

Wine Reviews & Rating Analysis

By: Ben Lew, Flora Choi, Janine Yoro, Patrick Po, Ahsan Tebha



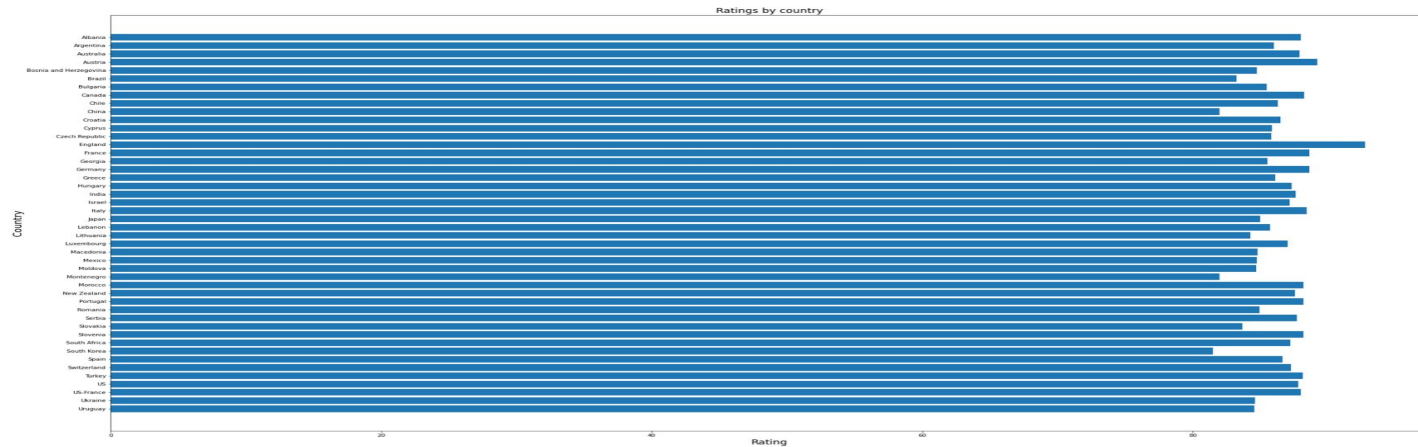
AGENDA

- PURPOSE
- PREPROCESSING
- ANALYSIS/MACHINE LEARNING
- TECHNOLOGY OVERVIEW
- RESULTS/DASHBOARD
- CONCLUSION
- Q&A

After a long day, we all unwind in some way....



- We choose the topic of wine
- Reason being many people like drinking, especially during a global pandemic



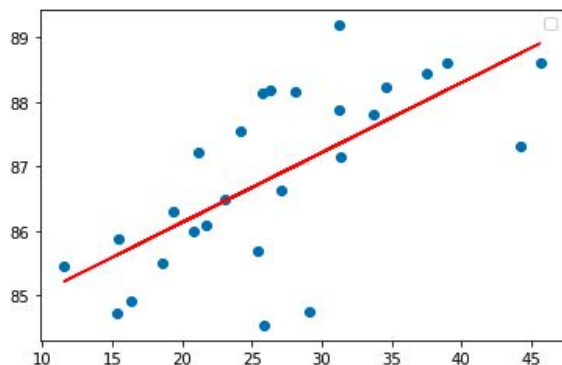
Drunk and Giving Directions

- Our source data originated from winemag.com through a web scrape that an avid wine enthusiast scraped up.
- The question we hoped to answer was *“Is rating predictive of future pricing?”*
- Challenges experienced with this question were the following:
 - We couldn’t make a unique identifier.
 - We couldn’t map the wines year to year.
 - Too much drift in the 2018 data set.
- Ultimately 2017 had enough data points to run through the model.
- This made the ASK evolve to:
 - **“Is rating predictive of price for the year?”**

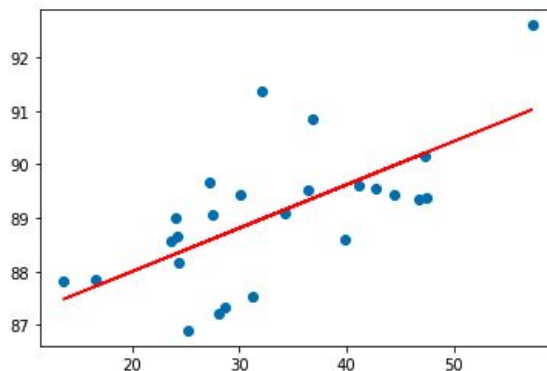
Better Preparation Leads to Better Execution

