

Wine Quality Machine Learning

...

UT Data Analytics Bootcamp 2020 Group 5

Why did we choose wine ???



Questions ???

1. How does weather impact wine quality?
2. How does score vary by region in USA , by type (red, white) ?
3. How does soil quality affect wine quality ?



#YOUNGERTV



I'LL BRING THE WINE

Data Exploration

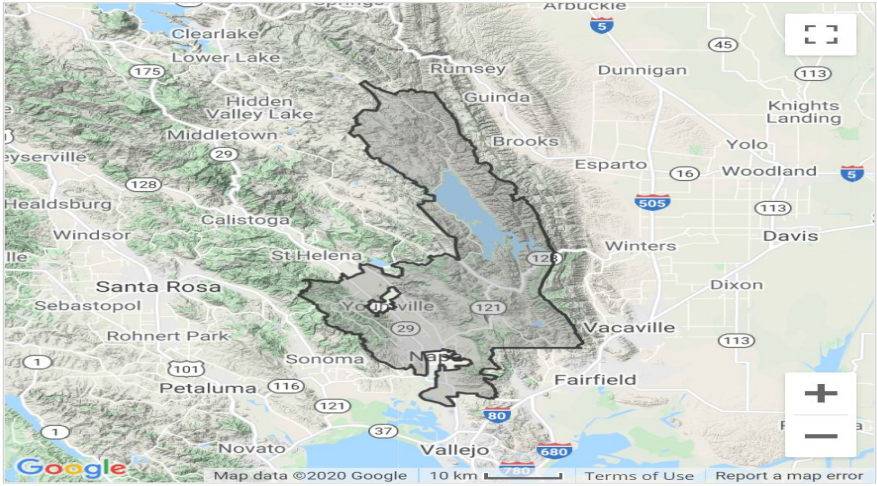
LOCATION DETAILS

Name	Napa, CA 94558
ID	ZIP:94558
Type	Zip Code
Included Stations	10 (See station list below)

