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 ?

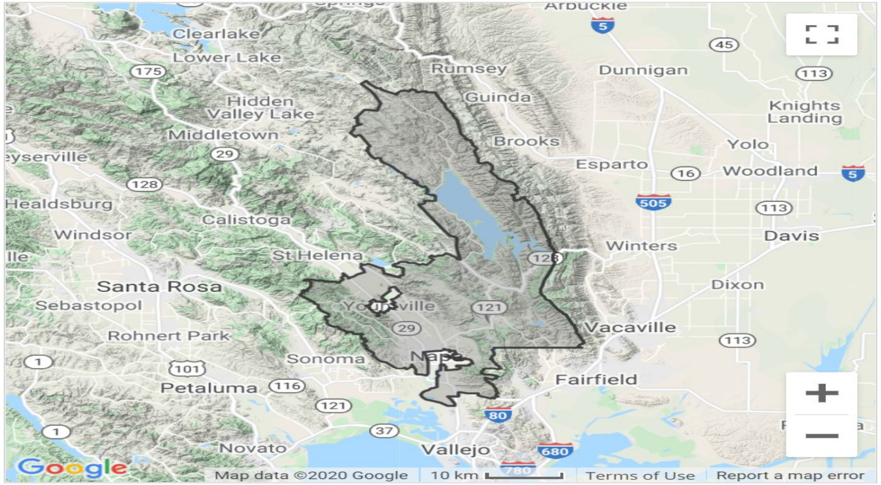

#YOUNGERTV
I'LL BRING THE WINE

Data Exploration

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

PERIOD OF RECORD	
Start¹	1906-04-01
End¹	2020-11-05
Coverage²	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	COVERAGE²
TAVG	Average Temperature.	1998-05-22	2005-07-31	100%
TMAX	Maximum temperature	1906-04-01	2020-11-05	100%
TMIN	Minimum temperature	1906-04-01	2020-11-05	100%
TOBS	Temperature at the time of observation	1958-05-01	2020-11-04	98%

Data Exploration - Continued

```
In [ ]: willamettedf = pd.DataFrame()
        states = {
        #     'Napa': {
        #         'zip': '94559',
        #         'years': ['1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','20
        #         },
        #     'Walla': {
        #         'zip': '99362',
        #         'years': ['1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','20
        #         },
        #     'Columbia': {
        #         'zip': '98813',
        #         'years':['1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','200
        #         },
        #     'Sonoma': {
        #         'zip': '95476',
        #         'years': ['1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','20
        #         },
        #     'Santa': {
        #         'zip': '95062',
        #         'years': ['1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','20
        #         },
        #     'Yakima': {
        #         'zip': '98903',
        #         'years':['1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','200
        #         },
        #     'Dundee': {
        #         'zip': '97148',
        #         'years': ['1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','20
        #         },
            'Willamette': {
                'zip': '97302',
                'years': ['1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','2007'
                }
        }
```

Data Exploration - Continued

Clear all × Usa × California ×

Show 25 entries

Search: Type appellation, wine, sc

GLOBAL WINE SCORE

74 100

CATEGORY

- Latest
- En primeur

COUNTRY

× Usa

Napa Valley
Russian River Valley
Oakville
Carneros
Howell Mountain
Sonoma County
Rutherford
Sonoma Coast

COLOR

	#	Wine	Vintage	GWS	CI	NBJ
1	1.	Kapcsandy Family Winery, State Lane Vineyard Grand-Vin Cabernet Sauvignon, Napa Valley	2015	**100**	B	4
2	2.	Abreu Vineyard, Thorevilos Cabernet Sauvignon, Napa Valley	2013	**100**	A	3
3	3.	Continuum, Proprietary Red, Oakville	2015	**99.85**	C	3
4	4.	Ridge Vineyards, Monte Bello, Santa Cruz Mountains	2015	**99.71**	C+	4
5	5.	Abreu Vineyard, Madrona Ranch Cabernet Sauvignon, Napa Valley	2002	**99.7**	A+	3
6	6.	Abreu Vineyard, Thorevilos Cabernet Sauvignon, Napa Valley	2007	**99.52**	A+	3
7	7.	Lokoya Winery, Mount Veeder Cabernet Sauvignon, Napa Valley	2015	**99.5**	C+	4
8	8.	Abreu Vineyard, Thorevilos Cabernet Sauvignon, Napa Valley	2015	**99.43**	A+	5
9	9.	Screaming Eagle, Cabernet Sauvignon, Napa Valley	2016	**99.43**	B	3
10	10.	Verite, La Joie, Sonoma County	2015	**99.37**	C+	4
11	11.	Screaming Eagle, Cabernet Sauvignon, Napa Valley	2010	**99.31**	A	6
12	12.	Dana Estates, Lotus Vineyard Cabernet Sauvignon, Napa Valley	2010	**99.08**	B	3
13	13.	Verite, La Muse, Sonoma County	2007	**99.04**	B	3