- What we did:
 - Downloaded 2017 Dataset (150k Rows)
 - Scraped WineMag 2018 Dataset (22k Rows)
 - Dropped columns not relevant to analysis
 - Dropped NaN values
 - Created Target Columns

2017



2018



Data Exploration & Analysis

- Our methodology for feature engineering was to first get a high level view, address the nulls, outliers and then organize/prepare the data.
- Analysis Overview
 - 59,790 rows
 - Few modifications applied to dataframe
 - Change columns names for readability
 - Change numerical data types to integers
 - Removed all the commas in the wine_type column to avoid creating additional columns in the csv



Preprocessing: Dataframes/Tables

```
# This is for 2017 DF
orig_url = "https://drive.google.com/file/d/1zFHNHw6mTh4kyx8pVd2rh2ttV8b7zfU/view?usp=sharing"
file_id = orig_url.split('/')[2]
dwn_url = 'https://drive.google.com/uc?export=download&id=' + file_id
url = requests.get(dwn_url).text
csv_raw = StringIO(url)
csv2017_df = pd.read_csv(csv_raw)
csv2017_df.head(5)
```

| Unnamed: 0 | country | description | designation | points | price | province | region_1 | region_2 | variety | winery |
|------------|---------|-------------|---|---------------------------------------|-------|----------|----------------|-------------------|--------------------|--------------------------|
| 0 | 0 | US | This tremendous 100% varietal wine hails from ... | Martha's Vineyard | 96 | 235.0 | California | Napa Valley | Cabernet Sauvignon | Heitz |
| 1 | 1 | Spain | Ripe aromas of fig, blackberry and cassis are ... | Carodorum SelecciÃ³n Especial Reserva | 96 | 110.0 | Northern Spain | Toro | Tinta de Toro | Bodega Carmen RodrÃ­guez |
| 2 | 2 | US | Mac Watson honors the memory of a wine once ma... | Special Selected Late Harvest | 96 | 90.0 | California | Knights Valley | Sauvignon Blanc | Macauley |
| 3 | 3 | US | This spent 20 months in 30% new French oak, an... | Reserve | 96 | 65.0 | Oregon | Willamette Valley | Pinot Noir | Ponzi |
| 4 | 4 | France | This is the top wine from La BÃ©gude, named af... | La BrÃ©lade | 95 | 66.0 | Provence | Bandol | Provence red blend | Domaine de la BÃ©gude |



| | country | wine_type | price_dollars | ratings_points |
|---|---------|--------------------|---------------|----------------|
| 0 | US | Cabernet Sauvignon | 235.0 | 96 |
| 1 | Spain | Tinta de Toro | 110.0 | 96 |
| 2 | US | Sauvignon Blanc | 90.0 | 96 |
| 3 | US | Pinot Noir | 65.0 | 96 |
| 4 | France | Provence red blend | 66.0 | 95 |

1 DATABASE

CSV FILE

www.quickdatabasediagrams.com

ML_DATA

country varchar(50)
wine_type varchar(100)
price_dollars int
ratings_points int

COUNTRY_REGIONS

country varchar(50)
region varchar(100)

2 CONTINUES FOR PROCESSING FOR MACHINE LEARNING

Database

- Establish a connection between AWS RDS and PostgreSQL
- Establish a connection between our notebook code to both AWS RDS & PostgreSQL
- Bring in the **COUNTRY_REGIONS** dataset for the purpose of joining and future reporting.

1

The screenshot shows the AWS Data Catalog console. On the left, the 'Databases (3)' list includes 'MachineLearningProject' and 'public'. The 'Query Editor' on the right displays a SQL query to create a table named 'COUNTRY_REGIONS' with columns 'Country' (varchar(50)) and 'Region' (varchar(100)). The 'Data Output' tab shows the successful execution of the query, returning 397 rows in 397 milliseconds.

```
1 CREATE TABLE COUNTRY_REGIONS (  
2   Country varchar(50),  
3   Region varchar(100)  
4 );
```

Query returned successfully in 397 msec.

2

The screenshot shows the 'Import - Copying table data' dialog box. It indicates that the table data 'public.country_regions' is being copied from the 'MachineLearningProject' database on the server 'winedata.cn4begnutv3q.us-east-2.rds.amazonaws.com:5432'. The process is scheduled for 'Sun Apr 25 2021 11:44:26 GMT-0700 (Pacific Daylight Time)' and has a duration of 5.01 seconds. A green checkmark and the text 'Successfully completed.' are visible at the bottom.

