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 ?

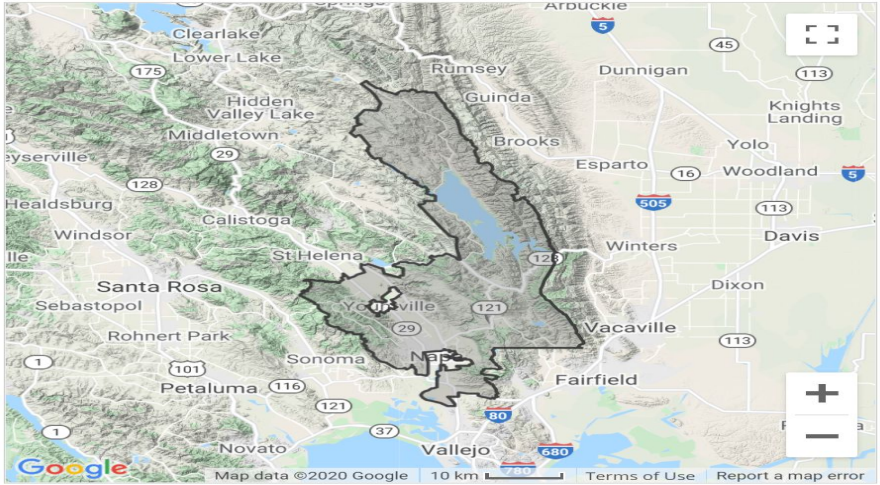

#YOUNGERTV
I'LL BRING THE WINE

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

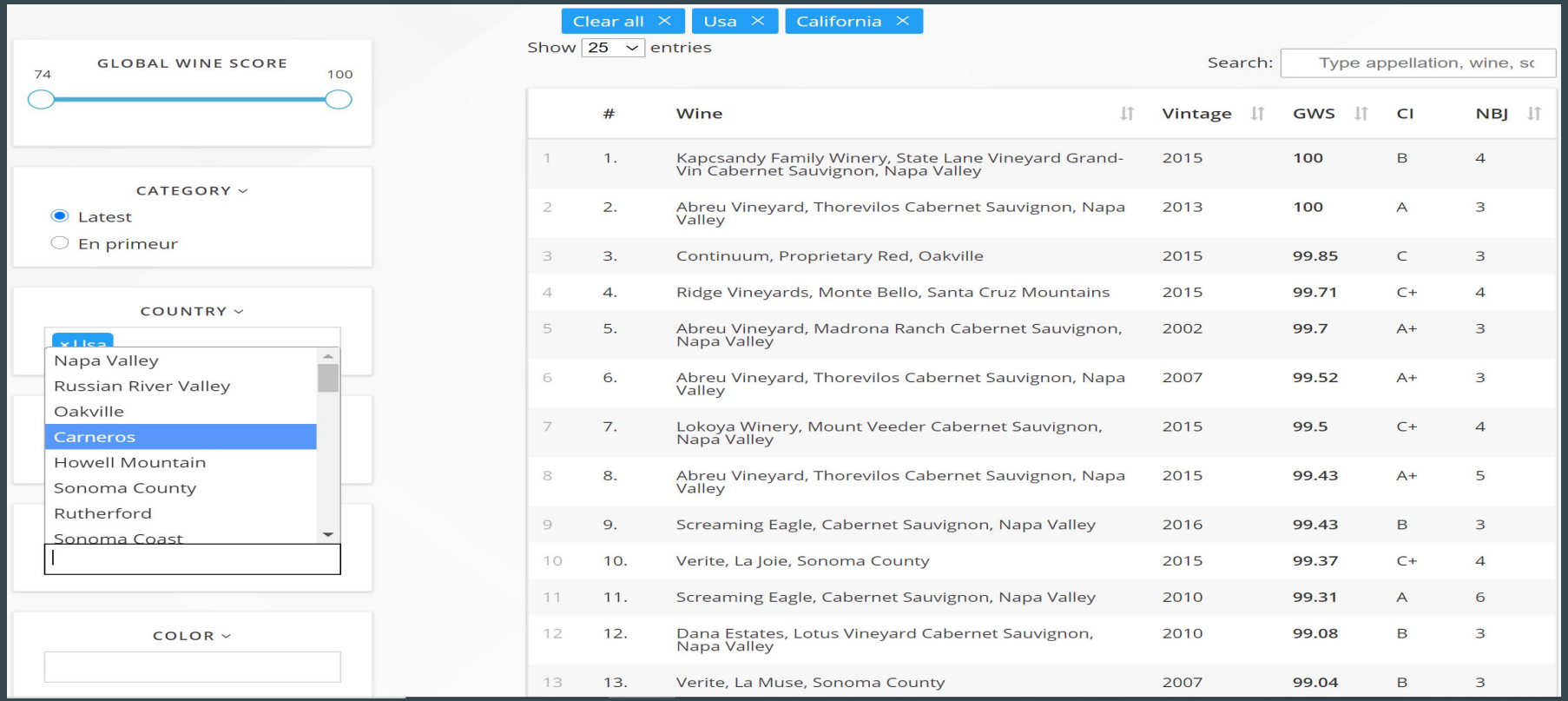
DATA TYPE	DESCRIPTION	START	END	COVERAGE[2]
TAVG	Average Temperature.	1998-05-22	2005-07-31	100%
TMAX	Maximum temperature	1906-04-01	2020-11-05	100%
TMIN	Minimum temperature	1906-04-01	2020-11-05	100%
TOBS	Temperature at the time of observation	1958-05-01	2020-11-04	98%