Data Exploration - Continued

Loop through API call for appellations and add to DataFram

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
```

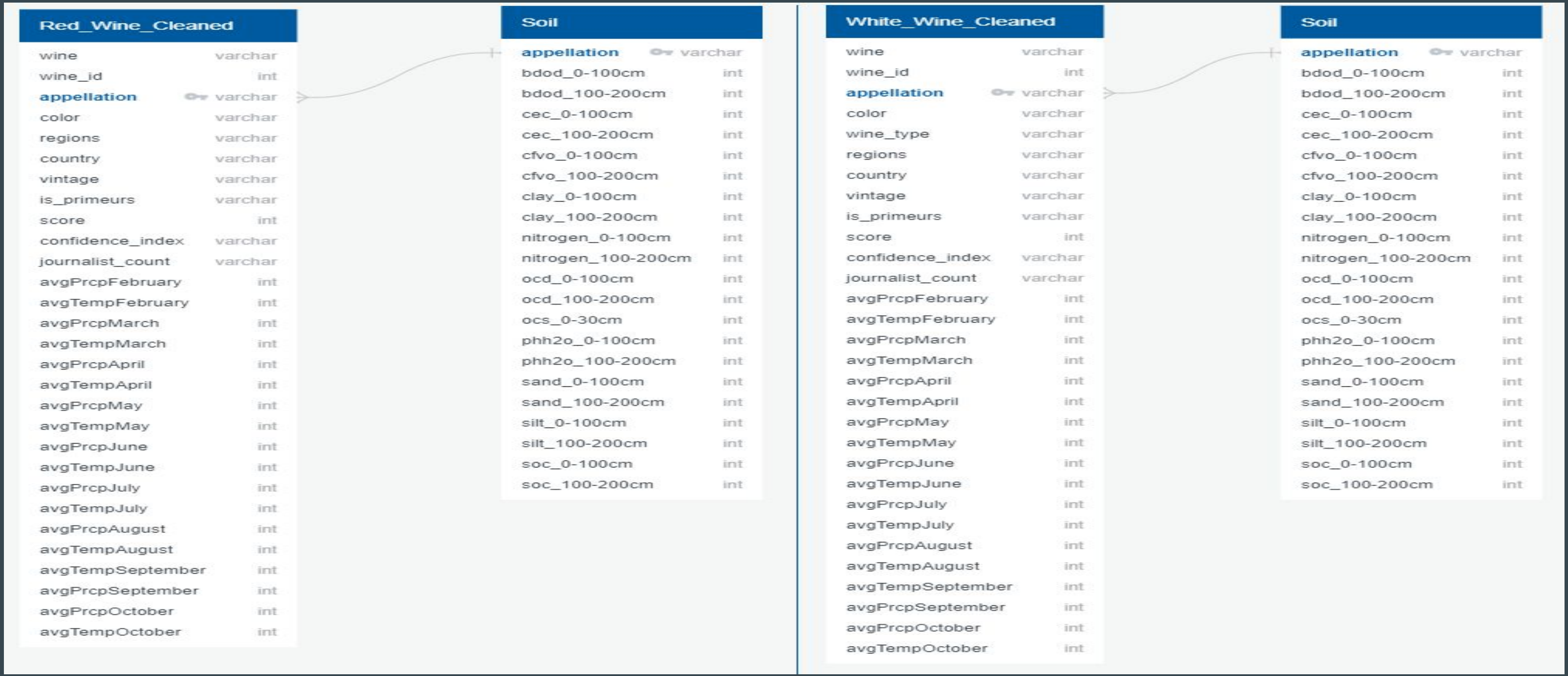
Data Exploration - Continued

