Methods

This notebook contains all of the code for generating the plots found in our write-up on the github page. Note that some of the plots render here, and others are found in the github pages.

Section 0: Imports

In [1]:
```
#data wrangling
import pandas as pd
import numpy as np

#statistical packages
from scipy import stats
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.metrics import mean_squared_error

#visualization packages
import seaborn as sns
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
import matplotlib.pyplot as plt

#api data-pulling packages/other misc
from ucimlrepo import fetch_ucirepo
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import warnings
warnings.filterwarnings("ignore")
```

Section 1: Learning to Taste Dataset visualizations

A) Word-Cloud Visualization [rendered]
B) Histogram of Ratings [github]
C) Wine Ratings by Country [github]
D) Wine alcohol content by Country and Year [github]
E) Count of Wines by Country and Grape Type [github]

1A: Word Cloud Visualization

The word cloud visualization was produced using this tutorial: https://www.datacamp.com/tutorial/wordcloud-python. It involved loading in the data, selecting the reveiew column, and utilizing the word cloud package to generate the image.

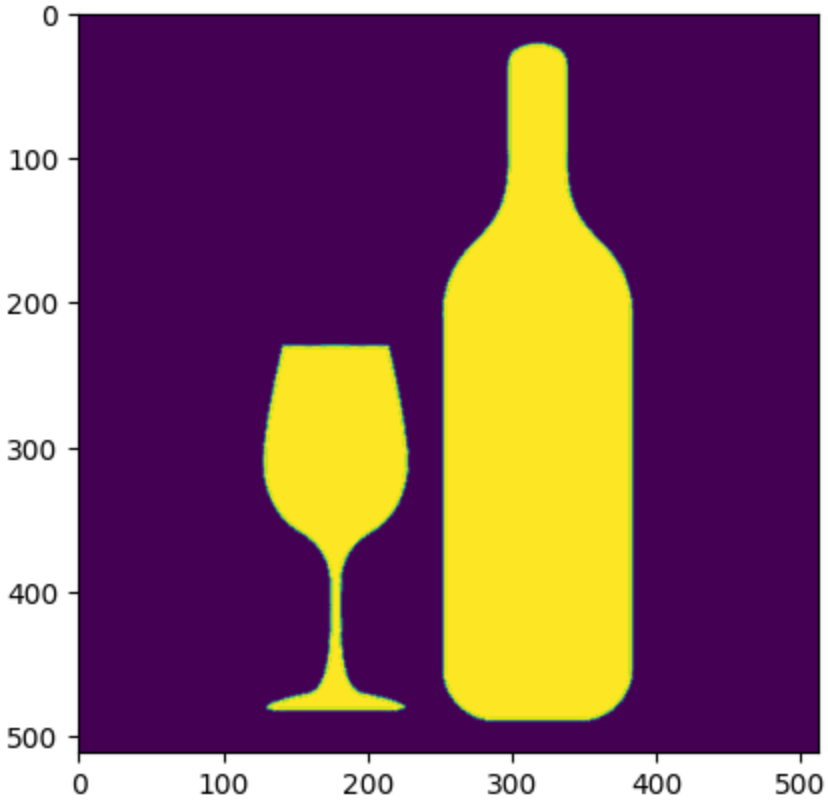
In [2]:
```
# Loading in data
images_reviews = pd.read_csv('/Users/isabellabaldacci/desktop/imageanalysis/data/images_reviews_attributes.csv')
df = images_reviews

#define text for wordcloud
text = df.review[0]

# Create and generate an initial word cloud image
wordcloud = WordCloud().generate(text)
```

In [3]:
```
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

While this was a good start, we noticed that the reviews section had many NAs, so we dropped the NAs and created a large string for the final output. We also wanted to make the shape of the word cloud more visually appealing, so loaded a mask to overlay the image. This involved transforming the pixel values to a smaller range.

In [4]:
```
#drop nas and join all reviews together to form long string
reviews = df['review'].dropna()
text = " ".join(review for review in reviews)

