Algorithm:

1. Start
2. Import pandas library
3. Create pandas series with values stored in a variable a and the index 'x', 'y', 'z'.
4. Create another pandas series from list colors and dictionary booking.
5. Print the series
6. Create a dictionary mydataset with mixed datatypes
7. Create a Pandas dataframe named dataf from the dictionary
8. Print the dataframe
9. Read the csv file 'food.csv'
10. Print the type, info, head(5), tail(2), shape ~~and describe~~ of csv file.
11. Use describe() function and iloc function to to slice the table.
12. Read another csv file 'food_cleaning.csv'
13. Use dropna() method to remove ~~null~~ row with null values.
14. Check whether there are any duplicates using duplicated().any() function and delete ~~the~~ duplicates using drop_duplicates() function.

Data handling using pandas

```
import pandas as pd
a = [1, 2, 3]
print (pd.Series(a, index =['x', 'y', 'z']))
colors = ['red', 'blue', 'green', 'yellow']
print (pd.Series (colors, index =[1, 2, 3, 4]))
booking = {"day1": 100, "day2": 200, "day3": 300}
bid = pd.Series(booking)
print(bid)
booking = {"day1": [100, 300], "day2": 200, "day3": 300}
bid = pd.Series(booking)
print(bid)

mydataset = {'names' : ['anu', 'ajmal', 'nihal'], 'marks':[30,78,45]}
dataf = pd.DataFrame(mydataset, index =[1, 2, 3])
print(dataf)
```

Read csv file

```
import pandas as pd
pf = pd.read_csv('food.csv')
print(type(pf))
print(pf)

print(pf.info())
```

15. Import datasets from sklearn python library
16. Load iris dataset
17. Print description, feature names and target names of iris dataset
18. Stop

Output :-

Series

```
x    1
y    2
z    3
dtype: int64

1         red
2        blue
3       green
4      yellow
dtype: object

day1    100
day2    200
day3    300
dtype: int64
day1    [100, 300]
day2           200
day3           300
dtype: object
```

Dataframe

	names	marks
1	anu	30
2	ajmal	78
3	nihal	45


```
print(pf.head(5))
print(pf.tail(2))
print(pf.shape)
print(pf.describe())
print(pf.iloc[2:, :3])

df = pd.read_csv('food_cleaning.csv')
print(df)

data = df.dropna()
print(data)

df.duplicated().any()
df.drop_duplicates(inplace = True)
print(df)

from sklearn import datasets
iris = datasets.load_iris()
print(iris.DESCR)

print(iris.feature_names)
print(iris.target_names)
```

head(5)

	Ingredient	Sweetness	Crunchiness	FoodType
0	apple	10	9	fruit
1	bacon	1	4	protein
2	banana	10	1	fruit
3	carrot	7	10	vegetable
4	celery	3	10	vegetable

tail(2)

	Ingredient	Sweetness	Crunchiness	FoodType
13	pear	10	7	fruit
14	shrimp	2	3	protein

Shape

(15, 4)

describe()

	Sweetness	Crunchiness
count	15.000000	15.000000
mean	4.733333	5.600000
std	3.514595	3.290679
min	1.000000	1.000000
25%	2.000000	3.000000
50%	3.000000	6.000000
75%	7.500000	8.500000
max	10.000000	10.000000

iloc

	Ingredient	Sweetness	Crunchiness
2	banana	10	1
3	carrot	7	10
4	celery	3	10
5	cheese	1	1
6	cucumber	2	8
7	fish	3	1
8	grape	8	5
9	green bean	3	7
10	lettuce	1	9
11	nuts	3	6
12	orange	7	3
13	pear	10	7
14	shrimp	2	3

food_cleaning.csv

	Ingredient	Sweetness	Crunchiness	FoodType
0	apple	10	9	fruit
1	bacon	1	4	protein
2	banana	10	1	fruit
3	carrot	7	10	vegetable
4	celery	NaN	10	vegetable
5	cheese	1	1	protein
6	cucumber	2	8	vegetable
7	fish	3	1	123
8	grape	8	5	fruit
9	green bean	3	7	vegetable
10	lettuce	AA	9	vegetable
11	nuts	3	6	protein
12	orange	7	3	fruit
13	pear	10	7	fruit
14	shrimp	2	3	protein
15	shrimp	2	3	protein
16	banana	10	1	fruit

dropna()

	Ingredient	Sweetness	Crunchiness	FoodType
0	apple	10	9	fruit
1	bacon	1	4	protein
2	banana	10	1	fruit
3	carrot	7	10	vegetable
5	cheese	1	1	protein
6	cucumber	2	8	vegetable
7	fish	3	1	123
8	grape	8	5	fruit
9	green bean	3	7	vegetable
10	lettuce	AA	9	vegetable
11	nuts	3	6	protein
12	orange	7	3	fruit
13	pear	10	7	fruit
14	shrimp	2	3	protein
15	shrimp	2	3	protein
16	banana	10	1	fruit

