```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tr
```

```python
import pandas as pd
df = pd.read_csv("/content/drive/MyDrive/Capstone/BigBasket Products.csv")
df.head()
```

	index	product	category	sub_category	brand	sale_price	market_pri
0	1	Garlic Oil - Vegetarian Capsule 500 mg	Beauty & Hygiene	Hair Care	Sri Sri Ayurveda	220.0	22
1	2	Water Bottle - Orange	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	180.0	18

Next steps: Generate code with df View recommended plots

```python
# Check data types of each column
df.info()

# Display basic statistics for numeric columns
df.describe()

# Display basic statistics for categorical columns
df.describe(include=['object'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27555 entries, 0 to 27554
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   index         27555 non-null  int64  
 1   product       27554 non-null  object 
 2   category      27555 non-null  object 
 3   sub_category  27555 non-null  object 
 4   brand         27554 non-null  object 
 5   sale_price    27555 non-null  float64
 6   market_price  27555 non-null  float64
 7   type          27555 non-null  object 
 8   rating        18929 non-null  float64
 9   description   27440 non-null  object 
dtypes: float64(3), int64(1), object(6)
memory usage: 2.1+ MB
```

	product	category	sub_category	brand	type	description
count	27554	27555	27555	27554	27555	27440
unique	23540	11	90	2313	426	21944
top	Turmeric Powder/Arisina Pudi	Beauty & Hygiene	Skin Care	Fresho	Face Care	A brand inspired by the Greek goddess of victo...

```python
# Check for missing values
missing_values = df.isnull().sum()

# Display columns with missing values
missing_values[missing_values > 0]
```

```
product           1
brand             1
rating         8626
description     115
dtype: int64
```

```
# Numeric columns statistics
numeric_stats = df.describe()

# Categorical columns statistics
categorical_stats = df.describe(include=['object'])

numeric_stats, categorical_stats
```

```
(             index    sale_price  market_price        rating
 count  27555.00000  27555.000000  27555.000000  18929.000000
 mean   13778.00000    322.514808    382.056664      3.943410
 std     7954.58767    486.263116    581.730717      0.739063
 min        1.00000      2.450000      3.000000      1.000000
 25%     6889.50000     95.000000    100.000000      3.700000
 50%    13778.00000    190.000000    220.000000      4.100000
 75%    20666.50000    359.000000    425.000000      4.300000
 max    27555.00000  12500.000000  12500.000000      5.000000,
                             product          category sub_category   brand  \
 count                         27554             27555        27555   27554
 unique                        23540                11           90    2313
 top     Turmeric Powder/Arisina Pudi  Beauty & Hygiene    Skin Care  Fresho
 freq                             26              7867         2294     638

              type                                        description
 count       27555                                              27440
 unique        426                                              21944
 top     Face Care  A brand inspired by the Greek goddess of victo...
 freq         1508                                                 47  )
```

```
# Checking for missing values again to confirm the columns and counts
missing_values = df.isnull().sum()
missing_values[missing_values > 0]

# Filling missing values for 'product' and 'brand' with mode
df['product'].fillna(df['product'].mode()[0], inplace=True)
df['brand'].fillna(df['brand'].mode()[0], inplace=True)

# Filling missing values for 'rating' with the mean
df['rating'].fillna(df['rating'].mean(), inplace=True)

# Filling missing values for 'description' with an empty string
df['description'].fillna('', inplace=True)

# Verify that there are no more missing values
df.isnull().sum()
```

```
index           0
product         0
category        0
sub_category    0
brand           0
sale_price      0
market_price    0
type            0
rating          0
description     0
dtype: int64
```

```
import pandas as pd
import numpy as np

# Round up the 'rating' column to the nearest integer and convert to int
df['rating'] = df['rating'].apply(np.ceil).astype(int)

# Verify the change
print(df['rating'].head())
```

```
0    5
1    3
2    4
3    4
4    5
Name: rating, dtype: int64
```

```
#Feature Selection
#For identifying key features influencing product ratings, we need to select relevant features from the dataset. Commonly cor

# Select relevant features
selected_features = ['category', 'sub_category', 'brand', 'sale_price', 'market_price', 'type', 'description', 'rating']

# Create a new dataframe with the selected features
df_selected = df[selected_features]

# Display the first few rows of the new dataframe
df_selected.head()