```
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']['layers'][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 measurments 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']['layers'][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
```

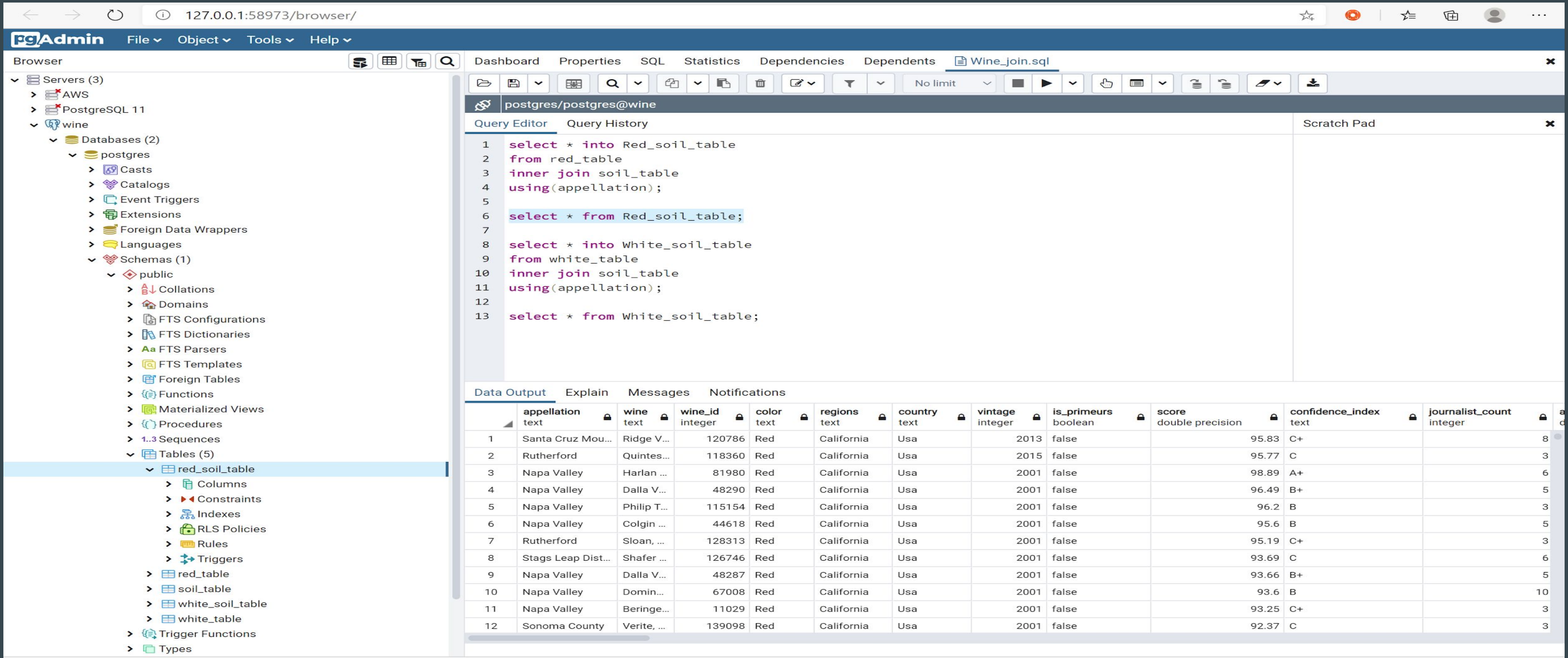
Database

- 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

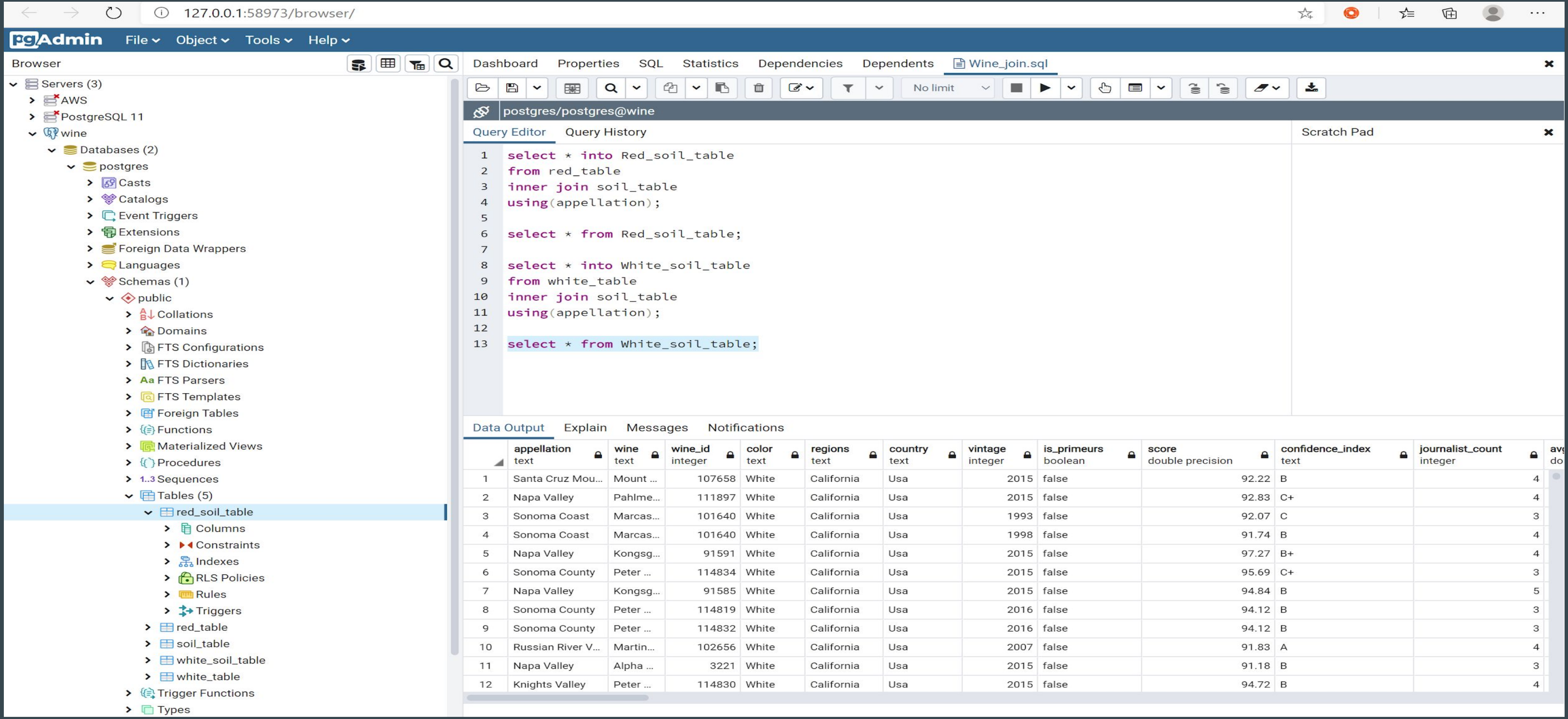
Database - continued



Database - continued



Database - continued