PERIOD OF RECORD

Start ¹	1906-04-01
End ¹	2020-11-05
Coverage ²	100%

ADD TO CART



Location Station List & Summarized Data Inventory

Available Data Types

Air Temperature

Evaporation

Precipitation

Sunshine

Weather Type

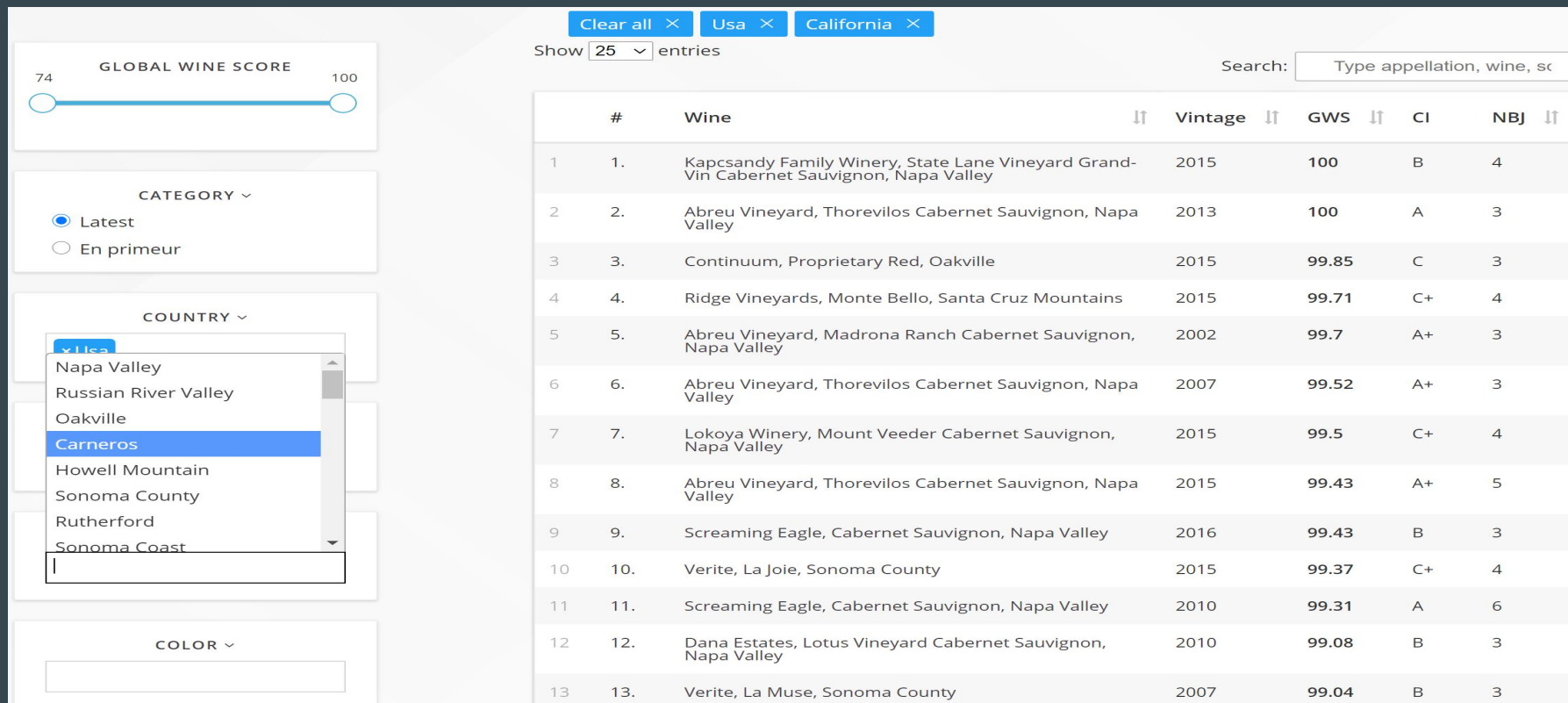
Wind

DATA TYPE	DESCRIPTION	START	END	COVERAGE ²
TAVG	Average Temperature,	1998-05-22	2005-07-31	<div></div> 100%
TMAX	Maximum temperature	1906-04-01	2020-11-05	<div></div> 100%
TMIN	Minimum temperature	1906-04-01	2020-11-05	<div></div> 100%
TOBS	Temperature at the time of observation	1958-05-01	2020-11-04	<div></div> 98%

Data Exploration - Continued

```
In [ ]: willamettdef = pd.DataFrame()
states = {
#     'Napa': {
#         'zip': '94559',
#         'years': ['1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007'],
#     },
#     'Walla': {
#         'zip': '99362',
#         'years': ['1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007'],
#     },
#     'Columbia': {
#         'zip': '98813',
#         'years': ['1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007'],
#     },
#     'Sonoma': {
#         'zip': '95476',
#         'years': ['1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007'],
#     },
#     'Santa': {
#         'zip': '95062',
#         'years': ['1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007'],
#     },
#     'Yakima': {
#         'zip': '98903',
#         'years': ['1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007'],
#     },
#     'Dundee': {
#         'zip': '97148',
#         'years': ['1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007'],
#     },
#     'Willamette': {
#         'zip': '97302',
#         'years': ['1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007'],
#     }
}
```

Data Exploration - Continued



Data Exploration - Continued

Loop through API call for appellations and add to DataFrame

```
In [5]: # Copy appellation_geo_df to store new coordinates if original coordinates do not return API results
appellations_geo_df_REVISED = appellations_geo_df.copy()

# Loop through coordinates of appellations to create API call and add to soils_df
for appellation in range(len(appellations_geo_df)):
    appLon = appellations_geo_df.iloc[appellation, 3]
    appLat = appellations_geo_df.iloc[appellation, 2]
    url = f'https://rest.soilgrids.org/soilgrids/v2.0/properties/query?lon={appLon}&lat={appLat}&property=bdod&property=
    appellation_soil_api_response = requests.get(url).json()

    # If API return is null, change coordinates until a nearby point returns non-null results
    while not appellation_soil_api_response['properties']['layers'][i]['depths'][j]['values']['Q0.5']:
        random_multiplier = 1
        appLon = appLon + random.uniform(-.005, 0.005)*random_multiplier
        appLat = appLat + random.uniform(-.005, 0.005)*random_multiplier
        # Update coordinates in revised dataframe
        appellations_geo_df_REVISED.iloc[appellation, 3] = appLon
        appellations_geo_df_REVISED.iloc[appellation, 2] = appLat
        url = f'https://rest.soilgrids.org/soilgrids/v2.0/properties/query?lon={appLon}&lat={appLat}&property=bdod&prop
        # Increment random_multiplier to help assure that random changes move iterations farther from original point
        random_multiplier += 0.1
        appellation_soil_api_response = requests.get(url).json()

    # Initiate list to hold data and append appellation name based on index number
    soils_row = []
    soils_row.append(appellations_geo_df.iloc[appellation, 0])

    # Loop through API return and add selected data to soils_row
    for i in range(len(appellation_soil_api_response['properties']['layers'])):
        for j in range(len(appellation_soil_api_response['properties']['layers'][i]['depths'])):
            soils_row.append(appellation_soil_api_response['properties']['layers'][i]['depths'][j]['values']['Q0.5'])

    # Add soils_row as row to end of soils_df
    soils_df.loc[len(soils_df)] = soils_row
#soils_df
```