#open wine mask
wine_mask = np.array(Image.open("/Users/isabellabaldacci/desktop/imageanalysis/wine_mask.png"))
plt.imshow(wine_mask)
```

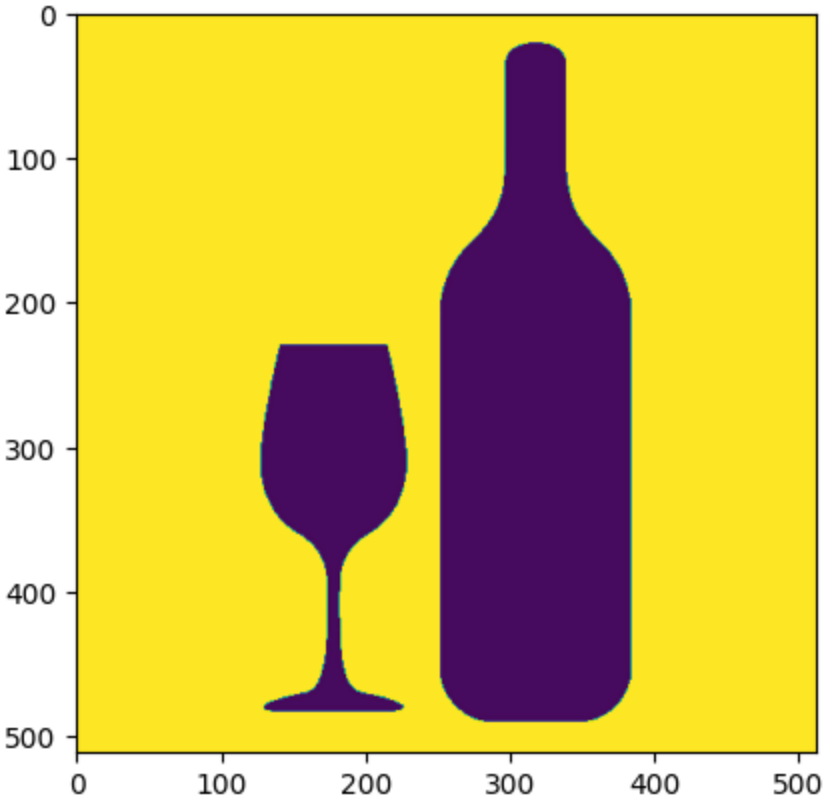
Out[4]:
```
<matplotlib.image.AxesImage at 0x15ffc7e00>
```



In [5]:
```python
#transforming wine mask
def transform_format(val):
    '''
    Transforms single value from image
    Args: val (int) - number to be assessed
    Returns: val (int) - if not 0, else returns 255
    '''
    if val == 0:
        return 255
    else:
        return val

# Transform your mask into a new one that will work with the function:
```

```
transformed_wine_mask = np.ndarray((wine_mask.shape[0],wine_mask.shape[1]), np.int32)

for i in range(len(wine_mask)):
    transformed_wine_mask[i] = list(map(transform_format, wine_mask[i]))
plt.imshow(transformed_wine_mask)
```

Out[5]: <matplotlib.image.AxesImage at 0x16000b6b0>



Once the mask was prepared and the NAs had been removed, we were able to plot the word cloud and store to a file.

In [6]:
```
wc = WordCloud(background_color="white", max_words=1000, mask=transformed_wine_mask,
               contour_width=3, contour_color='firebrick')

# Generate a wordcloud
```

```
wc.generate(text)

# store to file
#wc.to_file("wine.html")

# show
plt.figure(figsize=[20,10])
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```


complex
bit
good wine
Great wine
fruity
medium bodied
little
dry
aroma
time
much
taste
palate
full bodied
flavour
rich
elegant
still
high acidity
delicious
soft
money
Pinot Noir
black cherry
bold
smooth
bottle
spice
medium acidity
mouth
perfect
wine
price
better
red fruit
nose
long finish
Good value
open
smell
glass
hint
year
fresh
nice
bad
dark fruit
s
full body
strong
medium body
sweet
one
lot
touch
fruit forward
black fruit
green apple
Lovely
love

1B: Histogram of Ratings

We used the same dataset to visualize the distribution of wine ratings in the dataset.