```sql
select * into Red_soil_table
from red_table
inner join soil_table
using(appellation);

select * from Red_soil_table;

select * into White_soil_table
from white_table
inner join soil_table
using(appellation);

select * from White_soil_table;
```

	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...	Mount ...	107658	White	California	Usa	2015	false	92.22	B	4
2	Napa Valley	Pahlme...	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

Red_Wine_DLNN_metrics

	Loss	False Negatives	False Positives	True Negatives	True Positives	Precision	Recall	Accuracy
Red wine	0.537	86	148	201	590	0.799	0.873	0.771
Red wine with weather	0.708	125	142	207	551	0.795	0.815	0.740
Red wine with soil	0.531	101	144	205	575	0.800	0.851	0.761
Red wine with weather & soil	0.568	125	132	217	551	0.807	0.815	0.749

White_Wine_DLNN_Metrics

	Loss	False Negatives	False Positives	True Negatives	True Positives	Precision	Recall	Accuracy
White wine	0.670	27	18	57	81	0.818	0.75	0.754
White wine with weather	0.768	27	17	58	81	0.827	0.75	0.76
White wine with soil	0.585	28	14	61	80	0.851	0.741	0.770
White wine with weather & soil	0.685	24	15	60	84	0.848	0.778	0.787

Machine Learning - Random Forest Classifier

Red_Wine_RF_Metrics

	Loss	False Negatives	False Positives	True Negatives	True Positives	Precision	Recall	Accuracy
Red wine	N/A	103	137	212	573	0.807	0.848	0.766
Red wine with weather	N/A	105	160	189	571	0.781	0.845	0.741
Red wine with soil	N/A	103	138	211	573	0.806	0.848	0.765
Red wine with weather & soil	N/A	106	164	185	570	0.777	0.843	0.737

White_Wine_RF_Metrics

	Loss	False Negatives	False Positives	True Negatives	True Positives	Precision	Recall	Accuracy
White wine	N/A	28	17	58	80	0.825	0.741	0.754
White wine with weather	N/A	29	18	57	79	0.814	0.731	0.743
White wine with soil	N/A	28	17	58	80	0.825	0.741	0.754
White wine with weather & soil	N/A	32	19	56	76	0.8	0.704	0.721

Machine Learning - Logistic Regression

Red_Wine_LR_Metrics

	Loss	False Negatives	False Positives	True Negatives	True Positives	Precision	Recall	Accuracy
Red wine	N/A	92	154	195	584	0.791	0.864	0.760
Red wine with weather	N/A	82	151	198	594	0.797	0.879	0.773
Red wine with soil	N/A	92	154	195	584	0.791	0.864	0.760
Red wine with weather & soil	N/A	82	147	202	594	0.802	0.879	0.777

White_Wine_LR_Metrics

	Loss	False Negatives	False Positives	True Negatives	True Positives	Precision	Recall	Accuracy
White wine	N/A	21	17	58	87	0.837	0.806	0.792
White wine with weather	N/A	20	19	56	88	0.822	0.815	0.787
White wine with soil	N/A	21	17	58	87	0.837	0.806	0.792
White wine with weather & soil	N/A	20	19	56	88	0.822	0.815	0.787

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. With red soil and weather data it predicted 655 good wines and that was also actual and 323 Bad wines and that was actual. 26 times it predicted it to be a good wine but it was a bad wine and 21 times we predicted it to be a bad wine when it was a good wine . This logistic regression had the highest accuracy / Tableau

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.