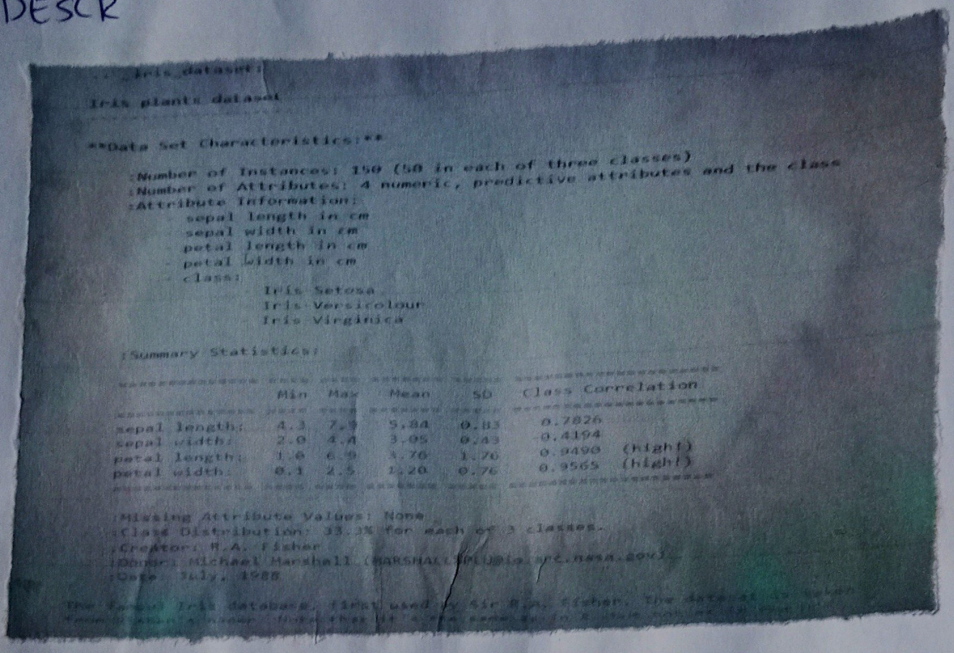
dupilated().any()

True

drop_duplicates()

	Ingredient	Sweetness	Crunchiness	FoodType
0	apple	10	9	fruit
1	bacon	1	4	protein
2	banana	10	1	fruit
3	carrot	7	10	vegetable
4	celery	NaN	10	vegetable
5	cheese	1	1	protein
6	cucumber	2	8	vegetable
7	fish	3	1	123
8	grape	8	5	fruit
9	green bean	3	7	vegetable
10	lettuce	AA	9	vegetable
11	nuts	3	6	protein
12	orange	7	3	fruit
13	pear	10	7	fruit
14	shrimp	2	3	protein

iris.DESCR



```
.. _iris_dataset:

Iris plants dataset

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
```

feature names and target names

```
print(iris.feature_names)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
print(iris.target_names)
['setosa' 'versicolor' 'virginica']
```

Algorithm

1. Start
2. import KNeighborsClassifier from sklearn.neighbors
3. import accuracy_score, confusion_matrix and ConfusionMatrixDisplay from sklearn.metrics
4. import tts train_test_split from sklearn.model_selection and LabelEncoder from sklearn.preprocessing
5. import pandas and matplotlib.pyplot
6. Read a csv file iris
7. Print the info(), head(), tail(), shape and describe() functions
8. Use dropna() method to remove rows with empty values.
9. Check whether there is any duplicates and remove it using drop_duplicates()
10. Using LabelEncoder() function transform the target values into corresponding numeric values
11. Split the variables y and zn into training data and testing data
12. Create a KNeighborsClassifier object with n_neighbors = 5.
13. Using fit function generate the model of training data
14. Using predict function make prediction on test data ytest.