Data Exploration - Continued

```
In [8]: # Create list of soil variables for use in selected calculations
```

```
soils_variables = []  
for i in range(len(appellation_sample['properties']['layers'])):  
    soils_variables.append(appellation_sample['properties'][i]['name'])  
soils_variables
```

```
Out[8]: ['bdod',  
         'cec',  
         'cfvo',  
         'clay',  
         'nitrogen',  
         'ocd',  
         'ocs',  
         'phh2o',  
         'sand',  
         'silt',  
         'soc']
```

```
In [9]: # Edit list of soil variables  
# Remove 'ocs' column as there is only one range for ocs  
soils_variables.remove('ocs')  
# Remove 'phh2o' column as average of pH is calculated  
soils_variables.remove('phh2o')  
soils_variables
```

```
Out[9]: ['bdod', 'cec', 'cfvo', 'clay', 'nitrogen', 'ocd', 'sand', 'silt', 'soc']
```

```
In [10]: # Create list of depths for measurements up to 100cm for use with calculations  
soils_range_0_100cm = []  
for j in range(len(appellation_sample['properties']['layers'][i]['depths'])-1):  
    soils_range_0_100cm.append(appellation_sample['properties'][i]['depths'][j]['label'])  
soils_range_0_100cm
```

```
Out[10]: ['0-5cm', '5-15cm', '15-30cm', '30-60cm', '60-100cm']
```

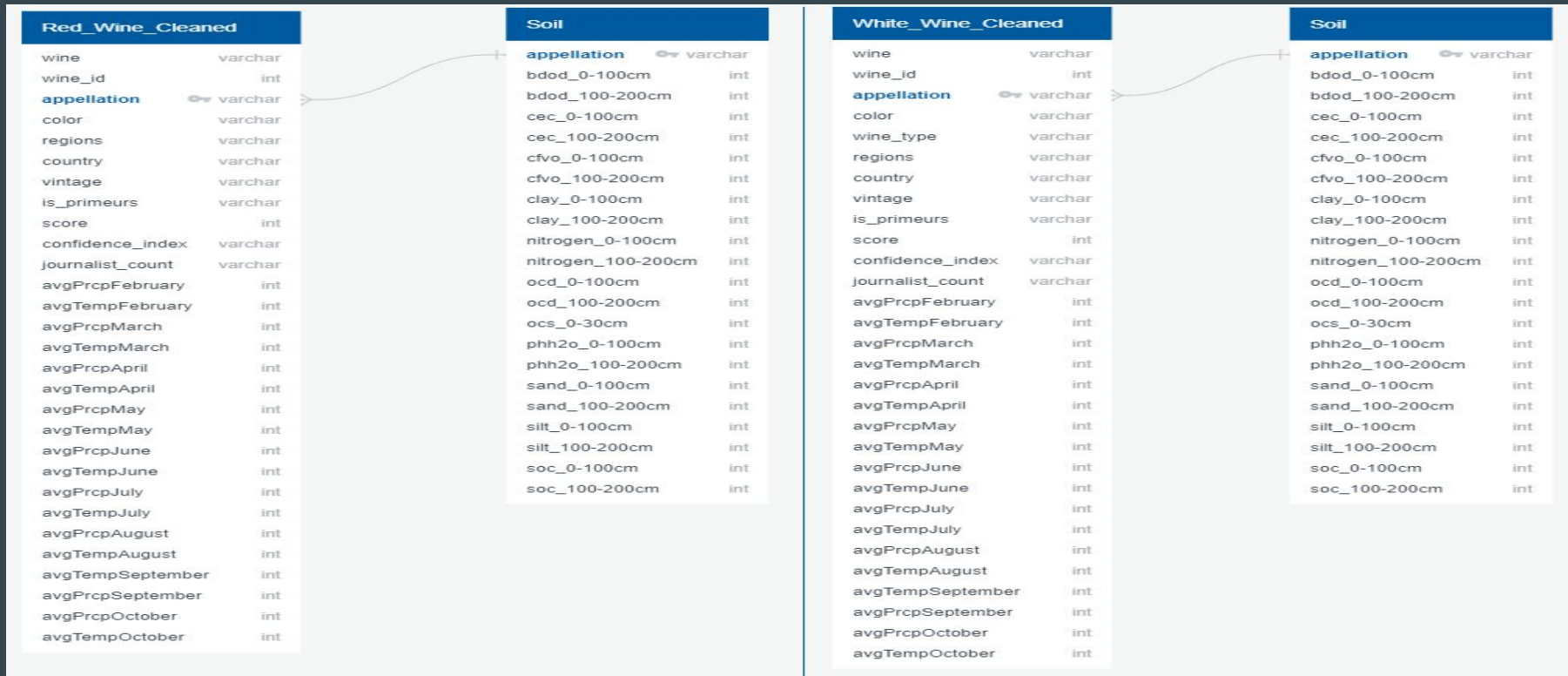
```
In [11]: # Create list of weights for measurements for use with calculations  
soils_range_weight = [0.05, 0.1, 0.15, 0.3, 0.4]
```

```
In [12]: # Calculate average values for depths down to 100cm and populate new column  
for i in range(len(soils_variables)):  
    variable = soils_variables[i]  
    variable_summator = 0.0  
    for j in range(len(soils_range_0_100cm)):  
        depth = soils_range_0_100cm[j]  
        variable_summator = variable_summator + soils_df[f'{variable}_{depth}']*soils_range_weight[j]  
    soils_df[f'{variable}_0-100cm'] = variable_summator  
#soils_df
```

