AIM: Implementation of Naïve Bayes Classifier

THEORY:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

1) **IMPORTING LIBRARIES:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import metrics
import seaborn as sns
```

2) **DATA PREPROCESSING:**

```
Dataframe = pd.read_csv('winequalityN.csv')
# getting info.
Dataframe.info()
Dataframe.describe()
# null value check
Dataframe.isnull().sum()
Dataframe = Dataframe.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
Dataframe.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   type                  6497 non-null   object 
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

3) DATA SPLITTING INTO TRAINING DATASET & TESTING DATASET & CREATING MODEL:

```
x = Dataframe.drop(columns = ['quality','type'])
y = Dataframe['quality']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30,random_state=1)
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(x_train,y_train)
```

```
GaussianNB()
```

4) PREDICTING QUALITY OF THE WINE:

```
model.score(x_test,y_test)
y_pred = model.predict(x_test)
np.set_printoptions(threshold=np.inf)
y_pred
```

```
array([6, 6, 6, 7, 7, 6, 7, 7, 7, 6, 6, 7, 6, 5, 6, 7, 6, 5, 5, 5, 7, 5,
       6, 5, 6, 7, 5, 7, 5, 5, 7, 6, 5, 6, 6, 7, 7, 5, 6, 8, 5, 7, 7, 7,
       6, 5, 6, 7, 6, 6, 5, 4, 6, 6, 7, 7, 5, 5, 5, 6, 5, 5, 7, 7, 7, 6,
       7, 7, 6, 7, 6, 7, 5, 7, 6, 5, 6, 6, 4, 3, 7, 5, 3, 5, 6, 5, 7, 6,
       6, 5, 6, 5, 6, 7, 7, 5, 7, 6, 7, 7, 5, 6, 6, 3, 6, 5, 5, 7, 6, 6,
       5, 5, 6, 6, 5, 5, 7, 6, 7, 5, 6, 4, 6, 7, 6, 6, 6, 5, 5, 7, 5, 5,
       5, 5, 6, 4, 5, 7, 6, 7, 6, 5, 7, 6, 5, 5, 5, 5, 7, 7, 6, 7, 7, 7,
       5, 6, 6, 7, 6, 7, 5, 7, 6, 5, 7, 7, 7, 7, 7, 5, 6, 7, 7, 5, 5, 6,
       7, 5, 4, 6, 6, 5, 7, 7, 5, 6, 6, 7, 5, 5, 6, 7, 5, 5, 7, 6, 5, 5,
       5, 5, 6, 5, 7, 6, 6, 5, 6, 5, 6, 6, 5, 7, 7, 7, 6, 5, 5, 5, 5, 6,
       7, 5, 7, 7, 5, 6, 5, 7, 5, 5, 5, 6, 6, 7, 6, 6, 7, 6, 7, 6, 7, 7,
       5, 6, 5, 5, 7, 7, 8, 7, 6, 6, 6, 5, 6, 6, 7, 6, 5, 5, 5, 6, 5, 5,
       5, 6, 5, 7, 5, 5, 6, 5, 7, 7, 7, 6, 5, 5, 5, 5, 6, 7, 6, 4, 5, 7,
       5, 6, 6, 7, 6, 7, 7, 4, 7, 7, 7, 3, 6, 6, 5, 7, 7, 7, 7, 4, 6, 7,
       7, 5, 6, 5, 6, 7, 6, 6, 7, 7, 6, 5, 5, 5, 7, 7, 7, 5, 5, 6, 5, 7,
       5, 5, 5, 6, 5, 5, 5, 7, 5, 6, 4, 5, 5, 6, 7, 6, 5, 7, 5, 7, 5, 3,
       6, 7, 6, 7, 6, 6, 6, 6, 5, 5, 6, 5, 6, 6, 5, 5, 5, 7, 5, 4, 6, 7,
```

