Orange Analysis

Michael Brown - Jorge Terrazas

Objectives

- 🍊 Determine the most important attributes in determining an orange’s quality.

- 🍊 Predict the quality of an orange from its measurable features using a regression model

- 🍊 Identify orange quality clusters using a clustering model

Context: Real World Applications

- 🍊 Uphold food quality standards
- 🍊 Reduce unnecessary food waste
- 🍊 Identify trends for pricing

About the Source Data

🍊 Numerical attributes describing the quality of oranges, including their size, weight, sweetness (Brix), acidity (pH), softness, harvest time, and ripeness.

🍊 Categorical attributes such as color, variety, presence of blemishes, and overall quality.

Data Wrangling

Cleanup, preprocessing

dataset:

```
OrangeDF.head(5)
```

✓ 0.0s

	Size (cm)	Weight (g)	Brix (Sweetness)	pH (Acidity)	Softness (1-5)	HarvestTime (days)	Ripeness (1-5)	Color	Variety	Blemishes (Y/N)	Quality (1-5)
0	7.5	180	12.0	3.2	2.0	10	4.0	Orange	Valencia	N	4.0
1	8.2	220	10.5	3.4	3.0	14	4.5	Deep Orange	Navel	N	4.5
2	6.8	150	14.0	3.0	1.0	7	5.0	Light Orange	Cara Cara	N	5.0
3	9.0	250	8.5	3.8	4.0	21	3.5	Orange-Red	Blood Orange	N	3.5
4	8.5	210	11.5	3.3	2.5	12	5.0	Orange	Hamlin	Y (Minor)	4.5

preprocessing & cleaning data:

```
print(OrangeDF["Color"].apply(type).unique())
print(OrangeDF["Variety"].apply(type).unique())
print(OrangeDF["Blemishes (Y/N)"].apply(type).unique())
print(OrangeDF.isna().sum())
```

[5] ✓ 0.0s Python

```
[<class 'str'>]
[<class 'str'>]
[<class 'str'>]
Size (cm)            0
Weight (g)           0
Brix (Sweetness)     0
pH (Acidity)         0
Softness (1-5)       0
HarvestTime (days)   0
Ripeness (1-5)       0
Color                0
Variety              0
Blemishes (Y/N)      0
Quality (1-5)        0
dtype: int64
```

checking for missing or wrong format values:

