

Data Collection and Preprocessing Phase

| | |
|---------------|---|
| Date | 20 June 2024 |
| Team ID | 739954 |
| Project Title | Cereal analysis basede on ratings by using mechine learning techinques |
| Maximum Marks | 6 Marks |

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

| Section | Description |
|---|--|
| Data Overview | Data overview enables a thorough analysis of cereal products based on consumer ratings, helping to identify consumer preferences, market trends, and opportunities for product improvement or market positioning. |
| Data Preparation | Gather cereal attributes and ratings from datasets or sources. Handle missing values, outliers, and inconsistencies. Create new features or modify existing ones to improve model performance. Convert categorical variables to numerical using encoding techniques. Train the model using the training data. Optimize model parameters for better performance. |
| Handling missing values | After loading it is important to check the complete information of data as it can indicate many of the hidden information such as null values in a column or a row. Check whether any null values are there or not. If it is present then following can be done: Imputing data using Imputation method in sklearn. Filling NaN values with mean, median and mode using fillna() method. Heatmap: It is a way of representing the data in 2-D form. It gives a coloured visual summary of the data. |
| Data Visualisation | Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data. Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly. To visualize the dataset we need libraries called Matplotlib and Seaborn. |
| Splitting The Dataset Into Dependent And Independent Variable | In machine learning, the concept of dependent variable (y) and independent variables (x) is important to understand. Dependent variable is nothing but output in the dataset and the independent variable is all inputs in the dataset. With this in mind, we need to split our dataset into the matrix independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column. |
| Data Preprocessing Code Screenshots | |

Import the libraries

"Import Necessary Libraries"

```
] import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
```

Importing The Dataset

```
] df=pd.read_csv("/content/cereal.csv")
```

Analyse The Data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 77 entries, 0 to 76
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  --
 0   name        77 non-null    object  
 1   mfr         77 non-null    object  
 2   type        77 non-null    object  
 3   calories    77 non-null    int64   
 4   protein     77 non-null    int64   
 5   fat         77 non-null    int64   
 6   sodium      77 non-null    int64   
 7   fiber       77 non-null    float64  
 8   carbo       77 non-null    float64  
 9   sugars      77 non-null    int64   
10  potass      77 non-null    int64   
11  vitamins    77 non-null    int64   
12  shelf       77 non-null    int64   
13  weight      77 non-null    float64  
14  cups        77 non-null    float64  
15  rating      77 non-null    float64  
dtypes: float64(5), int64(8), object(3)
memory usage: 9.8+ KB
```

Handling missing values

Handling Missing Values

```
df.isnull().sum()
```

```
name      0
mfr        0
type       0
calories   0
protein    0
fat         0
sodium     0
fiber       0
carbo       0
sugars      0
potass      0
vitamins    0
shelf       0
weight      0
cups        0
rating      0
dtype: int64
```

```
df.isnull().any()
```

```
name      False
mfr        False
type       False
calories   False
protein     False
fat         False
sodium      False
fiber       False
carbo       False
sugars      False
potass      False
vitamins    False
shelf       False
weight      False
cups        False
rating      False
dtype: bool
```

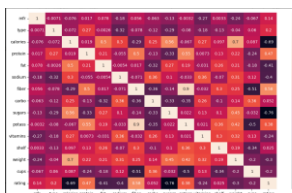
```
df.duplicated().any()
```

```
False
```

Data visualisation

"Heatmap"

```
plt.figure(figsize=(14, 8))
sns.heatmap(df.corr(), annot=True)
```



"BoxPlot"

```
plt.figure(figsize=(10, 1))
sns.boxplot(data=df, y="rating", x="mfr", p="True")
```

```
plt.xticks(rotation=45)
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

```
plt.savefig('boxplot.png')
```

```
plt.close('all')
```

```
plt.show()
```

Splitting The Dataset Into Dependent And Independent Variable

Splitting The Dataset Into Dependent And Independent Variable

```
[ ] x = df.iloc[:,0:14].values
    y = df.iloc[:,14:15].values
```

```
[ ] x
```

```
array([[ 5. ,  0. ,  70. , ...,  5. ,  1. ,  0.33],
       [ 5. ,  0. , 120. , ...,  5. ,  1. ,  1. ],
       [ 2. ,  0. ,  70. , ...,  5. ,  1. ,  0.33],
       ...,
       [ 5. ,  0. , 100. , ...,  5. ,  1. ,  0.47],
       [ 5. ,  0. , 100. , ...,  5. ,  1. ,  1. ],
       [ 5. ,  0. , 120. , ...,  5. ,  1. ,  0.75]])
```

```
[ ] x.shape
```

```
(77, 14)
```

OneHot Encoding

```
[ ] from sklearn.preprocessing import OneHotEncoder
    one = OneHotEncoder()
    x = one.fit_transform(x).toarray()
    x = np.concatenate([x,y],axis=1)
```

```
[ ] x.shape
```

```
(77, 24)
```

Splitting The Data Into Train And Test

```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=)
```