Import - Copying table data

Copying table data 'public.country_regions' on database 'MachineLearningProject' and server 'winedata.cn4begnutv3q.us-east-2.rds.amazonaws.com:5432'

Sun Apr 25 2021 11:44:26 GMT-0700 (Pacific Daylight Time)

5.01 seconds

More details... Stop Process

Successfully completed.

3

The screenshot shows the 'country_regions' table in the AWS Data Catalog console. The table has two columns: 'country' (character varying (50)) and 'region' (character varying (100)). The 'Data Output' tab displays a list of 10 rows, including Argentina, Brazil, Chile, Uruguay, Australia, China, India, Japan, New Zealand, and South Korea, each with its corresponding region.

country_regions

country character varying (50) region character varying (100)

| country | region |
|-------------|-----------------|
| Argentina | SAMER Countries |
| Brazil | SAMER Countries |
| Chile | SAMER Countries |
| Uruguay | SAMER Countries |
| Australia | APAC Countries |
| China | APAC Countries |
| India | APAC Countries |
| Japan | APAC Countries |
| New Zealand | APAC Countries |
| South Korea | APAC Countries |

4

The screenshot shows the 'SELECT' query in the AWS Data Catalog console. The query is a join between 'country_regions' and 'data_combined' on the 'Country' column. The 'Data Output' tab displays a table with 10 rows, including Namer Countries, EMEA Countries, and Middle Eastern Countries, each with its corresponding region, country, wine_type, price_dollars, and ratings_points.

```
12 SELECT B.Region, A.*  
13 INTO DATA_COMBINED  
14 FROM ML_DATA A  
15 JOIN COUNTRY_REGIONS B ON A.Country = B.Country  
16  
17 SELECT * FROM DATA_COMBINED LIMIT 10;
```

| region | country | wine_type | price_dollars | ratings_points |
|-----------------|---------|--------------------|---------------|----------------|
| NAMER Countries | US | Cabernet Sauvignon | 235 | 96 |
| EMEA Countries | Spain | Tinta de Toro | 110 | 96 |
| NAMER Countries | US | Sauvignon Blanc | 90 | 96 |
| NAMER Countries | US | Pinot Noir | 65 | 96 |
| EMEA Countries | France | Provence red blend | 66 | 95 |
| EMEA Countries | Spain | Tinta de Toro | 73 | 95 |
| EMEA Countries | Spain | Tinta de Toro | 65 | 95 |
| EMEA Countries | Spain | Tinta de Toro | 110 | 95 |
| NAMER Countries | US | Pinot Noir | 65 | 95 |
| NAMER Countries | US | Pinot Noir | 60 | 95 |

Ex

The screenshot shows the 'SELECT' query in the AWS Data Catalog console. The query is a join between 'country_regions' and 'data_combined' on the 'Country' column. The 'Data Output' tab displays a table with 5 rows, including EMEA Countries, NAMER Countries, Middle Eastern Countries, SAMER Countries, and APAC Countries, each with its corresponding region, country, wine_type, price_dollars, and ratings_points.

```
23 SELECT REGION,  
24 COUNT(COUNTRY)  
25 FROM DATA_COMBINED  
26 GROUP BY REGION;
```

| region | count |
|--------------------------|-------|
| EMEA Countries | 25725 |
| NAMER Countries | 25403 |
| Middle Eastern Countries | 384 |
| SAMER Countries | 5403 |
| APAC Countries | 2835 |

Machine Learning - Initial Model

MAIN OBJECTIVE: *GET A MODEL TO WORK*

Begin by building a working model at its simplest form.

Eliminate any noise related to descriptive columns & keep only the numerical values.

Create a new categorical value

GOOD WINE: Ratings 89 or Below
GREAT WINE: Ratings 90 or Above

| | price_dollars | ratings_desc |
|---|---------------|--------------|
| 1 | 110 | 1.0 |
| 2 | 90 | 1.0 |
| 3 | 65 | 1.0 |
| 4 | 66 | 1.0 |
| 5 | 73 | 1.0 |

DISTRIBUTION

- ★ Check the spread of price
- ★ Normal distribution?
- ★ Address Outliers (Remove top 1%)



LOGISTIC REGRESSION MODEL

Wine Spectator's Wine Ratings 100-Point Scale:

- **95-100** — Classic; a great wine
- **90-94** — Outstanding; superior character and style
- **80-89** — Good to very good; wine with special qualities
- **70-79** — Average; drinkable wine that may have minor flaws
- **60-69** — Below average; drinkable but not recommended
- **50-59** — Poor; undrinkable, not recommended