5, 5, 6, 5, 5, 5, 6, 4, 6, 3, 6, 7, 5, 6, 6, 5, 5, 5, 5, 6, 5, 6,
5, 6, 6, 5, 7, 6, 6, 4, 6, 6, 7, 5, 6, 5, 6, 4, 5, 5, 6, 5, 5, 6,
5, 5, 6, 6, 5, 6, 7, 6, 5, 5, 7, 5, 5, 6, 6, 7, 5, 5, 7, 7, 7, 7,
5, 6, 7, 6, 6, 6, 7, 6, 5, 6, 7, 5, 5, 7, 6, 5, 6, 5, 7, 6, 7, 7,
7, 4, 7, 7, 5, 6, 7, 7, 6, 5, 6, 4, 5, 5, 5, 6, 6, 5, 7, 6, 6, 6,
6, 6, 5, 5, 7, 6, 5, 6, 6, 7, 6, 5, 5, 7, 5, 5, 5, 6, 5, 5, 6, 6,
5, 6, 5, 6, 5, 7, 7, 6, 4, 6, 6, 6, 7, 6, 5, 6, 5, 7, 5, 6, 5, 6,
6, 7, 7, 6, 7, 6, 6, 6, 5, 7, 5, 5, 6, 6, 6, 5, 5, 6, 7, 7, 5, 7,
6, 4, 5, 6, 6, 7, 6, 5, 5, 7, 7, 7, 7, 5, 5, 6, 6, 4, 6, 5, 4, 5,
6, 7, 4, 7, 7, 6, 6, 7, 5, 5, 5, 7, 7, 7, 6, 5, 4, 6, 5, 5, 5, 6,
6, 5, 7, 6, 5, 5, 7, 6, 7, 7, 6, 6, 6, 6, 5, 6, 6, 7, 7, 6, 3, 7,
5, 5, 5, 7, 6, 5, 3, 5, 7, 5, 7, 6, 6, 5, 7, 7, 5, 7, 6, 5, 6, 5,
6, 5, 5, 6, 7, 4, 7, 5, 6, 6, 7, 5, 7, 6, 6, 6, 5, 5, 6, 6, 5, 5,
5, 5, 7, 7, 3, 4, 7, 6, 5, 7, 5, 5, 5, 7, 6, 7, 7, 5, 5, 6, 5, 5,
6, 6, 7, 6, 7, 6, 6, 7, 7, 7, 7, 5, 5, 5, 5, 6, 7, 7, 6, 6, 6, 5,
7, 6, 6, 7, 5, 7, 5, 6, 7, 7, 6, 7, 6, 4, 5, 3, 4, 5, 7, 6, 6, 5,
5, 7, 5, 7, 7, 5, 6, 6, 6, 5, 7, 6, 5, 5, 7, 6, 5, 6, 7, 7, 6, 5,
5, 6, 6, 5, 5, 5, 5, 7, 5, 6, 6, 6, 6, 3, 6, 7, 6, 5, 6, 6, 7, 5,
5, 7, 7, 5, 6, 5, 5, 3, 4, 7, 5, 7, 7, 7, 6, 7, 7, 6, 4, 6, 6, 7,
5, 6, 6, 5, 7, 5, 5, 7, 6, 6, 5, 5, 7, 5, 7, 5, 7, 7, 6, 3, 6, 6,
7, 5, 6, 7, 5, 7, 5, 7, 6, 5, 4, 5, 5, 7, 5, 6, 6, 3, 4, 7, 6, 6,
5, 4, 6, 7, 5, 5, 6, 7, 7, 7, 6, 7, 7, 5, 5, 4, 7, 6, 6, 6, 7, 7,
7, 6, 5, 6, 6, 5, 6, 5, 6, 5, 6, 6, 5, 7, 6, 6, 5, 7, 5, 7, 6, 7,
5, 7, 5, 7, 5, 5, 6, 7, 7, 6, 5, 6, 5, 6, 5, 4, 6, 7, 7, 5, 6, 5,
6, 5, 6, 6, 6, 6, 6, 7, 6, 6, 6, 4, 6, 6, 5, 5, 6, 4, 6, 6, 7, 5,
6, 6, 7, 5, 7, 6, 7, 7, 6, 7, 5, 7, 6, 5, 7, 7, 7, 7, 5, 6, 5, 7,
