```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB,GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,accuracy_score,classification_repor
from sklearn.preprocessing import StandardScaler,MinMaxScaler,LabelEncoder
```

```
df=pd.read_csv('/content/food_ingredients_and_allergens.csv')
df.head()
```

	Food Product	Main Ingredient	Sweetener	Fat/Oil	Seasoning	Allergens	Prediction
0	Almond Cookies	Almonds	Sugar	Butter	Flour	Almonds, Wheat, Dairy	Contains
1	Almond Cookies	Almonds	Sugar	Butter	Flour	Almonds, Wheat, Dairy	Contains
	Chicken					Chicken,	

```
df.tail()
```

	Food Product	Main Ingredient	Sweetener	Fat/Oil	Seasoning	Allergens	Predict
394	Lemon Bars	Lemon juice	Sugar	Butter	Flour, eggs	Wheat, Dairy, Eggs	Conta
395	Pecan Pie	Pecans	Sugar	Butter	Corn syrup	Wheat, Dairy, Nuts	Conta
						Wheat,	

```
df.drop_duplicates(inplace=True)
```

```
df.isna().sum()
```

```
Food Product       0
Main Ingredient    0
Sweetener          0
Fat/Oil            0
Seasoning          0
Allergens          0
Prediction         1
dtype: int64
```

```
md=df['Prediction'].mode()[0]
```

```
df.isna().sum()
```

```
Food Product       0
Main Ingredient    0
Sweetener          0
Fat/Oil            0
Seasoning          0
Allergens          0
Prediction         0
dtype: int64
```

```
df['Prediction'].value_counts()
```

```
Contains            198
Does not contain    111
Name: Prediction, dtype: int64
```

```
df.describe()
```

	Food Product	Main Ingredient	Sweetener	Fat/Oil	Seasoning	Allergens	Pre
count	309	309	309	309	309	309	
unique	259	101	10	36	186	40	
top	Ratatouille	Chicken	None	Olive oil	None	None	
freq	4	48	215	69	15	116	

```
lb=LabelEncoder()
lst=['Food Product',    'Main Ingredient' , 'Sweetener' ,   'Fat/Oil' , 'Seasoni

for i in lst:
  df[i]=lb.fit_transform(df[i])
```

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

xtr,xts,ytr,yts=train_test_split(x,y,test_size=0.30,random_state=42)
```

```
std=MinMaxScaler()
std.fit(xtr)
xtr=std.transform(xtr)
xts=std.transform(xts)
```

```
knn=KNeighborsClassifier()
naive=MultinomialNB()
sv=SVC()
```

```
model=[knn,naive,sv]
for i in model:
  print(i)
  i.fit(xtr,ytr)
  ypr=i.predict(xts)
  print(confusion_matrix(yts,ypr))
  print('*'*50)
  print(accuracy_score(yts,ypr))
  print('*'*50)
  print(classification_report(yts,ypr))
```

```
KNeighborsClassifier()
[[48 12]
 [ 0 33]]
**************************************************
0.8709677419354839
**************************************************
                  precision    recall  f1-score   support

        Contains       1.00      0.80      0.89        60
Does not contain       0.73      1.00      0.85        33

        accuracy                           0.87        93
       macro avg       0.87      0.90      0.87        93
    weighted avg       0.91      0.87      0.87        93

MultinomialNB()
[[60  0]
 [33  0]]
**************************************************
0.6451612903225806
**************************************************
                  precision    recall  f1-score   support

        Contains       0.65      1.00      0.78        60
Does not contain       0.00      0.00      0.00        33

        accuracy                           0.65        93
       macro avg       0.32      0.50      0.39        93
    weighted avg       0.42      0.65      0.51        93

SVC()
[[49 11]
 [ 5 28]]
**************************************************
0.8279569892473119
**************************************************
                  precision    recall  f1-score   support

        Contains       0.91      0.82      0.86        60
Does not contain       0.72      0.85      0.78        33

        accuracy                           0.83        93
       macro avg       0.81      0.83      0.82        93
    weighted avg       0.84      0.83      0.83        93

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py
```