Data Collection and Preprocessing Phase

Date	July 2024
Team ID	739929
Project Title	Cereal analysis basede on ratings by using mechine learning techinques
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

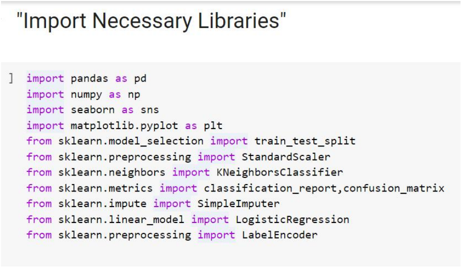
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

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 indication 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 way of representing the data in 2-D form.It gives 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

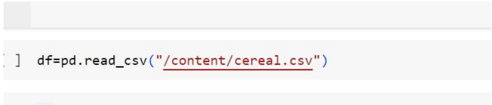
Import the libraries



"Import Necessary Libraries"

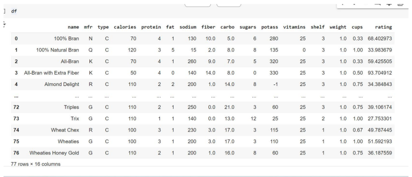
```
] import pandas as pd
  import numpy as np
  import seaborn as sns
  import matplotlib.pyplot as plt
  from sklearn.model_selection import train_test_split
  from sklearn.preprocessing import StandardScaler
  from sklearn.neighbors import KNeighborsClassifier
  from sklearn.metrics import classification_report,confusion_matrix
  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")
```

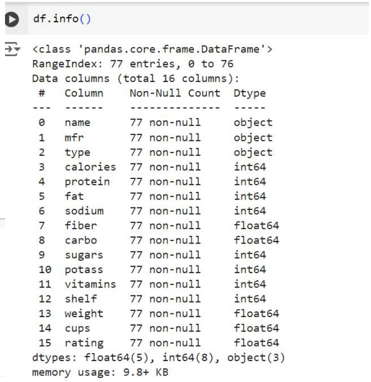
Analyse The Data



```
] df
```

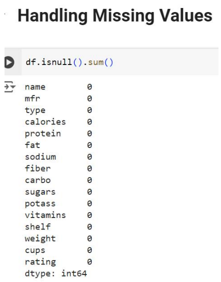
	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
0	100% Bran	N	C	70	4	1	130	10.0	5.0	6	280	25	3	1.0	0.33	68.402973
1	100% Natural Bran	Q	C	120	3	5	15	2.0	8.0	8	135	0	3	1.0	1.00	33.983679
2	All-Bran	K	C	70	4	1	260	9.0	7.0	5	320	25	3	1.0	0.33	59.425505
3	All-Bran with Extra Fiber	K	C	50	4	0	140	14.0	8.0	0	330	25	3	1.0	0.50	93.704912
4	Almond Delight	R	C	110	2	2	200	1.0	14.0	8	-1	25	3	1.0	0.75	34.384843
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
72	Triples	G	C	110	2	1	250	0.0	21.0	3	60	25	3	1.0	0.75	39.106174
73	Trix	G	C	110	1	1	140	0.0	13.0	12	25	25	2	1.0	1.00	27.753301
74	Wheat Chex	R	C	100	3	1	230	3.0	17.0	3	115	25	1	1.0	0.67	49.787445
75	Wheaties	G	C	100	3	1	200	3.0	17.0	3	110	25	1	1.0	1.00	51.592193
76	Wheaties Honey Gold	G	C	110	2	1	200	1.0	16.0	8	60	25	1	1.0	0.75	36.187559

77 rows × 16 columns



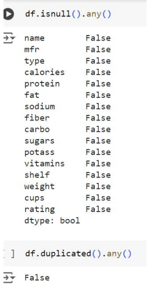
```
df.info()

<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



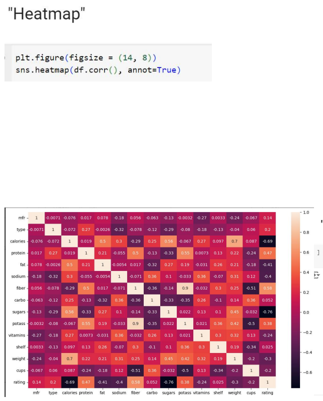
Handling Missing Values