5, 6, 6, 7, 5, 7, 7, 7, 6, 5, 5, 5, 7, 5, 7, 6, 6, 7, 7, 6, 6, 5,
6, 5, 5, 6, 7, 7, 7, 9, 6, 5, 6, 6, 6, 5, 7, 5, 6, 6, 6, 5, 5, 6,
5, 7, 7, 4, 7, 6, 7, 6, 7, 5, 6, 6, 5, 6, 5, 7, 5, 7, 5, 5, 6, 7,
5, 6, 7, 6, 5, 6, 5, 6, 6, 5, 7, 5, 5, 6, 6, 6, 5, 7, 5, 6, 5, 6,
5, 7, 6, 6, 7, 6, 4, 4, 4, 6, 7, 6, 5, 6, 7, 6, 5, 7, 6, 7, 5, 5,
6, 5, 7, 5, 7, 7, 7, 7, 6, 6, 7, 7, 6, 5, 7, 6, 7, 7, 6, 6, 5, 5,
5, 6, 6, 7, 7, 5, 6, 7, 6, 6, 7, 7, 7, 6, 6, 5, 5, 7, 7, 5, 5, 7,
7, 5, 6, 5, 6, 5, 7, 5, 6, 6, 6, 6, 6, 5, 5, 4, 4, 6, 6, 6, 6, 7,
6, 6, 7, 7, 5, 7, 6, 6, 4, 4, 5, 5, 6, 6, 5, 8, 5, 4, 5, 6, 5, 8,
5, 4, 6, 3, 7, 6, 6, 6, 4, 5, 7, 5, 7, 7, 7, 6, 6, 6, 5, 6, 6, 6,
6, 6, 7, 7, 5, 7, 5, 5, 6, 4, 5, 7, 7, 6, 5, 6, 6, 6, 6, 7, 6, 6,
4, 7, 7, 5, 5, 7, 7, 5, 5, 5, 8, 6, 5, 7, 6, 5, 5, 6, 7, 6, 7, 6,
5, 4, 7, 5, 7, 5, 5, 6, 6, 5, 6, 7, 5, 5, 5, 5, 7, 6, 5, 5, 6, 6,
5, 5, 6, 5, 5, 5, 7, 7, 6, 5, 5, 6, 5, 5, 7, 5, 7, 6, 6, 6, 5, 6,
5, 7, 5, 5, 6, 5, 6, 7, 7, 7, 6, 7, 6, 6, 5, 7, 7, 6, 6, 5, 7, 5,
6, 5, 7, 7, 6, 5, 5, 6, 7, 6, 6, 6, 5, 5, 5, 5, 5, 7, 7, 7, 6, 5,
5, 6, 5, 5, 6, 5, 7, 7, 6, 5, 6, 7, 7, 6, 6, 6, 5, 5, 5, 6, 5, 6,
6, 6, 7, 6, 5, 6, 6, 3, 7, 6, 5, 6, 5, 6, 6, 6, 6, 7, 6, 7, 5, 7,
6, 5, 6, 5, 5, 6, 6, 5, 6, 6, 7, 7, 5, 5, 5, 5, 6, 5, 5, 7, 6, 5,
5, 5, 6, 5, 5, 6, 5, 4, 7, 5, 6, 5, 4, 6, 5, 7, 6, 5, 6, 5, 5, 6,
5, 7, 5, 6, 7, 5, 7, 7, 7, 4, 7, 5, 5, 5, 6, 7, 6, 7, 5, 5, 5, 7,
7, 6, 6, 6, 6, 6, 6, 5, 5, 6, 5, 7, 5, 6, 6, 6, 7, 7, 7, 5, 5, 7,
7, 7, 6, 6, 7, 6, 7, 5, 5, 6, 7, 7, 5, 6, 6, 6, 5, 6, 7, 5, 6, 6,
7, 5, 7, 7, 5, 5, 6, 5, 7, 6, 6, 5, 6, 6, 5, 5, 7, 6, 5, 7, 7, 5,
7, 7, 7, 5, 7, 7, 7, 6, 6, 5, 5, 7, 7, 7, 5, 5, 6, 6, 5, 7, 5, 7,