Source: <http://www.winewins.com/wine-ratings/>

FEATURES VS TARGET

Feature: PRICE_DOLLARS
Target: RATINGS_DESC
("0" For Good Wine, "1" For Great Wine")

```
#get stats of data -- notice the max  
new2017_df.describe()
```

| | price_dollars | ratings_desc |
|-------|---------------|--------------|
| count | 59750.000000 | 59750.000000 |
| mean | 36.216268 | 0.360854 |
| std | 36.442206 | 0.480252 |
| min | 4.000000 | 0.000000 |
| 25% | 17.000000 | 0.000000 |
| 50% | 27.000000 | 0.000000 |
| 75% | 45.000000 | 1.000000 |
| max | 2013.000000 | 1.000000 |

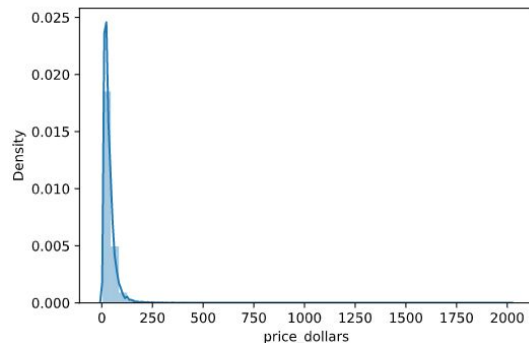
FINAL DATAFRAME

1. Split the data into Training and Testing
2. Created the Logistic Regression Model
3. Fit / Trained the model
4. Made Predictions
5. Validated model using confusion matrix
6. Generated an accuracy score

Machine Learning - Model Optimization

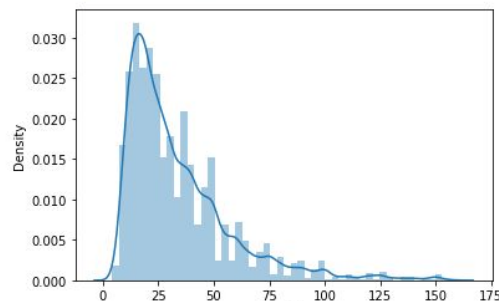


<AxesSubplot:xlabel='price_dollars', ylabel='Density'>



Full Dataset Distribution

<AxesSubplot:xlabel='price_dollars', ylabel='Density'>



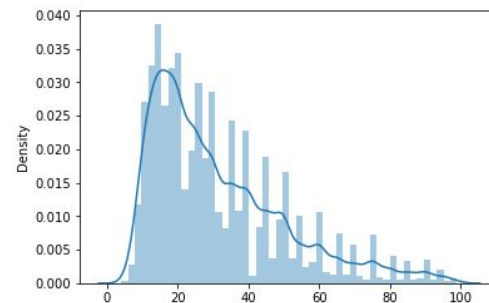
Outliers
re-defined
to >\$100
price

Standard
Scalar
applied

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

0.7366107654855288

<AxesSubplot:xlabel='price_dollars', ylabel='Density'>



```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

0.7368128431440684

| | Predicted good_wine | Predicted great_wine |
|-------------------|---------------------|----------------------|
| Actual good_wine | 8585 | 2943 |
| Actual great_wine | 952 | 2308 |

```
from imblearn.metrics import classification_report_imbalanced
print(classification_report_imbalanced(y_test,predictions))
```

| | pre | rec | spe | f1 | geo | iba | sup |
|-------------|------|------|------|------|------|------|-------|
| 0.0 | 0.74 | 0.90 | 0.44 | 0.82 | 0.63 | 0.41 | 9537 |
| 1.0 | 0.71 | 0.44 | 0.90 | 0.54 | 0.63 | 0.38 | 5251 |
| avg / total | 0.73 | 0.74 | 0.60 | 0.72 | 0.63 | 0.40 | 14788 |

Predicted good_wine Predicted great_wine

| Actual good_wine | 8360 | 2674 |
|-------------------|------|------|
| Actual great_wine | 1113 | 2242 |