Program to implement KNN classification using any standard dataset available in public domain and find accuracy of the algorithm.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
 ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import matplotlib.pyplot as plt

x=pd.read_csv('iris.csv')
print('---INFO---')
print(x.info())
print('---HEAD---')
print(x.head(5))
print('---TAIL---')
print(x.tail(4))
print('---SHAPE---')
print(x.shape)
print('---DESCRIBE---')
print(x.describe())

data=x.dropna()
print(data)
```

15. Print test class and predict class
16. Print the accuracy score of test data and predicted data ~~and use~~
17. Use predict_proba method to get the probability for test data
18. Print the confusion matrix
19. Stop

Output:-

info()

```
---INFO---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   sepal.length    150 non-null    float64
 1   sepal.width     150 non-null    float64
 2   petal.length    150 non-null    float64
 3   petal.width     150 non-null    float64
 4   variety         150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

head(5)

```
---HEAD---
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
```

tail(6)

```
---TAIL---
     sepal.length  sepal.width  petal.length  petal.width    variety
146           6.3          2.5           5.0          1.9  Virginica
147           6.5          3.0           5.2          2.0  Virginica
148           6.2          3.4           5.4          2.3  Virginica
149           5.9          3.0           5.1          1.8  Virginica
```

shape

```
---SHAPE---
(150, 5)
```



```
x.duplicated().any()
x.drop_duplicates(inplace=True)
print(x)

y = x.iloc[:, 0:4].values
z = x.iloc[:, 4].values

le = LabelEncoder()
zn = le.fit_transform(z)
print(zn)

ytrain, ytest, ztrain, ztest = train_test_split(y, zn, test_size=0.2,
        random_state=25)
print(ztest)

Knn = KNeighborsClassifier(n_neighbors=5)
Knn.fit(ytrain, ztrain)
z_predict = Knn.predict(ytest)
print("----Test class----\n", ztest)
print("----Predict Class----\n", z_predict)
print("Accuracy:", accuracy_score(ztest, z_predict, normalize=
                                  False))
print(Knn.predict_proba(ytest))

cm = confusion_matrix(ztest, z_predict, normalize=None,
        labels=Knn.classes_)
print(cm)
cm_dis = ConfusionMatrixDisplay(confusion_matrix=cm)
cm_dis.plot()
plt.show()
```

describe()

---DESCRIBE---

	sepal.length	sepal.width	petal.length	petal.width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

dropna()

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
..	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

[150 rows x 5 columns]

duplicated.any()

True

drop duplicates()

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
..	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

[149 rows x 5 columns]

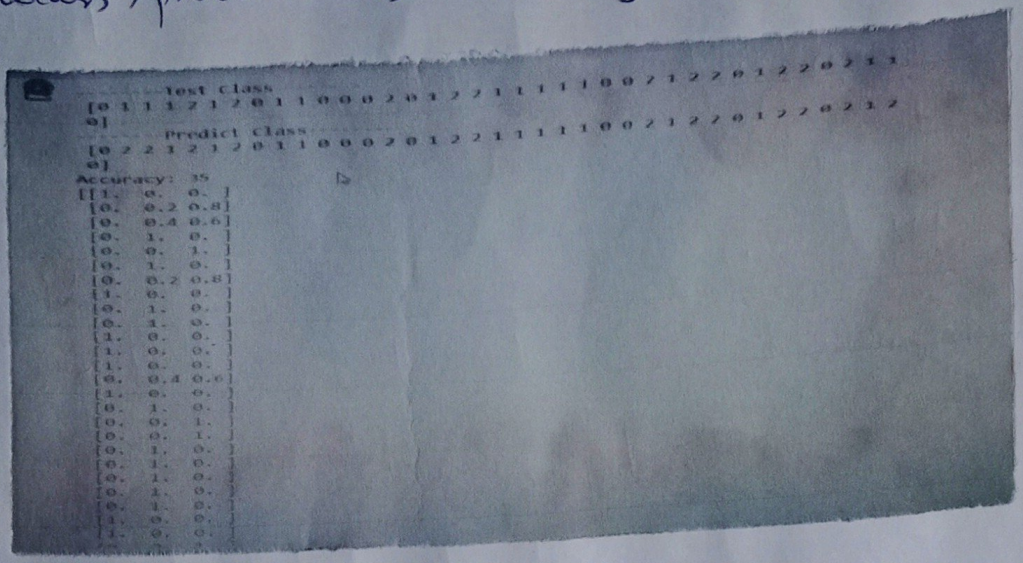
LabelEncoder()