```
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
```

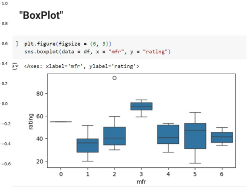
Data visualisation



"Heatmap"

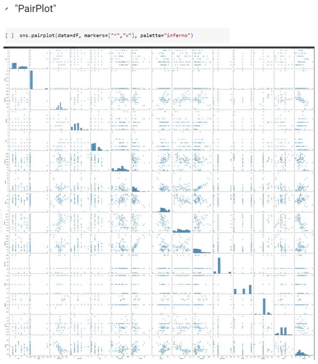
```
plt.figure(figsize = (14, 8))
sns.heatmap(df.corr(), annot=True)
```

	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
mfr	1	-0.0071	-0.076	0.017	0.078	-0.18	0.056	-0.063	-0.13	-0.0032	-0.27	0.0033	-0.24	-0.067	0.14
type	-0.0071	1	-0.072	0.27	-0.0026	-0.32	-0.078	-0.12	-0.29	-0.08	-0.18	-0.13	-0.04	0.06	0.2
calories	-0.076	-0.072	1	0.019	0.5	0.3	-0.29	0.25	0.56	-0.067	0.27	0.097	0.7	0.087	-0.69
protein	0.017	0.27	0.019	1	0.21	-0.055	0.5	-0.13	-0.33	0.55	0.0073	0.13	0.22	-0.24	0.47
fat	0.078	-0.0026	0.5	0.21	1	-0.0054	0.017	-0.32	0.27	0.19	-0.031	0.26	0.21	-0.18	-0.41
sodium	-0.18	-0.32	0.3	-0.055	-0.0054	1	-0.071	0.36	0.1	-0.033	0.36	-0.07	0.31	0.12	-0.4
fiber	0.056	-0.078	-0.29	0.5	0.017	-0.071	1	-0.36	-0.14	0.9	-0.032	0.3	0.25	-0.51	0.58
carbo	-0.063	-0.12	0.25	-0.13	-0.32	0.36	-0.36	1	-0.33	-0.35	0.26	-0.1	0.14	0.36	0.052
sugars	-0.13	-0.29	0.56	-0.33	0.27	0.1	-0.14	-0.33	1	0.022	0.13	0.1	0.45	-0.032	-0.76
potass	-0.0032	-0.08	-0.067	0.55	0.19	-0.033	0.9	-0.35	0.022	1	0.021	0.36	0.42	-0.5	0.38
vitamins	-0.27	-0.18	0.27	0.0073	-0.031	0.36	-0.032	0.26	0.13	0.021	1	0.3	0.32	0.13	-0.24
shelf	0.0033	-0.13	0.097	0.13	0.26	-0.07	0.3	-0.1	0.1	0.36	0.3	1	0.19	-0.34	0.025
weight	-0.24	-0.04	0.7	0.22	0.21	0.31	0.25	0.14	0.45	0.42	0.32	0.19	1	-0.2	-0.3
cups	-0.067	0.06	0.087	-0.24	-0.18	0.12	-0.51	0.36	-0.032	-0.5	0.13	-0.34	-0.2	1	-0.2
rating	0.14	0.2	-0.69	0.47	-0.41	-0.4	0.58	0.052	-0.76	0.38	-0.24	0.025	-0.3	-0.2	1



"BoxPlot"

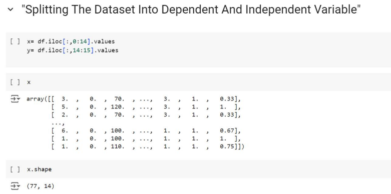
```
] plt.figure(figsize = (6, 3))
  sns.boxplot(data = df, x = "mfr", y = "rating")

<Axes: xlabel='mfr', ylabel='rating'>
```



"PairPlot"

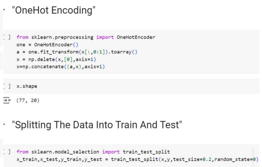
```
[ ] sns.pairplot(data=df, markers=["^","v"], palette="inferno")
```

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([[  3.  ,   0.  ,  70.  , ...,   3.  ,   1.  ,   0.33],
       [  5.  ,   0.  , 120.  , ...,   3.  ,   1.  ,   1.  ],
       [  2.  ,   0.  ,  70.  , ...,   3.  ,   1.  ,   0.33],
       ...,
       [  6.  ,   0.  , 100.  , ...,   1.  ,   1.  ,   0.67],
       [  1.  ,   0.  , 100.  , ...,   1.  ,   1.  ,   1.  ],
       [  1.  ,   0.  , 110.  , ...,   1.  ,   1.  ,   0.75]])

[ ] x.shape

(77, 14)
```



"OneHot Encoding"

```
[ ] from sklearn.preprocessing import OneHotEncoder
    one = OneHotEncoder()
    a = one.fit_transform(x[:,0:1]).toarray()
    x = np.delete(x,[0],axis=1)
    x=np.concatenate((a,x),axis=1)

[ ] x.shape

(77, 20)
```

"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,test_size=0.2,random_state=0)
```