```
from imblearn.metrics import classification_report_imbalanced
print(classification_report_imbalanced(y_test,predictions))
```

| | pre | rec | spe | f1 | geo | iba | sup |
|-------------|------|------|------|------|------|------|-------|
| 0.0 | 0.76 | 0.88 | 0.46 | 0.82 | 0.63 | 0.42 | 9473 |
| 1.0 | 0.67 | 0.46 | 0.88 | 0.54 | 0.63 | 0.39 | 4916 |
| avg / total | 0.73 | 0.74 | 0.60 | 0.72 | 0.63 | 0.41 | 14389 |

Machine Learning - Validation Models



| | US | | | Other Countries | | |
|------------------|-------------------|---------------------|----------------------|-------------------|---------------------|----------------------|
| Accuracy Score | 73% | | | 80% | | |
| Confusion Matrix | | | | | | |
| | | Predicted Good Wine | Predicted Great Wine | | Predicted Good Wine | Predicted Great Wine |
| | Actual Good Wine | 9,433 | 3,193 | Actual Good Wine | 12,558 | 2,907 |
| | Actual Great Wine | 921 | 1,907 | Actual Great Wine | 751 | 2,240 |

Classification Reports

| | pre | rec | spe | f1 | geo | iba | sup |
|-------------|------|------|------|------|------|------|-------|
| 0.0 | 0.81 | 0.94 | 0.44 | 0.87 | 0.64 | 0.43 | 13309 |
| 1.0 | 0.75 | 0.44 | 0.94 | 0.55 | 0.64 | 0.39 | 5147 |
| avg / total | 0.79 | 0.80 | 0.58 | 0.78 | 0.64 | 0.42 | 18456 |

| | pre | rec | spe | f1 | geo | iba | sup |
|-------------|------|------|------|------|------|------|-------|
| 0.0 | 0.81 | 0.94 | 0.44 | 0.87 | 0.64 | 0.43 | 13309 |
| 1.0 | 0.75 | 0.44 | 0.94 | 0.55 | 0.64 | 0.39 | 5147 |
| avg / total | 0.79 | 0.80 | 0.58 | 0.78 | 0.64 | 0.42 | 18456 |

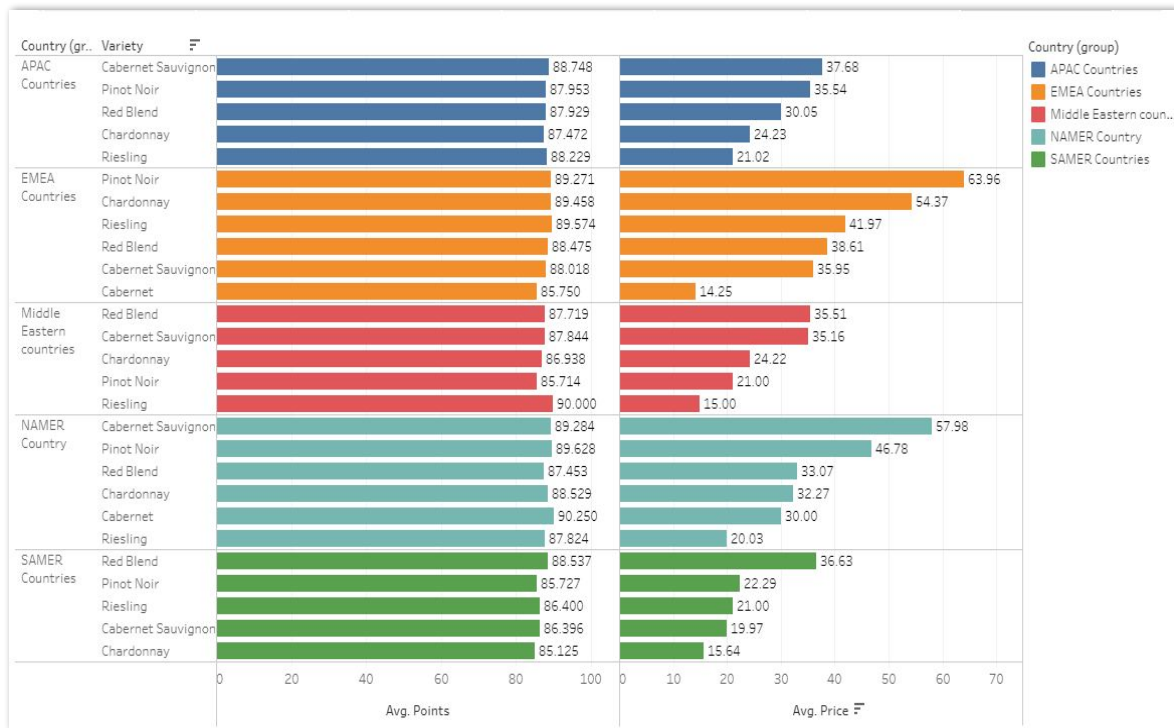
Tech Used & Results

Technology Overview:

- Python, Tableau, Postgres
- Google Colab, Github

Dashboard:

- Link to our Tableau project:
[Link to Tableau](#)



Further Analysis & Alternative Actions

- The data set could be further drilled down...
- Run multiple regression models both linear and logistic to find best correlating features



Q&As

Thank You

Powered By:



B

A



F