7, 4, 5, 6, 6, 6, 5, 5, 6, 7, 6, 5, 6, 6, 6, 5, 3, 6, 7, 6, 7, 9,
6, 7, 5, 5, 6, 5, 6, 7, 3, 7, 4, 7, 5, 5, 5, 5, 5, 6, 7, 5, 7, 6,
4, 7, 6, 6, 6, 7, 7, 5, 5, 5, 5, 5, 7, 5, 6, 6, 7, 5, 5, 5, 5, 5,
7, 5, 6, 7, 6, 5, 6, 5, 5, 6, 7, 7, 5, 7, 5, 6, 6, 6, 5, 7, 6, 5,
7, 7, 5, 5, 5, 7, 5, 5, 6, 6, 5, 7, 5, 7, 7, 6, 6, 7, 6, 5, 5, 5,
7, 5, 5, 6, 7, 7, 7, 7, 6, 5, 7, 5, 5, 5, 7, 5, 7, 7, 7, 6, 7, 7,
7, 7, 5, 5, 5, 6, 6, 7, 3, 6, 7, 5, 6, 6, 7, 7, 4, 5, 7, 6, 6, 5,
5, 7, 6, 5, 5, 5, 5, 6, 6, 5, 6, 6, 5, 5, 5, 6, 5, 7, 7, 7, 5, 6,
7, 6, 4, 5, 6, 5, 6, 7, 7, 6, 5, 6, 6, 6, 5, 6, 5, 7, 6, 5, 5, 5,
7, 5, 6, 6, 6, 5, 6, 5, 5, 5, 5, 5, 5, 6, 5, 5, 7, 5, 5, 6, 6, 6,
7, 7, 5, 5, 6, 6, 7, 7, 6, 6, 6, 7, 6, 6, 7, 6, 7, 7, 6, 6, 7, 5,
5, 5, 5, 5, 7, 5, 5, 6, 5, 4, 7, 6, 5, 6, 6, 5, 5, 5, 5, 5, 5, 7,
6, 7, 7, 6, 7, 5, 6, 7, 7, 3, 7, 5, 6, 6, 7, 6, 6, 6, 6, 5, 4, 7,
7, 5, 7, 6, 6, 6, 6, 7, 5, 7, 5, 7, 6, 5, 5, 6, 6, 5, 5, 7, 6, 7,
5, 5, 5, 6, 6, 7, 5, 5, 6, 4, 6, 6, 7, 7, 5, 6, 4, 7, 7, 7, 6, 6,
7, 7, 7, 5, 6, 7, 7, 6, 7, 5, 5, 5, 6, 6, 7, 7, 6, 5, 7, 6, 6, 7,
7, 3, 7, 5, 7, 6, 7, 6, 6, 5, 6, 6, 7, 4, 6, 6, 9, 5, 6, 5, 5, 5,
6, 5, 7, 5, 4, 4, 6, 5, 5, 6, 7, 6, 7, 4, 5, 5, 5, 5, 5, 6, 6, 5,
7, 6, 7, 5, 6, 5, 5, 7, 6, 6, 6, 7, 6, 7, 5, 6, 6, 7, 6, 5, 6, 7,
6, 5, 7, 5, 7, 5, 6, 7, 5, 7, 7, 6, 7, 7, 5, 6, 6, 5, 5, 7, 7, 7,
7, 6, 5, 5, 7, 6, 5, 5, 7, 5, 5, 5, 6, 6])