^No columns with mixed values and no missing values in dataset

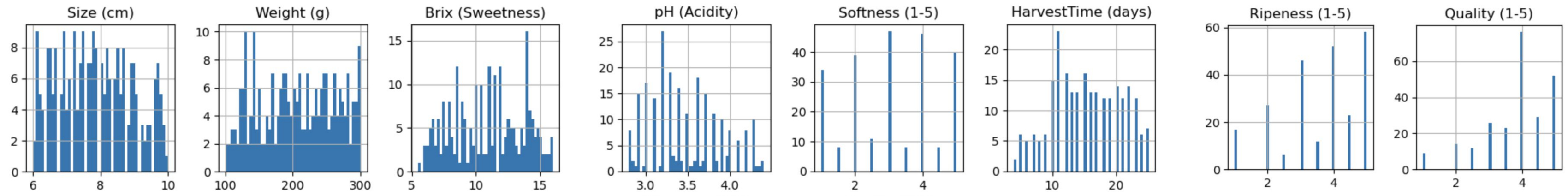
Data Analysis

Distributions, basic relationships

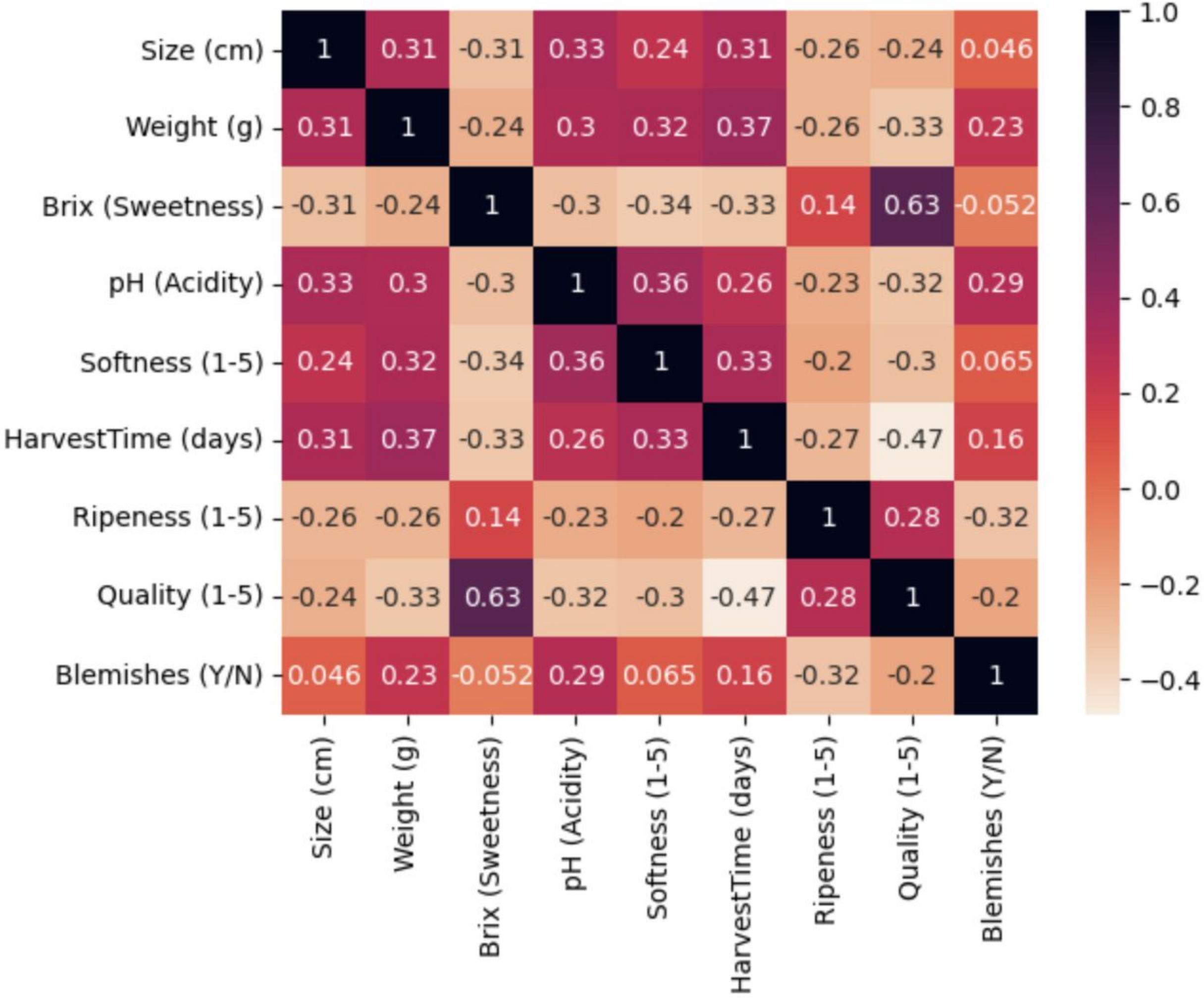
checking for outliers & data distribution of columns:

	Size (cm)	Weight (g)	Brix (Sweetness)	pH (Acidity)	Softness (1-5)	HarvestTime (days)	Ripeness (1-5)	Quality (1-5)
count	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000	241.000000
mean	7.844813	205.128631	10.907884	3.473900	3.072614	15.344398	3.599585	3.817427
std	1.086002	56.461012	2.760446	0.421007	1.323630	5.323852	1.205214	1.014410
min	6.000000	100.000000	5.500000	2.800000	1.000000	4.000000	1.000000	1.000000
25%	6.900000	155.000000	8.500000	3.200000	2.000000	11.000000	3.000000	3.000000
50%	7.800000	205.000000	11.000000	3.400000	3.000000	15.000000	4.000000	4.000000
75%	8.700000	252.000000	13.400000	3.800000	4.000000	20.000000	4.500000	4.500000
max	10.000000	300.000000	16.000000	4.400000	5.000000	25.000000	5.000000	5.000000