```

	category	sub_category	brand	sale_price	market_price	type	descriptic
0	Beauty & Hygiene	Hair Care	Sri Sri Ayurveda	220.0	220.0	Hair Oil & Serum	This Produ contains Garl Oil that known
1	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	180.0	180.0	Water & Fridge Bottles	Each produ is microwav safe (witho lid)

Next steps: Generate code with df_selected View recommended plots

```
#Data Preprocessing
#We'll now one-hot encode categorical variables (brand, category, sub_category) and prepare the dataset for further analysis.

# Selecting relevant features for analysis
selected_features = ['sale_price', 'brand', 'category', 'sub_category', 'rating']

# Create a new dataframe with selected features
df_selected = df[selected_features]

# One-hot encode categorical variables
df_encoded = pd.get_dummies(df_selected, columns=['brand', 'category', 'sub_category'], drop_first=True, dtype=int)

# Display the first few rows to verify
print(df_encoded.head())


```

```
   sale_price  rating  brand_&Stirred  brand_109°F  brand_137 Degree  \
0       220.0       5               0            0                 0   
1       180.0       3               0            0                 0   
2       119.0       4               0            0                 0   
3       149.0       4               0            0                 0   
4       162.0       5               0            0                 0   

   brand_18 Herbs  brand_1mg  brand_1st Bites  brand_24 Mantra  brand_3 Roses  \
0               0          0                0                0              0   
1               0          0                0                0              0   
2               0          0                0                0              0   
3               0          0                0                0              0   
4               0          0                0                0              0   

   ...  sub_category_Skin Care  sub_category_Snacks & Namkeen  \
0  ...                       0                              0   
1  ...                       0                              0   
2  ...                       0                              0   
3  ...                       0                              0   
4  ...                       0                              0   

   sub_category_Snacks, Dry Fruits, Nuts  \
0                                      0   
1                                      0   
2                                      0   
3                                      0   
4                                      0   

   sub_category_Spreads, Sauces, Ketchup  sub_category_Stationery  \
0                                      0                        0   
1                                      0                        0   
2                                      0                        0   
3                                      0                        0   
4                                      0                        0   

   sub_category_Steel Utensils  sub_category_Storage & Accessories  \
0                            0                                   0
1                            0                                   1
2                            0                                   0
3                            0                                   0
4                            0                                   0

   sub_category_Tea  sub_category_Tinned & Processed Food  sub_category_Water
0                 0                                     0                   0
1                 0                                     0                   0
2                 0                                     0                   0
3                 0                                     0                   0
4                 0                                     0                   0

[5 rows x 2413 columns]
```

```
#Importing Libraries and Splitting Data and then implementing three classification algorithms: Decision Tree Classifier, Ranc
#We'll evaluate each model's performance using cross-validation.

import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Selecting features and target
X = df_encoded.drop(columns=['rating'])
y = df_encoded['rating']

# Splitting the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Display the shapes of the train and test sets to verify
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)