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2]
```

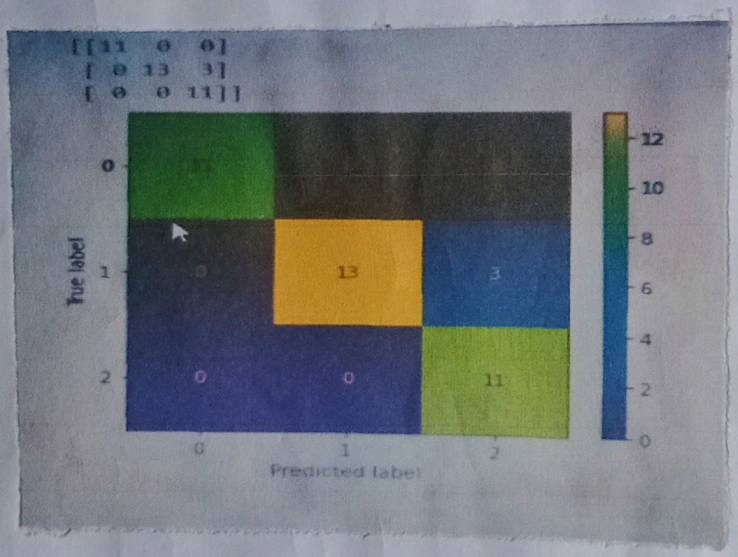
train_test_split()

```
ytrain,ytest,ztrain,ztest=train_test_split(y,zn,test_size=.25,random_stat
print(ztest)

[2 0 1 1 2 1 1 0 1 2 0 0 0 1 2 1 1 2 2 1 2 1 0 0 0 2 2 2 0 2 1 2 0 0 2 1 1
 0]
```

testclass, predict class, accuracy



confusion_matrix()



Algorithm

1. Start
2. import GaussianNB from sklearn.naive_bayes
3. import accuracy_score, confusion_matrix, ConfusionMatrixDisplay from sklearn.metrics
4. import train_test_split from sklearn.model_selection
5. import pandas as pd
6. import datasets from sklearn
7. Load iris dataset and store it in variable iris
8. print the description, feature names and target names
9. Use train_test_split function to split v and u into training data and testing data
10. Create a GaussianNB object and store it in variable NB
11. Use fit method to generate the model for training data
12. Use predict method to make predictions on test data 'utest'
13. Print the test class and predict and accuracy score by comparing the test data and predicted data.

6. Program to implement Naive Bayes algorithm using any standard dataset available in public domain and find the accuracy of the algorithm

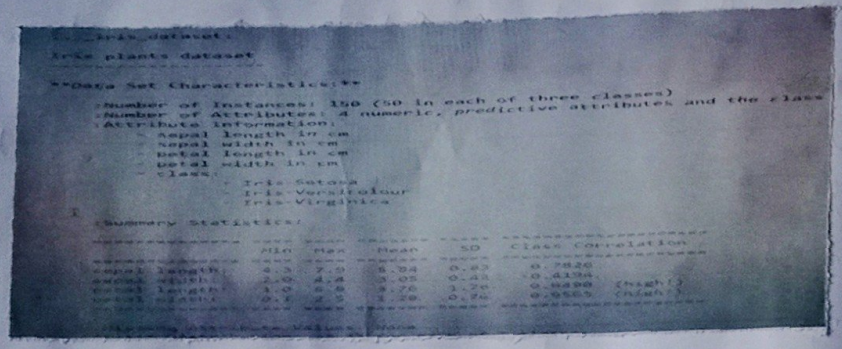
```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
    ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import sklearn import datasets

iris = datasets.load_iris()
u = iris.target
v = iris.data

print(iris.DESCR)
print(iris.feature_names)
print(iris.target_names)
vtrain, vtest, utrain, utest = train_test_split(v,u,
                                test_size = .25, random_state
                                =25)

print(utest)

NB = GaussianNB
NB.fit(vtrain, utrain)
u_predict = NB.predict(vtest)
```

14. Plot the confusion matrix

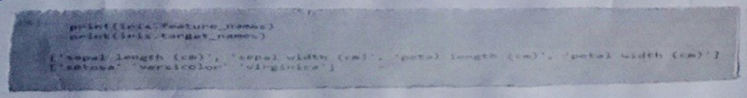
15. Stop

Output :-

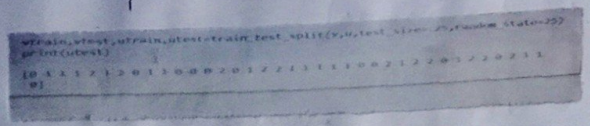
iris.DESCR



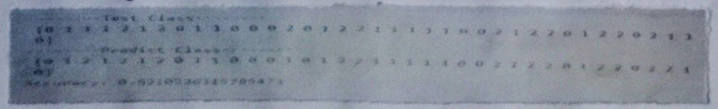
feature names and target names



train_test_split()



Accuracy_score



```
print("--- Test Class ----\n", utest)
print("---- Predict Class ----\n", u_predict)
print("Accuracy:", accuracy_score(utest, u_predict,
                                  normalize=True))

cm = confusion_matrix(utest, u_predict, normalize=None)
print(cm)
cm_dis = ConfusionMatrixDisplay(confusion_matrix=cm)
cm_dis.plot()
plt.show()
```

confusion_matrix