In [7]:
```
# Create a subplot with 2 rows
fig = make_subplots(rows=2, cols=1,
                    specs=[[{"type": "box"}], [{"type": "histogram"}]],
                    subplot_titles=("Ratings", "Histogram of Ratings"))
# Add horizontal box plot
fig.add_trace(go.Box(y=df['rating'], name='Box Plot', orientation='h', boxmean='sd'), row=1, col=1)
# Add histogram
fig.add_trace(go.Histogram(x=df['rating'], nbinsx=12, histnorm='probability density',
                           name='Histogram', opacity=0.75), row=2, col=1)
# Update layout
fig.update_layout(title='Horizontal Box Plot and Histogram of Ratings',
                  xaxis_title='Rating',
                  yaxis_title='Density',
                  showlegend=True)
#commented out so that it is not rewriting a figure every time
#fig.write_html('horizontal_box_histogram.html')
```

1C: MapViz - Wine Ratings by Country

To view wine ratings by country, we first had to find the average rating of wine by country. We also included the wine_alcohol percent as part of the visualization, despite the lack of a value for South Africa. This was accomplished through grouping by country and using aggregation metric "mean". We could then plot on a world map using plotly express.

In [8]:
```
#group by country and take the average of the rating and wine alcohol content
average_ratings = df.groupby('country')[['rating','wine_alcohol']].mean().reset_index()
```

```
#round to two decimal places for each for readability
average_ratings['rating'] = average_ratings['rating'].round(2)
average_ratings['wine_alcohol'] = average_ratings['wine_alcohol'].round(2)

#display
print(f'checking alc % in South Africa: {images_reviews[images_reviews['country']=='South Africa']['wine_alcohol'].u
display(average_ratings)
```

```
checking alc % in South Africa: [nan]
```

	country	rating	wine_alcohol
0	Argentina	4.25	13.23
1	Australia	4.08	14.55
2	France	4.09	13.93
3	Italy	4.21	14.17
4	Portugal	4.00	14.61
5	South Africa	4.30	NaN
6	Spain	4.15	13.69
7	United States	3.99	13.99

Even though South Africa did not have a value for the wine alcohol content, the rest of the data were displayed on a map colored by rating.

In [ ]:
```
#figure code
fig = px.choropleth(
    average_ratings,
    locations='country',         # Country names
    locationmode='country names', # Use country names
    color='rating',              # The average rating to color countries
    hover_name='country',        # Hover text shows the country name
    hover_data = {
        'rating': True,           # Show average rating
```

```
        'wine_alcohol': True,     # Show average wine alcohol
        'country': False          # Do not show country name again
    },
    color_continuous_scale=px.colors.sequential.Plasma,  # Choose a color scale
    labels={'rating': 'Average Rating'},  # Axis labels
    title='Average Wine Ratings by Country'
)

# Update layout for better aesthetics
fig.update_geos(
    projection_type="natural earth",  # Change the projection if needed
)
fig.update_layout(
    title='Average Wine Ratings by Country',
    geo=dict(showland=True, landcolor='lightgray',
             subunitcolor='white', countrycolor='white'),  # Map aesthetics
)
fig.show(renderer = 'notebook')
#commented out so that it doesn't write every time
#fig.write_html('globe_map.html')
```

1D: Wine alcohol content by Country and Year in relationship to price

The following plots reviews by year and price vs the wine alcohol content in an interactive way.

In [ ]:
```
#make the scatterplot,
# setting frame to year to group by year
# and group to region to group by region,
# color by region
fig = px.scatter(df, x="price", y="wine_alcohol", animation_frame="year", animation_group="region",
           size="rating", color="region", hover_name="region")

fig["layout"].pop("updatemenus")
fig.show()
```

1E: Count of Wines by Country and Grape type

For a visualization of the count of wines by country and grape type, we dropped the na values from the grape and country columns. We then grouped by country and grape, aggregated by size, and reset the index to be count. This was visualized as an interactive stacked barchart.