5) **CONFUSION MATRIX:**

metrics.confusion_matrix(y_test,y_pred)

```
array([[  0,   2,   2,   1,   0,   0,   0],
       [  0,   4,  30,  16,  10,   0,   0],
       [  8,  26, 353, 207,  53,   0,   0],
       [  9,  26, 238, 319, 245,   0,   0],
       [  3,   6,  30, 105, 194,   3,   3],
       [  1,   0,   3,  12,  38,   2,   0],
       [  0,   0,   0,   0,   1,   0,   0]])
```

CONCLUSION:

From this practical, I have learned the implementation of naïve bayes classifier in python.

AIM: Implementation of ID3

THEORY:

ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step.
Invented by Ross Quinlan, ID3 uses a top-down greedy approach to build a decision tree. In simple words, the top-down approach means that we start building the tree from the top and the greedy approach means that at each iteration we select the best feature at the present moment to create a node. Most generally ID3 is only used for classification problems with nominal features only.

1) IMPORTING LIBRARIES:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.preprocessing import LabelEncoder #if data is string thn use it to preprocess the data
from sklearn.tree import DecisionTreeClassifier
```

2) READING DATASET:

```
df=pd.read_csv('glass.csv')
df
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

3) EXTRACTING FEATURES:

```
x=df.iloc[:,:-1]
```

```
y=df.iloc[:,9]
```

4) **MAKING MODEL:**

```
dt=DecisionTreeClassifier(criterion="entropy")
dt
```

5) **SPLITTING DATASET INTO TRAINING , TESTING & PREDICTING VALUES:**

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = .3, random_state=0)
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
y_pred
```

```
array([7, 1, 2, 6, 5, 2, 1, 2, 2, 2, 1, 1, 2, 2, 2, 7, 3, 2, 3, 2, 5, 1,
       7, 7, 7, 1, 7, 1, 1, 2, 1, 3, 2, 7, 2, 1, 1, 3, 1, 7, 2, 6, 2, 1,
       2, 1, 1, 2, 1, 2, 1, 7, 7, 1, 2, 1, 1, 2, 1, 3, 1, 1, 1, 6, 1])
```

6) **CONFUSION MATRIX:**

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[17  3  1  0  0  0]
 [ 6 15  1  0  1  3]
 [ 2  2  3  0  0  0]
 [ 0  0  0  2  0  0]
 [ 0  0  0  0  2  0]
 [ 0  0  0  0  0  7]]
```

CONCLUSION:

From this practical, I have learned about implementation of ID3 algorithm in python.

AIM: Implementation of C4.5

THEORY:

The C4.5 algorithm is used in Data Mining as a Decision Tree Classifier which can be employed to generate a decision, based on a certain sample of data (univariate or multivariate predictors). C4.5 is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. C4.5 converts the trained trees (i.e. the output of the ID3 algorithm) into sets of if-then rules. This accuracy of each rule is then evaluated to determine the order in which they should be applied. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it.

SOURCE CODE:

1) INSTALLING CHEFBOOST FOR C4.5:

```
!pip install chefboost
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting chefboost
  Downloading chefboost-0.0.17-py3-none-any.whl (26 kB)
Requirement already satisfied: pandas>=0.22.0 in /usr/local/lib/python3.7/dist-packages (from chefboost) (1.3.5)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/dist-packages (from chefboost) (1.21.6)
Requirement already satisfied: psutil>=5.4.3 in /usr/local/lib/python3.7/dist-packages (from chefboost) (5.4.8)
Requirement already satisfied: tqdm>=4.30.0 in /usr/local/lib/python3.7/dist-packages (from chefboost) (4.64.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.22.0->chefboost
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.22.0->chefboost) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=0.22.0-
Installing collected packages: chefboost
Successfully installed chefboost-0.0.17
```

2) IMPORTING LIBRARIES AND READING DATA:

```
import pandas as pd
data = pd.read_csv('https://raw.githubusercontent.com/serengil/chefboost/master/tests/dataset/golf.txt')
data
```

	Outlook	Temp.	Humidity	Wind	Decision
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes

3) BUILDING MODEL:

```
from chefboost import Chefboost as chef
config = {'algorithm': 'C4.5'}
model = chef.fit(data, config = config, target_label = 'Decision')
```

```
[INFO]:  1 CPU cores will be allocated in parallel running
C4.5  tree is going to be built...
-------------------------
finished in  0.5118496417999268  seconds
-------------------------
Evaluate  train set
-------------------------
Accuracy:  100.0 % on  14  instances
Labels:  ['No' 'Yes']
Confusion matrix:  [[5, 0], [0, 9]]
Precision:  100.0 %, Recall:  100.0 %, F1:  100.0 %
```

4) PREDICTING VALUES:


```
for i in range(data.shape[0]):
  prediction = chef.predict(model, param = data.iloc[i])
  print(prediction)
```

```
No
No
Yes
Yes
Yes
No
Yes
No
Yes
Yes
Yes
Yes
Yes
No
```

CONCLUSUION:

From this practical, I have learned the implementation of C4.5 in python.