```

```
Training set shape: (22044, 2412) (22044,)
Testing set shape: (5511, 2412) (5511,)
```

```
#Implementing Classification Algorithms

from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Initialize Decision Tree and Random Forest Classifiers
dt_classifier = DecisionTreeClassifier(random_state=42)
rf_classifier = RandomForestClassifier(random_state=42)

# List of classifiers
classifiers = [('Decision Tree', dt_classifier),
               ('Random Forest', rf_classifier)]

# Evaluate each classifier using cross-validation and on the test set
for clf_name, clf in classifiers:
    print(f"Training and evaluating {clf_name}...")

    # Cross-validation
    scores = cross_val_score(clf, X_train, y_train, cv=5, scoring='accuracy', n_jobs=-1)
    print(f"Cross-validation scores: {scores}")
    print(f"Mean accuracy: {scores.mean():.3f}")
    print()

    # Fit the classifier on the training data
    clf.fit(X_train, y_train)

    # Predict on the test data and evaluate
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"{clf_name} Accuracy on test set: {accuracy:.3f}")
    print(classification_report(y_test, y_pred))
    print()
```

```
Training and evaluating Decision Tree...
Cross-validation scores: [0.61193014 0.62463144 0.61510547 0.61555908 0.61297641]
```

```
Mean accuracy: 0.616

Decision Tree Accuracy on test set: 0.625
              precision    recall  f1-score   support

           1       0.15      0.16      0.15        73
           2       0.06      0.06      0.06        72
           3       0.16      0.17      0.16       252
           4       0.69      0.73      0.71      3065
           5       0.62      0.57      0.59      2049

    accuracy                           0.62      5511
   macro avg       0.33      0.34      0.33      5511
weighted avg       0.62      0.62      0.62      5511


Training and evaluating Random Forest...
Cross-validation scores: [0.64118848 0.64254933 0.63461102 0.64073486 0.63112523]
Mean accuracy: 0.638

Random Forest Accuracy on test set: 0.648
              precision    recall  f1-score   support

           1       0.13      0.10      0.11        73
           2       0.08      0.06      0.06        72
           3       0.19      0.16      0.18       252
           4       0.70      0.75      0.72      3065
           5       0.64      0.60      0.62      2049

    accuracy                           0.65      5511
   macro avg       0.35      0.33      0.34      5511
weighted avg       0.64      0.65      0.64      5511
```

```python
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import pandas as pd

# Splitting data into features (X) and target variable (y)
X = df.drop(columns=['rating'])  # Features
y = df['rating']  # Target variable

# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns

# Create a column transformer with OneHotEncoder for categorical columns
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_columns)
    ],
    remainder='passthrough'
)

# Create a pipeline with the preprocessor and Logistic Regression classifier
logreg_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000, random_state=42))
])

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train and evaluate Logistic Regression
print("Training and evaluating Logistic Regression...")
logreg_pipeline.fit(X_train, y_train)

# Cross-validation
print("Cross-validation scores:")
cv_scores = cross_val_score(logreg_pipeline, X_train, y_train, cv=5, scoring='accuracy', n_jobs=-1)
print(cv_scores)
print(f"Mean accuracy: {cv_scores.mean():.3f}")
print()

# Evaluate on the test set
y_pred_logreg = logreg_pipeline.predict(X_test)
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
print(f"Logistic Regression Accuracy on test set: {accuracy_logreg:.3f}")
print(classification_report(y_test, y_pred_logreg))
```

```
Training and evaluating Logistic Regression...
Cross-validation scores:
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[0.59423906 0.55069177 0.56112497 0.55091858 0.59255898]
Mean accuracy: 0.570

Logistic Regression Accuracy on test set: 0.556
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        73
           2       0.20      0.01      0.03        72
           3       1.00      0.00      0.01       252
           4       0.56      1.00      0.71      3065
           5       0.50      0.00      0.00      2049

    accuracy                           0.56      5511
   macro avg       0.45      0.20      0.15      5511
weighted avg       0.54      0.56      0.40      5511

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
```

```
      warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.metrics import confusion_matrix

# Initialize Decision Tree and Random Forest Classifiers
dt_classifier = DecisionTreeClassifier(random_state=42)
rf_classifier = RandomForestClassifier(random_state=42)
logreg_classifier = LogisticRegression(max_iter=1000, random_state=42)

# List of classifiers
classifiers = [('Decision Tree', dt_classifier),
               ('Random Forest', rf_classifier),
               ('Logistic Regression', logreg_classifier)]

# Evaluate each classifier and print confusion matrix
for clf_name, clf in classifiers:
    print(f"Training and evaluating {clf_name}...")

    # Fit the classifier on the training data
    clf.fit(X_train, y_train)

    # Predict on the test data
    y_pred = clf.predict(X_test)

    # Calculate confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Print confusion matrix
    print(f"Confusion Matrix for {clf_name}:")
    print(cm)
    print()
```

```
Training and evaluating Decision Tree...
Confusion Matrix for Decision Tree:
[[  12    5    6   33   17]
 [   1    4    6   43   18]
 [   8    9   42  140   53]
 [  42   35  145 2225  618]
 [  19   13   69  789 1159]]

Training and evaluating Random Forest...
Confusion Matrix for Random Forest:
[[   7    4    6   39   17]
 [   0    4    7   43   18]
 [   4    6   41  147   54]
 [  30   27  106 2290  612]
 [  13   11   53  742 1230]]

Training and evaluating Logistic Regression...
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```