In [28]:
```
df_grape = df.dropna(subset=['grape', 'country'])
counts = df_grape.groupby(['country', 'grape']).size().reset_index(name='count')

custom_colors = [
    '#8B0000',  # Dark Red
    '#B22222',  # Firebrick
    '#A52A2A',  # Brown
    '#DC143C',  # Crimson
    '#C71585',  # Medium Violet Red
    '#FF4500',  # Orange Red
    '#FF0000',  # Red
    '#FF6347',  # Tomato
    '#FF7F50',  # Coral
    '#FF1493',  # Deep Pink
    '#CD5C5C',  # Indian Red
    '#D2691E',  # Chocolate
    '#F08080',  # Light Coral
    '#FA8072',  # Salmon
    '#FFB6C1',  # Light Pink
    '#FF69B4',  # Hot Pink
    '#C71585',  # Medium Violet Red
    '#FF81A0',  # Lightly Red
    '#FF6F61',  # Light Coral Red
    '#F75D59',  # Soft Red
    '#FF4F4F',  # Pure Red
    '#FF9999',  # Very Light Red
    '#D50000',  # Red A700
    '#BF360C',  # Red A400
    '#E53935',  # Red A200
    '#C62828',  # Red A100
    '#B71C1C',  # Dark Red
    '#FF8A80',  # Light Red
    '#D32F2F',  # Red
    '#D32F2F',  # Medium Red
```

```
    '#FF5252',  # Bright Red
]

fig = px.bar(counts,
             x='country',
             y='count',
             color='grape',
             title='Count of Wines by Country and Grape Type',
             labels={'count': 'Count of Wines', 'country': 'Country', 'grape': 'Grape Type'},
             color_discrete_sequence=custom_colors
            )

# Show the figure
fig.show()
#fig.write_html('bar_grape.html')
```

Section 2: Machine Learning Model for Quality prediction

A) Barplot of the number of wines with different quality scores [rendered]

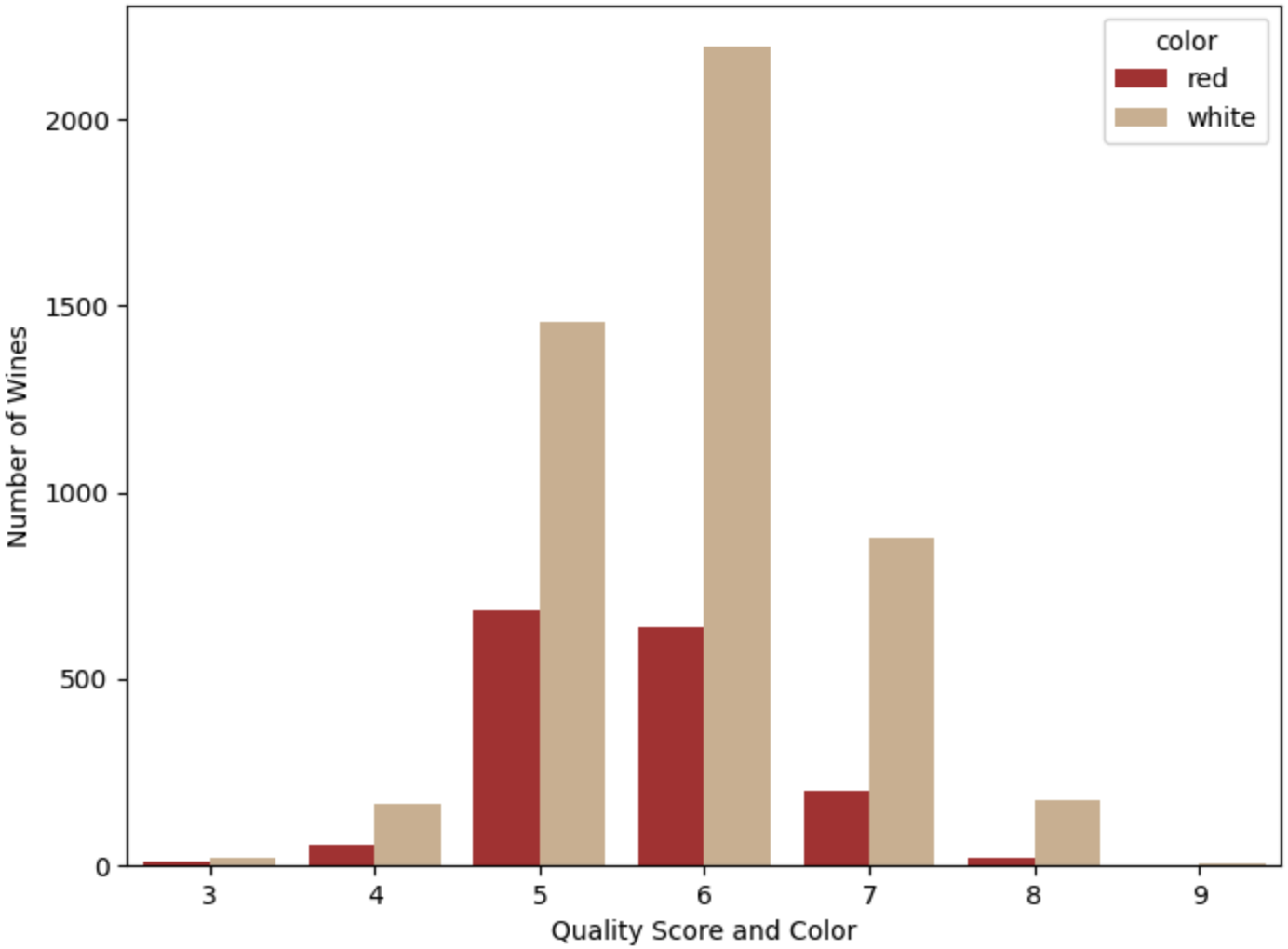
B) Feature Importance for wine quality prediction [github]