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



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']
```

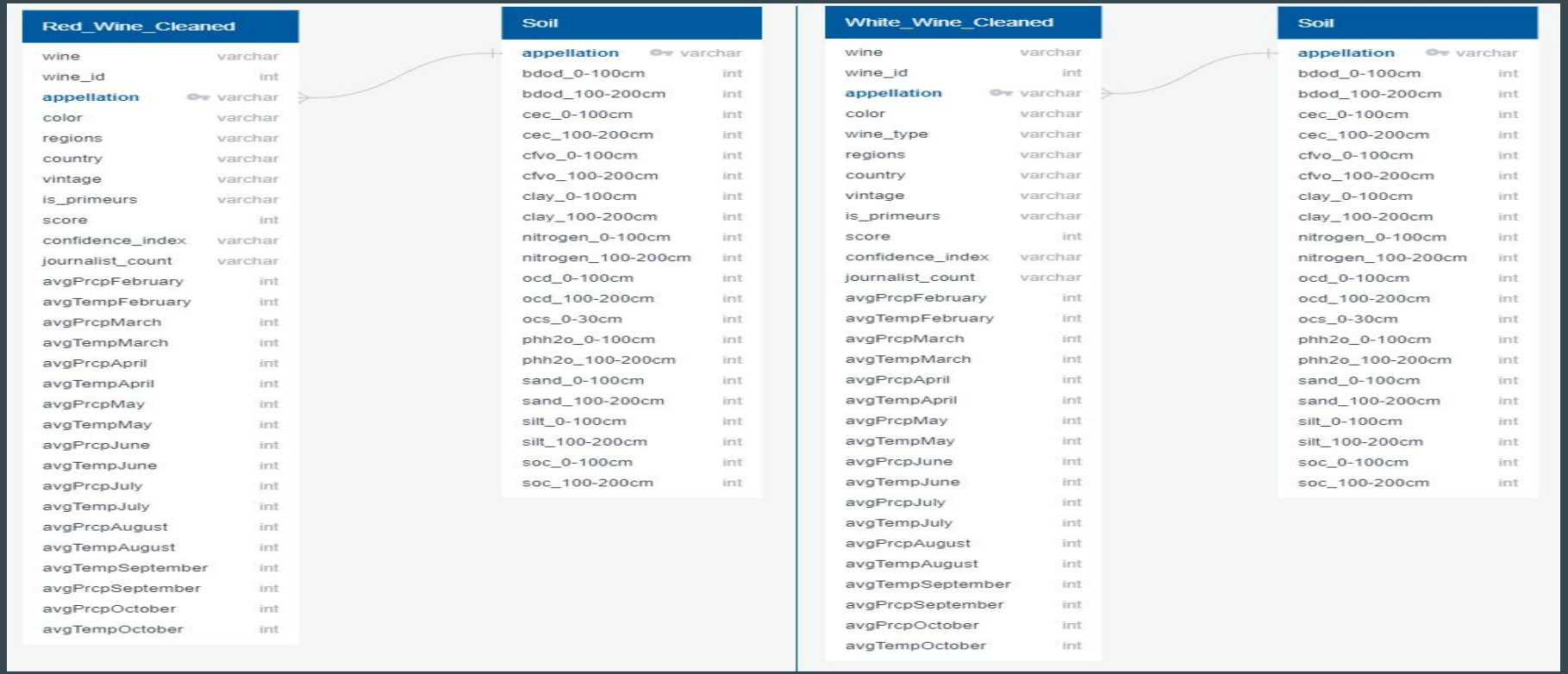
In [11]:
```
# Create list of weights for measurements for use with calculations
soils_range_weight = [0.05, 0.1, 0.15, 0.3, 0.4]
```

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
```