Database

- We utilized Amazon AWS and PgAdmin for SQL database
- Created 4 tables in the database
- Joined red wine and white wine dataframes with the soil dataframe

Database - continued



Database - continued

The screenshot shows the pgAdmin 4 web interface. The left sidebar displays the database structure for 'postgres/postgres@wine'. The main pane shows the 'Query Editor' with a SQL query that joins 'red_table' and 'soil_table' on 'appellation'. The 'Data Output' tab shows the results of the query, displaying columns like 'appellation', 'wine', 'wine_id', 'color', 'regions', 'country', 'vintage', 'is_primeurs', 'score', 'confidence_index', and 'journalist_count'.

	appellation text	wine text	wine_id integer	color text	regions text	country text	vintage integer	is_primeurs boolean	score double precision	confidence_index text	journalist_count integer
1	Santa Cruz Mou...	Ridge V...	120786	Red	California	Usa	2013	false	95.83	C+	8
2	Rutherford	Quintes...	118360	Red	California	Usa	2015	false	95.77	C	3
3	Napa Valley	Harlan ...	81980	Red	California	Usa	2001	false	98.89	A+	6
4	Napa Valley	Dalla V...	48290	Red	California	Usa	2001	false	96.49	B+	5
5	Napa Valley	Philip T...	115154	Red	California	Usa	2001	false	96.2	B	3
6	Napa Valley	Colgin ...	44618	Red	California	Usa	2001	false	95.6	B	5
7	Rutherford	Sloan, ...	128313	Red	California	Usa	2001	false	95.19	C+	3
8	Stags Leap Dist...	Shafer ...	126746	Red	California	Usa	2001	false	93.69	C	6
9	Napa Valley	Dalla V...	48287	Red	California	Usa	2001	false	93.66	B+	5
10	Napa Valley	Domin...	67008	Red	California	Usa	2001	false	93.6	B	10
11	Napa Valley	Beringe...	11029	Red	California	Usa	2001	false	93.25	C+	3
12	Sonoma County	Verite, ...	139098	Red	California	Usa	2001	false	92.37	C	3

Database - continued

127.0.0.1:58973/browser/

pgAdmin File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents Wine_join.sql

postgres/postgres@wine

Query Editor Query History

```
1 select * into Red_soil_table
2 from red_table
3 inner join soil_table
4 using(appellation);
5
6 select * from Red_soil_table;
7
8 select * into White_soil_table
9 from white_table
10 inner join soil_table
11 using(appellation);
12
13 select * from White_soil_table;
```