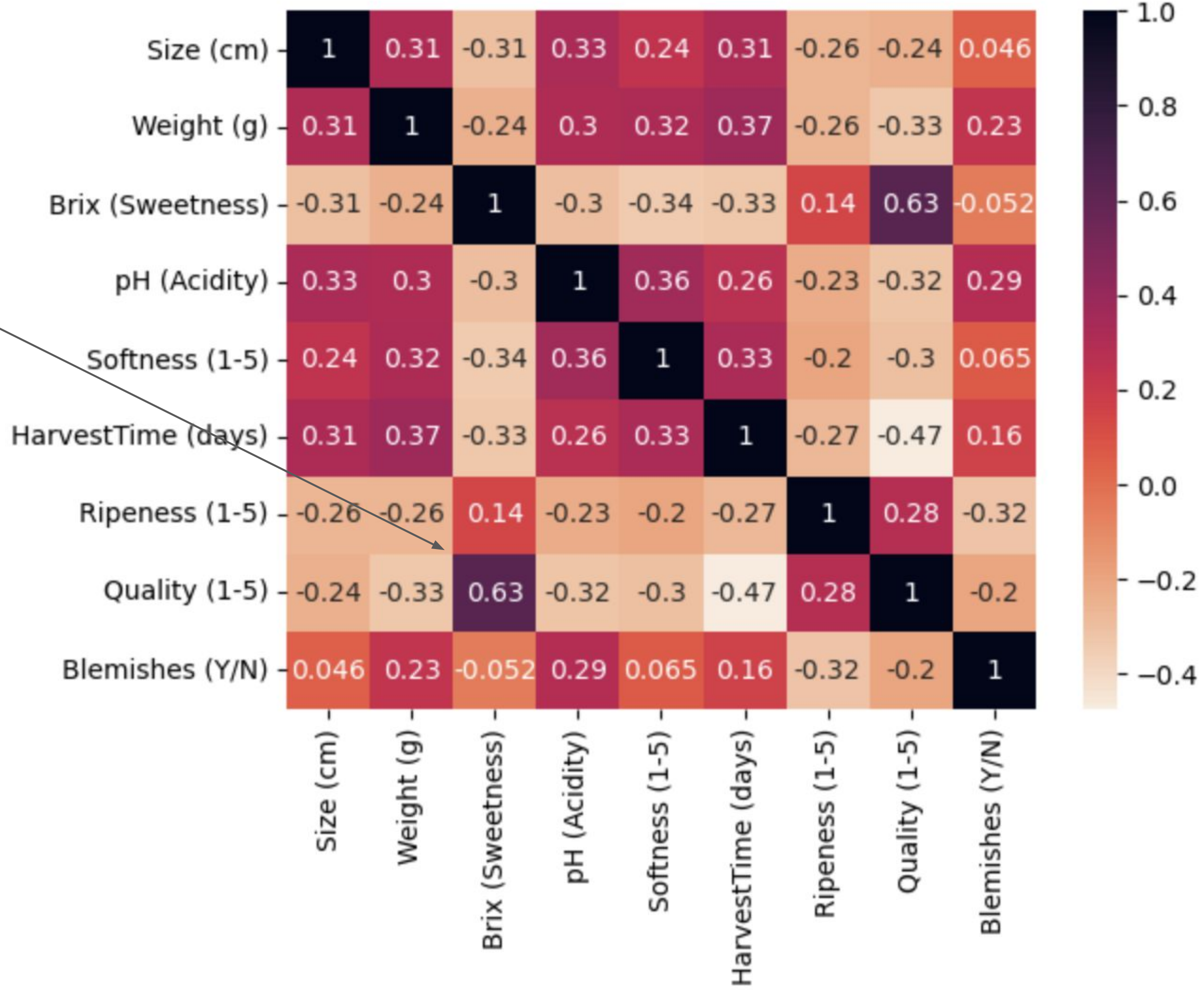
No extreme outliers and fairly normally distributed data for most columns as shown below



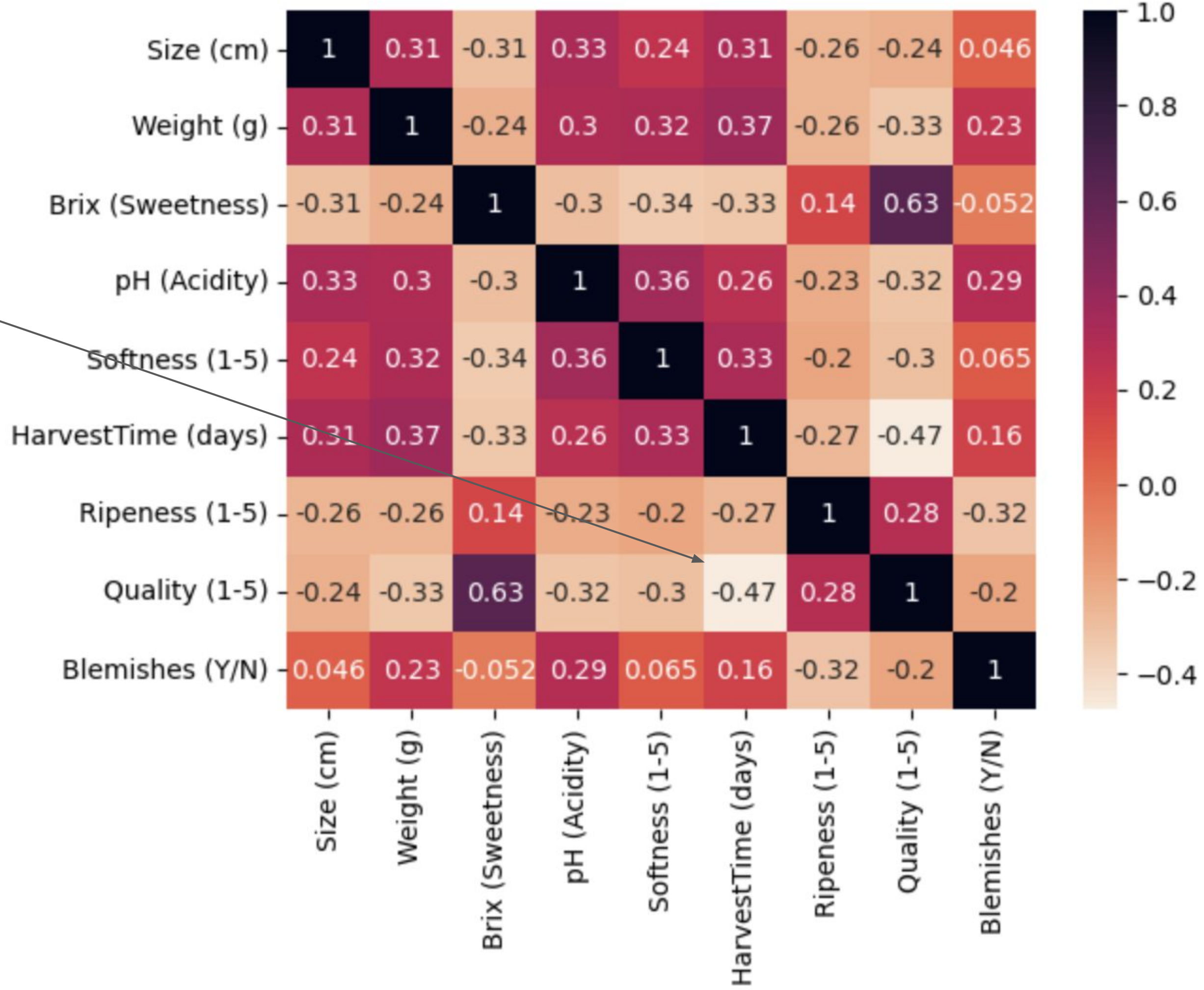
correlation matrix to check for correlations between features



Highest correlation is between Sweetness and Quality



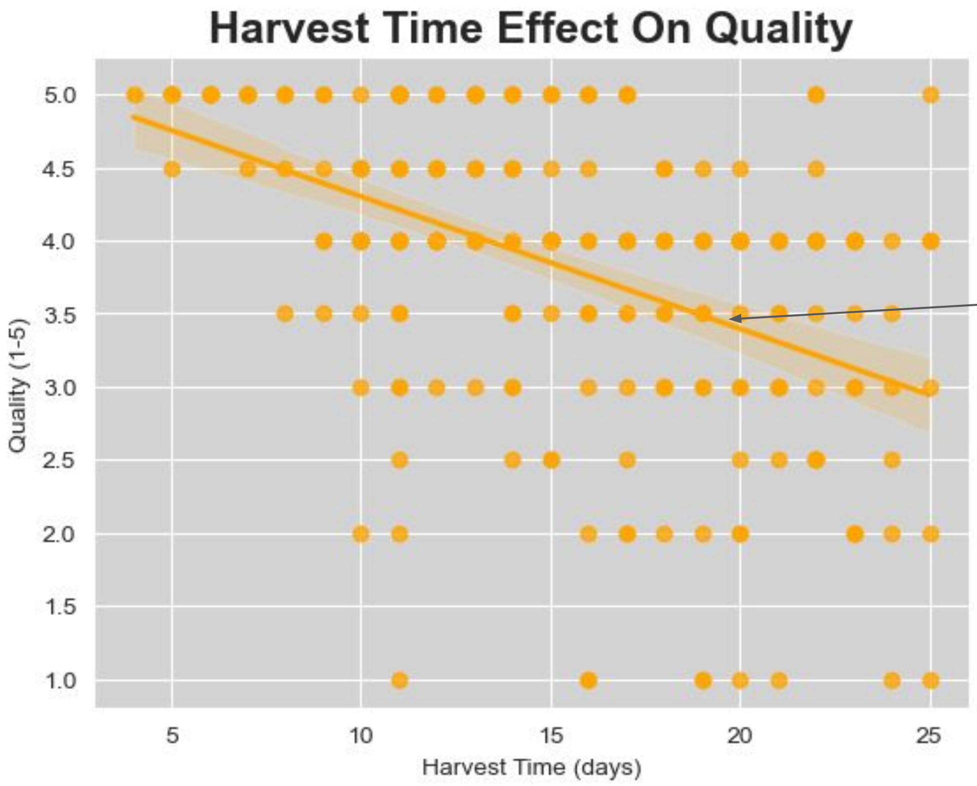
Largest negative correlation is between Harvest Time and Quality



Visual relationship between: Sweetness & Quality

Relatively strong *positive* relationship

What does this tell us?



Visual relationship between:
Harvest Time & Quality

Relatively strong *negative* relationship

Implication?

Models

Baseline, regression, clustering

Baseline Model

```
baseline = OrangeDF["Quality (1-5)"].mean()
mean_guesses = np.full(OrangeDF.shape[0], baseline)
print("mean baseline guess r^2:", r2_score(OrangeDF["Quality (1-5)"], mean_guesses))
print("mean baseline guess MAE:", mean_absolute_error(OrangeDF["Quality (1-5)"], mean_guesses))
print("mean baseline guess MSE:", mean_squared_error(OrangeDF["Quality (1-5)"], mean_guesses))
```

✓ 0.0s

```
mean baseline guess r^2: 0.0
mean baseline guess MAE: 0.7897419121571596
mean baseline guess MSE: 1.0247585268848678
```

Quality Predictions with Regression Models

```python
X = OrangeDF.drop(columns=['Quality (1-5)', 'Variety'])
y = OrangeDF['Quality (1-5)']

lin_reg = LinearRegression()
lin_scores = cross_validate(estimator=lin_reg, X=X, y=y, scoring = ('r2', 'neg_mean_absolute_error', 'neg_mean_squared_error'))
mlp_model = MLPRegressor(hidden_layer_sizes= 1000, activation="relu", early_stopping=True, max_iter=500)
mlp_scores = cross_validate(estimator=mlp_model, X=X, y=y, scoring = ('r2', 'neg_mean_absolute_error', 'neg_mean_squared_error'))
tree_model = DecisionTreeRegressor()
tree_scores = cross_validate(estimator=tree_model, X=X, y=y, scoring = ('r2', 'neg_mean_absolute_error', 'neg_mean_squared_error'))
print("LINEAR REGRESSION:")
print("r^2s:", lin_scores["test_r2"], "\nMAE:", -lin_scores["test_neg_mean_absolute_error"], "\nMSE", -lin_scores['test_neg_mean_squared_error'])
print("\nMULTILAYER PERCEPTRON:")
print("r^2s:", mlp_scores["test_r2"], "\nMAE:", -mlp_scores["test_neg_mean_absolute_error"], "\nMSE", -mlp_scores['test_neg_mean_squared_error'])
print("\nDECISION TREE:")
print("r^2s:", tree_scores["test_r2"], "\nMAE:", -tree_scores["test_neg_mean_absolute_error"], "\nMSE", -tree_scores['test_neg_mean_squared_error'])
```

✓ 3.3s Python

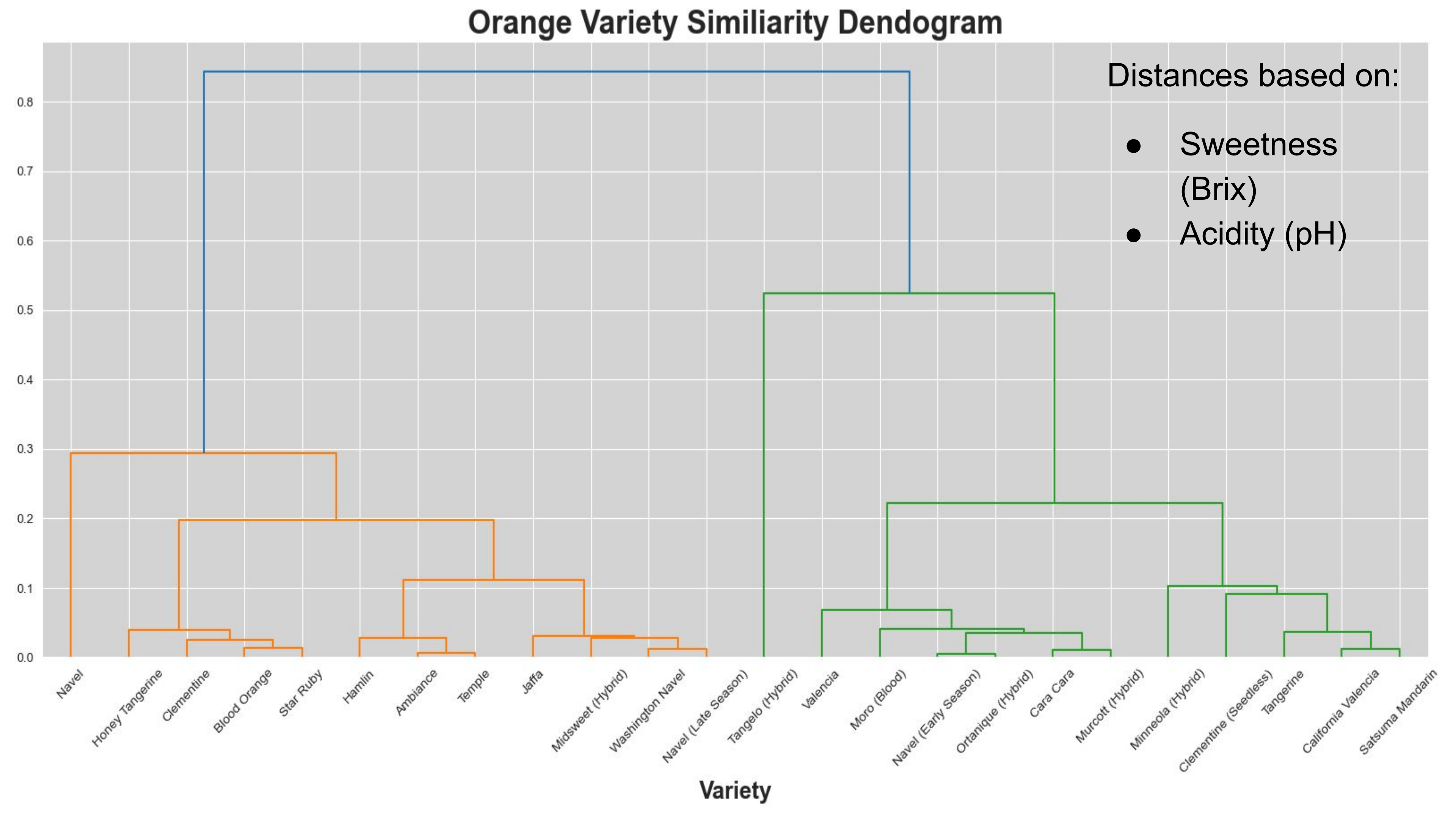
```
LINEAR REGRESSION:
r^2s: [0.39824754 0.42717193 0.23199958 0.23614747 0.22132411]
MAE: [0.63176573 0.53574796 0.72742815 0.54049215 0.69251087]
MSE [0.63496037 0.39375714 0.80391711 0.56286052 0.8922328 ]

MULTILAYER PERCEPTRON:
r^2s: [0.3950977  0.39415192 0.31429317 0.05016218 0.33410057]
MAE: [0.64016155 0.55822475 0.69835993 0.64626905 0.6906529 ]
MSE [0.63828404 0.41645481 0.71777493 0.69990762 0.76300977]

DECISION TREE:
r^2s: [ 0.54549043  0.74238358 -0.09961646 -0.43484023 -0.04545455]
MAE: [0.32653061 0.20833333 0.82291667 0.63541667 0.72916667]
MSE [0.47959184 0.17708333 1.15104167 1.05729167 1.19791667]
```

K-Means Clustering Model: Good or Moderate Quality

```python
n_clusters = 2
KM = KMeans(n_clusters = n_clusters)
to_cluster = X[["Brix (Sweetness)", "HarvestTime (days)"]]
KM.fit(to_cluster)
labels = KM.labels_
cluster_df = pd.DataFrame({"quality score": y.to_list(), "cluster label": labels})
cluster_df.sort_values(by="cluster label")
for i in range(n_clusters):
    print("cluster", i, "median quality score:", cluster_df.loc[cluster_df["cluster label"] == i]["quality score"].median())
```

✓ 0.1s

```
cluster 0 median quality score: 3.5
cluster 1 median quality score: 4.5
```


Orange Variety Similiarity Dendogram
Distances based on:
Sweetness (Brix)
Acidity (pH)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Navel
Honey Tangerine
Clementine
Blood Orange
Star Ruby
Hamlin
Ambiance
Temple
Jaffa
Midsweet (Hybrid)
Washington Navel
Navel (Late Season)
Tangelo (Hybrid)
Valencia
Moro (Blood)
Navel (Early Season)
Ortanique (Hybrid)
Cara Cara
Murcott (Hybrid)
Minneola (Hybrid)
Clementine (Seedless)
Tangerine
California Valencia
Satsuma Mandarin
Variety

Thank You

Citations

Source Data: https://www.kaggle.com/datasets/shruthiiiee/orange-quality/data

https://scikit-learn.org/stable/

https://matplotlib.org/

https://pandas.pydata.org/

https://numpy.org/

https://seaborn.pydata.org/

https://github.com/kiat/Elements-of-Data-Analytics