz.Source(dot_data)
graph
```