Scratch Pad

Data Output Explain Messages Notifications

	appellation	wine	wine_id	color	regions	country	vintage	is_primeurs	score	confidence_index	journalist_count	av
	text	text	integer	text	text	text	integer	boolean	double precision	text	integer	do
1	Santa Cruz Mou...	Mount ...	107658	White	California	Usa	2015	false	92.22	B		4
2	Napa Valley	Pahime...	111897	White	California	Usa	2015	false	92.83	C+		4
3	Sonoma Coast	Marcas...	101640	White	California	Usa	1993	false	92.07	C		3
4	Sonoma Coast	Marcas...	101640	White	California	Usa	1998	false	91.74	B		4
5	Napa Valley	Kongsg...	91591	White	California	Usa	2015	false	97.27	B+		4
6	Sonoma County	Peter ...	114834	White	California	Usa	2015	false	95.69	C+		3
7	Napa Valley	Kongsg...	91585	White	California	Usa	2015	false	94.84	B		5
8	Sonoma County	Peter ...	114819	White	California	Usa	2016	false	94.12	B		3
9	Sonoma County	Peter ...	114832	White	California	Usa	2016	false	94.12	B		3
10	Russian River V...	Martin...	102656	White	California	Usa	2007	false	91.83	A		4
11	Napa Valley	Alpha ...	3221	White	California	Usa	2015	false	91.18	B		3
12	Knights Valley	Peter ...	114830	White	California	Usa	2015	false	94.72	B		4

Machine Learning - Data Preprocessing

- Checked for null values
- Checked the data types
- Converted the score column from float to integer and split score into good(1) which is any wine with a score ≥ 91 and bad(0) and making it its own column "quality" to use as our target.
- Checked for number of unique values in each column to find out which columns required binning and binned appellation.
- Created the OneHotEncoder instance, Fitted the encoder and produce encoded DataFrame and renamed encoded columns.
- Merged one-hot encoded features and drop the originals

Machine Learning - Prelim Feature Engineering and Selection

Utilized the wine score to determine the quality of wine and for feature we decided to look at wine data by itself and dropping all weather and soil data columns and unnecessary columns from the wine data, wine data with weather data, wine data with soil data and wine data with weather and soil data.

Machine Learning - Training and Test Data

Used sklearn `train_test_split` to split the dataset into random train and test subsets.

Machine Learning - Model types

- Deep Learning Neural Network - The limitation of the model is that it requires a large amount of data and it's not easy to comprehend. The benefits of this model can solve complex problems.
- Random Forest Classifier - The limitation is that features need to have some predictive power to work. The benefit is handling of huge amount of data, No problem of overfitting
- Logistic Regression - The limitation of the model is that it can be easily outperformed by sturdier model like Neural Networks, also its high reliance on a proper presentation of your data. The benefits of this model are that it's easier to implement, very efficient to train and it outputs well-calibrated predicted probabilities.

Machine Learning - Deep Learning Neural Network

	Loss	False Negativves	False Positives	True Negatives	True Positives	Precision	Recall	Accuracy
Red wine	0.83	79	43	306	597	0.93	0.88	0.88
Red wine with weather	0.87	77	60	289	599	0.91	0.89	0.87
Red wine with soil	0.66	66	43	306	610	0.93	0.90	0.89
Red wine with weather & soil	0.80	70	58	291	606	0.91	0.90	0.88

	Loss	False Negativves	False Positives	True Negatives	True Positives	Precision	Recall	Accuracy
White wine	0.75	27	8	67	81	0.91	0.75	0.80
White wine with weather	0.94	31	6	69	77	0.92	0.71	0.80
White wine with soil	0.74	30	8	67	78	0.90	0.72	0.79
White wine with weather & soil	0.75	31	7	68	77	0.91	0.71	0.79

Machine Learning - Random Forest Classifier

Machine Learning - Logistic Regression

Analysis

Logistic Regression model as the best fit model for this analysis after trying multiple models and considering the structure of our data to help answer our questions. Will add more here

Description of the source of data

Wine API - <https://www.globalwinescore.com/api/>

Weather API - <https://www.ncei.noaa.gov/access/search/data-search/global-summary-of-the-year>

Weather Analysis - <https://www.evineyardapp.com/blog/2019/01/17/climate-weather-and-vineyard-management/>

All data source will be adding to this slide.