2A: Barplot of the number of wines with different quality scores

To get an idea of the distribution of the quality scores across the two wine types, we aggregated by color and quality and counted the number of wines for each quality score.

In [ ]:
```
# fetch dataset
wine_quality = fetch_ucirepo(id=186)

# data (as pandas dataframes)
X = wine_quality.data.features
y = wine_quality.data.targets

# view metadata
#print(wine_quality.metadata)
```

```
# view variable information
#print(wine_quality.variables)

wine_quality['data'].keys()
whole_dataset = wine_quality['data']['original']

grouped_color = whole_dataset.groupby(['color', 'quality']).count()

fig = plt.figure(figsize=(8, 6))
sns.barplot(data = grouped_color, x = 'quality', y = 'alcohol', hue = 'color', palette = ["#B22222","#D2B48C"])
plt.ylabel('Number of Wines')
plt.xlabel('Quality Score and Color')
#plt.savefig('/Users/isabellabaldacci/desktop/dist.png')
grouped_color_avg = whole_dataset.groupby(['color']).mean()
display(grouped_color_avg[['quality']])
```



2B: Feature importance for wine quality prediction

To determine feature importance for wine quality prediction, we first split the dataset into red and white separetly, and then into feature and target dataframes.

In [ ]:

```
# split the dataset into red and white wines
red_wine = whole_dataset[whole_dataset['color'] == 'red'].drop('color', axis = 1)
white_wine = whole_dataset[whole_dataset['color'] == 'white'].drop('color', axis = 1)

# defining features (predictors) and target variable
features = ['fixed_acidity', 'alcohol',
            'volatile_acidity', 'citric_acid',
            'residual_sugar', 'chlorides',
            'free_sulfur_dioxide', 'total_sulfur_dioxide',
            'density', 'pH', 'sulphates']

# separating features and target for each wine type
X_red = red_wine[features]
y_red = red_wine['quality']
X_white = white_wine[features]
y_white = white_wine['quality']

## INTIALIZING AND TRAINING ##
#red wine dataset
tree_red = DecisionTreeRegressor(random_state=42)
tree_red.fit(X_red, y_red)
feature_importances_red = pd.Series(tree_red.feature_importances_, index=features) # providing

#white wine dataset
tree_white = DecisionTreeRegressor(random_state=42)
tree_white.fit(X_white, y_white)
feature_importances_white = pd.Series(tree_white.feature_importances_, index=features)

## COMBINING FOR PLOTTING ##
feature_importances_df = pd.DataFrame({
    'Feature': features,
    'Red Wine': feature_importances_red,
    'White Wine': feature_importances_white
}).melt(id_vars='Feature', var_name='Wine Type', value_name='Importance') # converting df from

# custom colors
color = {"Red Wine": "#B22222", "White Wine": "#D2B48C"}
background = "#ffe6e6"
```

```
# using 'px.bar' API to create interactive bar chart
fig = px.bar(
    feature_importances_df,
    #defining features and importance
    x='Feature',
    y='Importance',
    #defining colors and groups
    color='Wine Type',
    barmode='group',
    title="Feature Importance Comparison for Red and White Wine",
    labels={'Importance': 'Feature Importance', 'Feature': 'Features'},
    color_discrete_map=color
)

# chart background color + text color
fig.update_layout(
    plot_bgcolor=background,
    paper_bgcolor=background,
    font=dict(color="black")
)

fig.show()


#fig.write_html('/Users/isabellabaldacci/desktop/feature_quality.html')
```

In [ ]: