



APPLE QUALITY ML PREDICTIONS

DATA ANALYTICS BOOTCAMP PROJECT 4
PRESENTED BY KENTO NAKAJIMA
JANUARY 2024



DATA SOURCE TOOLS MODELS



Kaggle.com



Python Pandas, Python
Matplotlib, Python
Seaborn, Sqlite3, Scikit-
Learn



Logistic Regression
Random Forest



ETL

EXTRACT DATA:

- ▶ Apple Quality CSV from Kaggle.com
- ▶ 4000+ rows
- ▶ 9 columns ; A_id, Size, Weight, Sweetness, Crunchiness, Juiciness, Ripeness, Acidity, and Quality

TRANSFORM DATA:

- ▶ Cleaning – Missing Values
- ▶ Conversion – Data Types
- ▶ Selection – Column Removal

LOAD DATA:

- ▶ Load SQLite Database
- ▶ Execute SQL Query
- ▶ Retrieve SQLite Database to DataFrame.
- ▶ Visualization - Seaborn's Pairplot.

TRANSFORM DATA:

```
1 # Find null values
2 print(rawdata.isnull().sum())
```

```
A_id      1
Size      1
Weight    1
Sweetness 1
Crunchiness 1
Juiciness 1
Ripeness  1
Acidity   0
Quality   1
dtype: int64
```

```
1 # Show cleaned results
2 print(rawdata.isnull().sum())
```

```
A_id      0
Size      0
Weight    0
Sweetness 0
Crunchiness 0
Juiciness 0
Ripeness  0
Acidity   0
Quality   0
dtype: int64
```

```
1 # Identify dtype.
2 rawdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4000 entries, 0 to 3999
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  -
0   A_id         4000 non-null   float64
1   Size         4000 non-null   float64
2   Weight       4000 non-null   float64
3   Sweetness    4000 non-null   float64
4   Crunchiness  4000 non-null   float64
5   Juiciness    4000 non-null   float64
6   Ripeness     4000 non-null   float64
7   Acidity      4000 non-null   object
8   Quality      4000 non-null   object
dtypes: float64(7), object(2)
memory usage: 312.5+ KB
```

```
1 # Convert "Acidity" dtype to numerical value
2 rawdata['Acidity'] = rawdata['Acidity'].astype(float)
3 print(rawdata.dtypes)
```

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

```
1 # Remove "A_id" column
2 data = rawdata.drop(columns=['A_id'])
3 data
```

LOAD DATA:

```
1 # Load and insert data into SQL database.
2 conn = sqlite3.connect("appledata.db")
```

```
1 # Write the data to a SQL table named "apple_quality".
2 data.to_sql("apples", conn, if_exists="replace", index=False)
```

]: 4000

```
1 # Close the database connection
2 conn.close()
```

```
1 # Connect to the SQLite database.
2 conn = sqlite3.connect("appledata.db")
```

```
1 # Create a cursor object to execute SQL queries
2 cursor = conn.cursor()
```

```
1 # Execute a SQL query to fetch all data from the "apples" table
2 cursor.execute("SELECT * FROM apples")
```

]: <sqlite3.Cursor at 0x1803bacfc0>

```
1 # Fetch all rows from the result.
2 rows = cursor.fetchall()
```

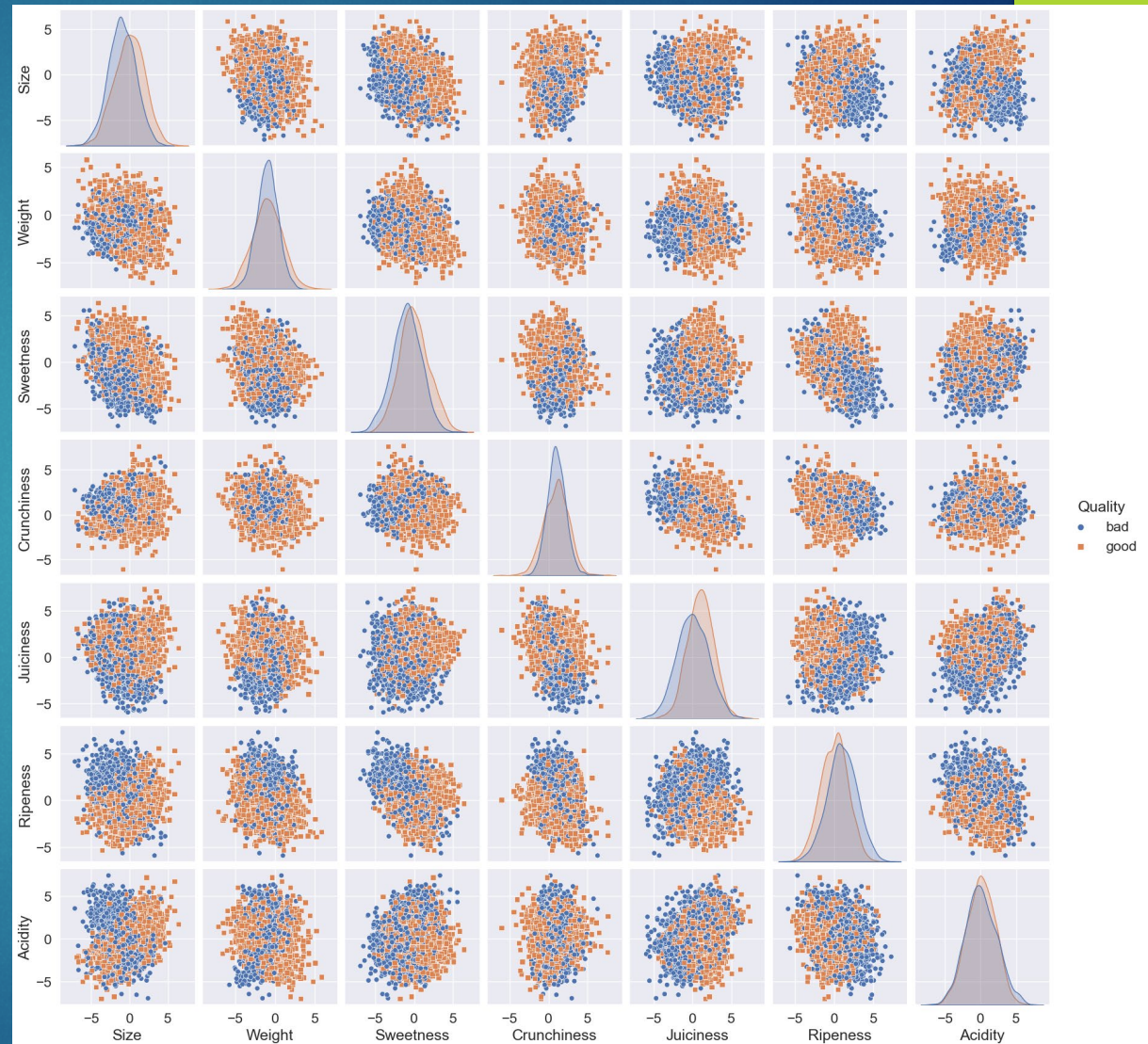
```
1 # Display the fetched rows.
2 for row in rows:
3     print(row)
```

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness	Acidity	Quality
0	-3.970049	-2.512336	5.346330	-1.012009	1.844900	0.329840	-0.491590	good
1	-1.195217	-2.839257	3.664059	1.588232	0.853286	0.867530	-0.722809	good
2	-0.292024	-1.351282	-1.738429	-0.342616	2.838636	-0.038033	2.621636	bad
3	-0.657196	-2.271627	1.324874	-0.097875	3.637970	-3.413761	0.790723	good
4	1.364217	-1.296612	-0.384658	-0.553006	3.030874	-1.303849	0.501984	good
...
3995	0.059386	-1.067408	-3.714549	0.473052	1.697986	2.244055	0.137784	bad
3996	-0.293118	1.949253	-0.204020	-0.640196	0.024523	-1.087900	1.854235	good
3997	-2.634515	-2.138247	-2.440461	0.657223	2.199709	4.763859	-1.334611	bad
3998	-4.008004	-1.779337	2.366397	-0.200329	2.161435	0.214488	-2.229720	good
3999	0.278540	-1.715505	0.121217	-1.154075	1.268677	-0.776571	1.599796	good

4000 rows x 8 columns



DATA VISUALIZATION



LOGISTIC REGRESSION

Relatively balanced in precision, recall, and F1-score – Fair class treatment

Close precision and recall values – No class imbalance issues

Balanced false positives and false negatives – No material bias in types of errors – Room for further investigation

RANDOM FOREST

High accuracy

Few errors

Well balanced precision, recall, and F1-score in both classes

A few false positives and negatives

*** LOGISTIC REGRESSION MODEL EVALUATIONS ***

All Features Included

ACCURACY SCORE: 0.754

CONFUSION MATRIX:

	Predicted 0	Predicted 1
Actual 0	368	131
Actual 1	115	386

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0.0	0.76	0.74	0.75	499
1.0	0.75	0.77	0.76	501
accuracy			0.75	1000
macro avg	0.75	0.75	0.75	1000
weighted avg	0.75	0.75	0.75	1000

*** RANDOM FOREST MODEL EVALUATIONS ***

All Features Included

ACCURACY SCORE: 0.886

CONFUSION MATRIX:

	Predicted 0	Predicted 1
Actual 0	450	62
Actual 1	52	436

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0.0	0.90	0.88	0.89	512
1.0	0.88	0.89	0.88	488
accuracy			0.89	1000
macro avg	0.89	0.89	0.89	1000
weighted avg	0.89	0.89	0.89	1000

Feature Importance Ranking:

	Feature	Importance
4	Juiciness	0.165536
5	Ripeness	0.164077
0	Size	0.160593
2	Sweetness	0.147335
6	Acidity	0.128319
1	Weight	0.121169
3	Crunchiness	0.112971



LOGISTIC REGRESSION MODEL OPTIMIZATION

*** LOGISTIC REGRESSION MODEL EVALUATIONS ***

All Features Included

ACCURACY SCORE: 0.754

CONFUSION MATRIX:

	Predicted 0	Predicted 1
Actual 0	368	131
Actual 1	115	386

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0.0	0.76	0.74	0.75	499
1.0	0.75	0.77	0.76	501
accuracy			0.75	1000
macro avg	0.75	0.75	0.75	1000
weighted avg	0.75	0.75	0.75	1000

*** LOGISTIC REGRESSION MODEL #2 EVALUATIONS ***

Less Features - Size and Weight Removed

ACCURACY SCORE: 0.67

CONFUSION MATRIX:

	Predicted 0	Predicted 1
Actual 0	368	131
Actual 1	115	386

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
bad	0.67	0.68	0.67	499
good	0.67	0.66	0.67	501
accuracy			0.67	1000
macro avg	0.67	0.67	0.67	1000
weighted avg	0.67	0.67	0.67	1000

*** LOGISTIC REGRESSION MODEL #3 EVALUATIONS ***

Less Features - Crunchiness Removed

ACCURACY SCORE: 0.755

CONFUSION MATRIX:

	Predicted 0	Predicted 1
Actual 0	368	131
Actual 1	115	386

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
bad	0.76	0.74	0.75	499
good	0.75	0.77	0.76	501
accuracy			0.76	1000
macro avg	0.76	0.75	0.75	1000
weighted avg	0.76	0.76	0.75	1000



FINAL THOUGHTS

Winner is
Random
Forest!



Thank You, Drew!

