kevin

April 5, 2024

[ ]:
```
import pandas as pd

df = pd.read_csv("apple_quality.csv")
```

[ ]:
```
df
```

[ ]:
```
         A_id      Size    Weight  Sweetness  Crunchiness  Juiciness  Ripeness  \
0         0.0 -3.970049 -2.512336   5.346330    -1.012009   1.844900  0.329840
1         1.0 -1.195217 -2.839257   3.664059     1.588232   0.853286  0.867530
2         2.0 -0.292024 -1.351282  -1.738429    -0.342616   2.838636 -0.038033
3         3.0 -0.657196 -2.271627   1.324874    -0.097875   3.637970 -3.413761
4         4.0  1.364217 -1.296612  -0.384658    -0.553006   3.030874 -1.303849
…       …       …       …         …            …          …         …
3996   3996.0 -0.293118  1.949253  -0.204020    -0.640196   0.024523 -1.087900
3997   3997.0 -2.634515 -2.138247  -2.440461     0.657223   2.199709  4.763859
3998   3998.0 -4.008004 -1.779337   2.366397    -0.200329   2.161435  0.214488
3999   3999.0  0.278540 -1.715505   0.121217    -1.154075   1.266677 -0.776571
4000      NaN       NaN       NaN        NaN          NaN        NaN       NaN

                                 Acidity Quality
0                          -0.491590483    good
1                          -0.722809367    good
2                           2.621636473     bad
3                           0.790723217    good
4                           0.501984036    good
…                                     …     …
3996                        1.854235285    good
3997                       -1.334611391     bad
3998                       -2.229719806    good
3999                        1.599796456    good
4000  Created_by_Nidula_Elgiriyewithana     NaN

[4001 rows x 9 columns]
```

[ ]:
```
missing_percentage = (df.isna().sum() / len(df)) * 100

print(missing_percentage)
```

```
A_id           0.024994
Size           0.024994
Weight         0.024994
Sweetness      0.024994
Crunchiness    0.024994
Juiciness      0.024994
Ripeness       0.024994
Acidity        0.000000
Quality        0.024994
dtype: float64
```

[ ]:
```python
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import pandas as pd

numeric_features = df.select_dtypes(include=['float64']).columns
categorical_features = ['Quality']  # Update with your categorical column(s)

# Define the transformations for numeric and categorical columns
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Combine the transformations using ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Fit and transform the data
df_encoded_scaled = pd.DataFrame(preprocessor.fit_transform(df))

# Print the encoded and scaled dataframe
print("Encoded and Scaled DataFrame:")
print(df_encoded_scaled)
```

```
Encoded and Scaled DataFrame:
             0         1         2         3         4         5         6  \
0    -1.731618 -1.798424 -0.950373  2.993421 -1.424150  0.690545 -0.089872
1    -1.730752 -0.359060 -1.154404  2.127698  0.429746  0.176767  0.197020
2    -1.729886  0.109445 -0.225759 -0.652507 -0.946892  1.205422 -0.286156
3    -1.729020 -0.079977 -0.800146  0.923916 -0.772399  1.619575 -2.087320
```

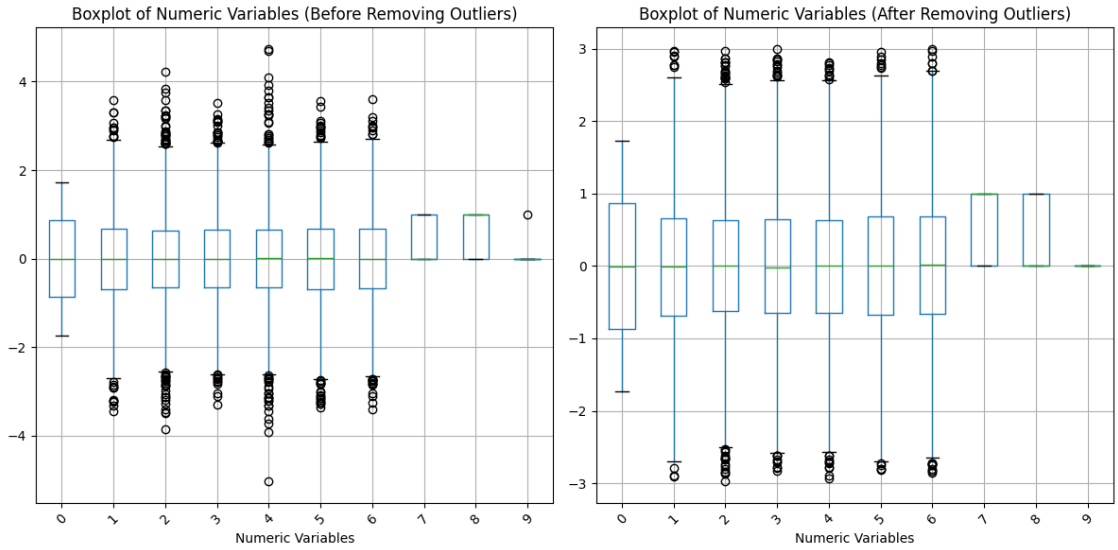
```
4    -1.728154  0.968573 -0.191640  0.044164 -1.096894  1.305025 -0.961548
…          …         …        …         …         …         …         …
3996  1.729020  0.108878  1.834105  0.137124 -1.159058 -0.252634 -0.846326
3997  1.729886 -1.105655 -0.716904 -1.013784 -0.234036  0.874379  2.275957
3998  1.730752 -1.818112 -0.492908  1.459901 -0.845446  0.854549 -0.151419
3999  1.731618  0.405409 -0.453071  0.304496 -1.525439  0.390954 -0.680212
4000       NaN       NaN       NaN       NaN       NaN       NaN       NaN

        7    8    9
0     0.0  1.0  0.0
1     0.0  1.0  0.0
2     1.0  0.0  0.0
3     0.0  1.0  0.0
4     0.0  1.0  0.0
…   …  …  …
3996  0.0  1.0  0.0
3997  1.0  0.0  0.0
3998  0.0  1.0  0.0
3999  0.0  1.0  0.0
4000  0.0  0.0  1.0

[4001 rows x 10 columns]
```

[ ]:
```python
import numpy as np

# Calculate z-scores for each column
z_scores = np.abs((df_encoded_scaled - df_encoded_scaled.mean()) /␣
 ↪df_encoded_scaled.std())

# Define a threshold for considering a value as an outlier
threshold = 3

# Remove rows with any z-score greater than the threshold
df_no_outliers = df_encoded_scaled[(z_scores < threshold).all(axis=1)]

# Print the shape of the dataframe before and after removing outliers
print("Shape of dataframe before removing outliers:", df_encoded_scaled.shape)
print("Shape of dataframe after removing outliers:", df_no_outliers.shape)
```

```
Shape of dataframe before removing outliers: (4001, 10)
Shape of dataframe after removing outliers: (3910, 10)
```

[ ]:
```python
import matplotlib.pyplot as plt

# Create boxplots for each numeric variable before removing outliers
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
df_encoded_scaled.boxplot(rot=45)
```

```
plt.title('Boxplot of Numeric Variables (Before Removing Outliers)')
plt.xlabel('Numeric Variables')

# Create boxplots for each numeric variable after removing outliers
plt.subplot(1, 2, 2)
df_no_outliers.boxplot(rot=45)
plt.title('Boxplot of Numeric Variables (After Removing Outliers)')
plt.xlabel('Numeric Variables')

plt.tight_layout()
plt.show()
```



[ ]:
```
from sklearn.model_selection import train_test_split

# Split the dataset into features (X) and target variable (y)
X = df_encoded_scaled  # Features
y = df['Quality']  # Target variable

# Split the dataset 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)

# Print the shapes of the resulting datasets
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

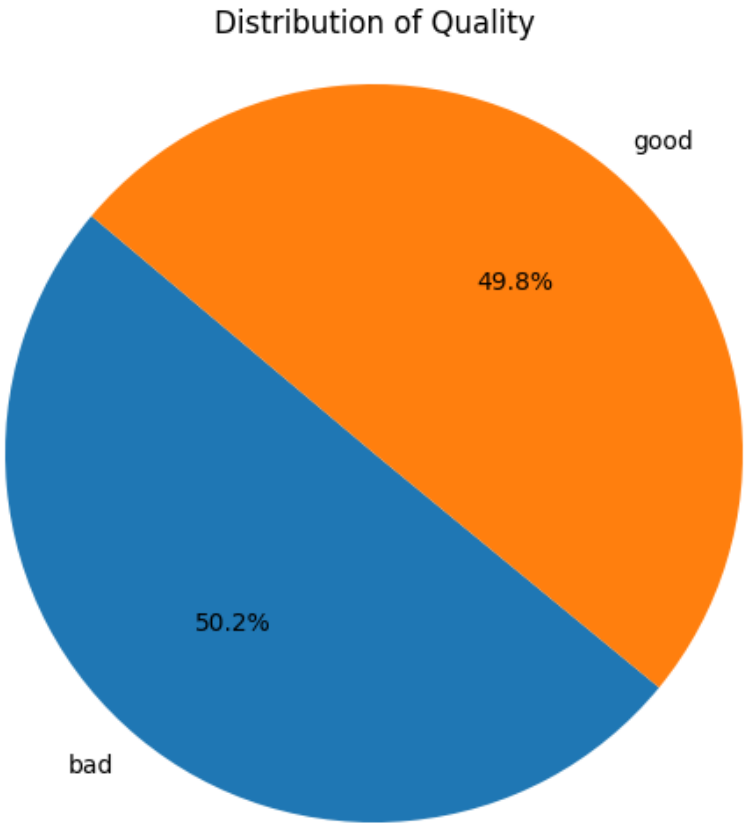
```
Shape of X_train: (3200, 10)
```

```
Shape of X_test: (801, 10)
Shape of y_train: (3200,)
Shape of y_test: (801,)
```

[ ]:
```
from sklearn.model_selection import train_test_split

# Split the dataset into features (X) and target variable (y)
X = df_encoded_scaled  # Features
y = df['Quality']  # Target variable

# Split the dataset 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)

# Print the shapes of the resulting datasets
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (3200, 10)
Shape of X_test: (801, 10)
Shape of y_train: (3200,)
Shape of y_test: (801,)
```

[ ]:
```
import matplotlib.pyplot as plt

# Count the occurrences of each class in the target variable
class_counts = y_train.value_counts()

# Create a pie chart
plt.figure(figsize=(8, 6))
plt.pie(class_counts, labels=class_counts.index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title('Distribution of Quality')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```


Distribution of Quality
good
49.8%
50.2%
bad