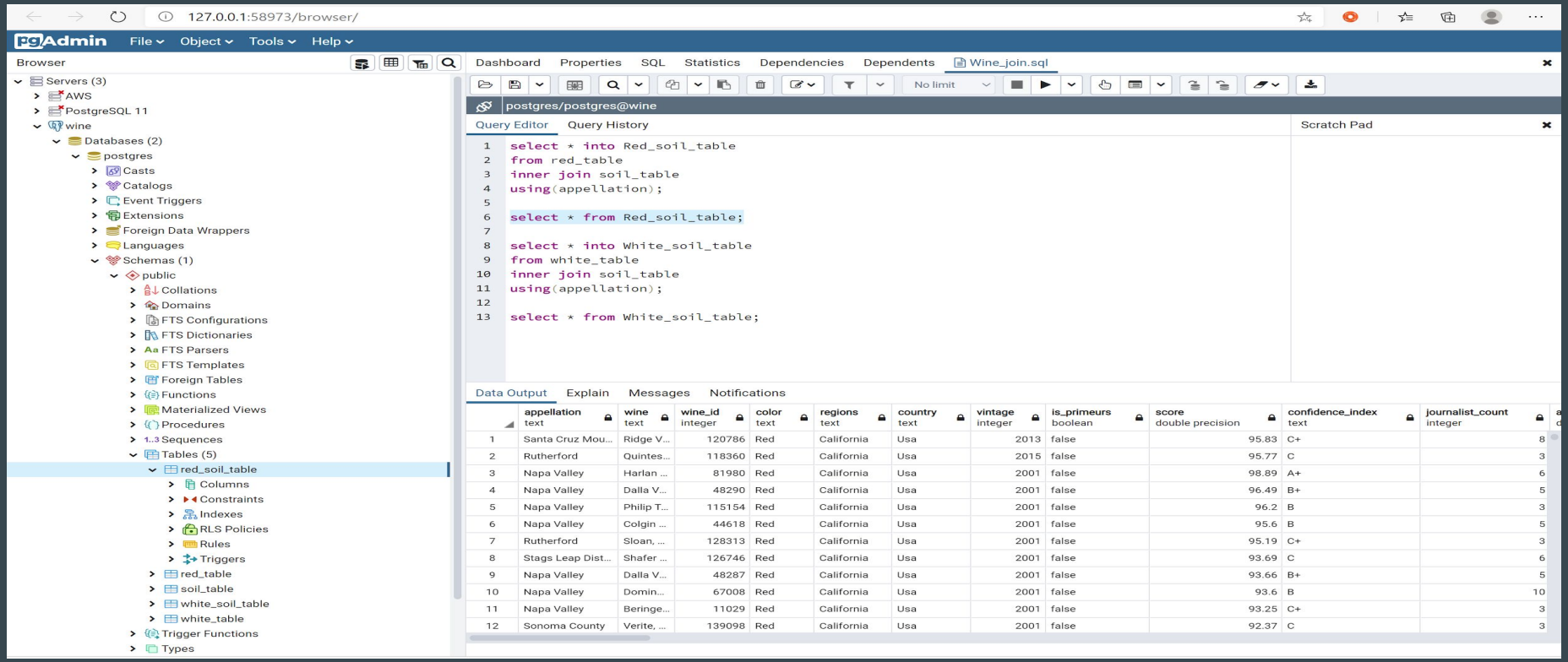
Database

- 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

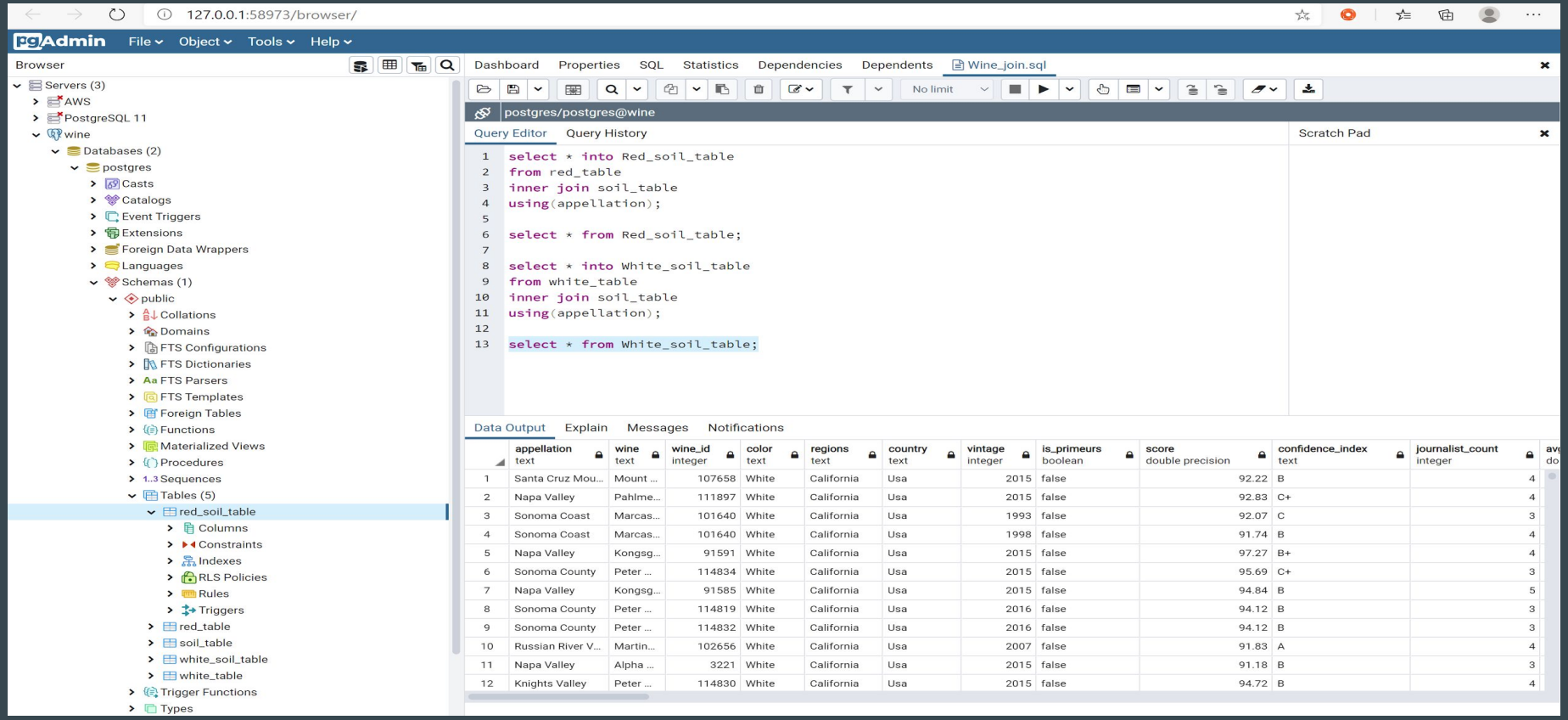
Database - continued



Database - continued



Database - continued



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.