Victoria University Business School

BCO7000

Business Analytics and Visualisation

Group Assessment

Instructor name: Dr. Amanda Cole

Group: Team 4

Name	**Student ID**
Hardik Nagpal	s8083963
Vrishank Mani	s4680492
M T N D Perera	s8075741

Table of Contents

Introduction

Winemaking is a popular activity across the globe and traces back to ancient times and countries such as France, Italy and Spain have earned much recognition for their winemaking activities (Hagan 2020). Amongst these countries, France takes a prominent place especially for producing French wines such as Merlot, Chardonnay, Pinot Noir, Sauvignon Blanc, etc. (Hagan 2020).

In the dynamic world of wine, producers, wine merchants, and consumers alike seek insights into the selection and procurement of wines. Many turn to the discerning palates of wine tasters, seasoned connoisseurs, and even innovative wine apps for guidance (Smith 2019).

Within this context, our chosen dataset encompasses a plethora of wine-related attributes, including variety, winery, price, and reviews from wine experts. This report aims to understand how wineries and wine merchants can optimise the wine selection process and customer's wine buying experience so that they can manage their inventory better to improve their revenue and customer satisfaction.

Executive Summary

This report was prepared in the context of a wine data distribution relating to wine and reviews from wine tasters were cleaned and analysed to predict the price of wine and to categorise wines priced below $200 as value picks and wines priced above $200 as premium picks to support and enhance the wine selection process and optimise inventory for wine merchants and wineries.

Key performance metrics were used to evaluate the developed models and visualisation techniques such as scatter plots and box plots were used to understand the findings effectively.

Some of the key findings are that the price distribution of most wines falls around the price range of 0 to $100. Furthermore, The US leads the way from the rest with over 50,000 wineries. It has more than twice the number of wineries than France and Italy combined.

Two models were created based on P-values and research. Based on the evaluation metrics, Model 2 outperformed Model 1 in terms of MAE, MSE, and R-squared. It had lower errors, explained a larger proportion of the variance in the dependent variable, and provided better predictive performance.

Problem Statement

The main business problem this report intends to identify is, “How can we optimise the wine selection process and customer’s wine buying experience, in order to curate an inventory that would ultimately lead to increased customer satisfaction and revenue of this particular winery?”. Today one of the key challenges that wineries face is inventory curation and understanding which wines must be stocked (Maurel et al. 2019). This is a critical decision since customer satisfaction and revenue depend heavily on this. Overstocking wine can cause financial strains and storage problems. Understocking wines and wine unavailability can cause customer dissatisfaction and lose revenue in this regard.

Furthermore, customers have a variety of tastes (Navarro, Pedraja-Inglesias and Marta 2010) and different financial budgets when making a purchasing decision regarding wine(Gustafson, Lybbert and Sumner 2016).

Based on this understanding, the main problem was further broken down into 2 sub problems to create a regression model and a classification model:

Sub-Problem 1: Regression Model: “How can we predict the price of wine effectively to enhance our inventory curation for better customer experiences and stock control?”

As mentioned above, predicting wine prices accurately could benefit wineries and ensure that these prices are marked according to the market demands and in a profitable manner. This would help to understand which wines must be stocked to increase revenue, customer satisfaction and manage storage in wineries.

Sub-Problem 2: Classification Model: “How can we classify wines into ‘value picks’ and ‘premium choices’ for our inventory, to cater to our customers' diverse preferences and budgets?” Since customers have different preferences and budgets, this would help wineries to curate their inventories in an efficient manner and help customers and even employees to easily identify which wine varieties to be suggested must faster to customers based on their budgets. This would also save time when providing recommendations and also enhance customer experience. For this

particular problem, we have made an assumption that wines priced above $200.00 is considered as premium picks and wines below $200 as value picks.

Methodology

Data Collection

The dataset used for this analysis is the "winemag-data-130k-v2.csv" file, which contains information about various wine varieties. The dataset includes information such as the grape name, the name of the winery that produced the wine, country of origin, review of the wine, points, price, the province which the wine is from, tasters name, etc.

The dataset and the relevant libraries were imported. The dataset was printed to validate this.

```
#Importing the required libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.ensemble import RandomForestRegressor
from statsmodels.stats.outliers_influence import variance_inflation_factor

#Importng and loading the dataset
data = pd.read_csv("/Users/teharaperera/Documents/VU/Sem 2/Block 3/BCO7000/Assessment/winemag-data-130k-v2.csv")

#Printing the first few lines of the dataset
data.head()
```

Figure 1

	Unnamed: 0	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	win
0	0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicc
1	1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Qui Avida
2	2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainst
3	3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Ju
4	4	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot Noir	Sw Che

Figure 2

Data Cleaning and Pre-Processing (Data Wrangling)

Handling Missing Values

Many missing values were identified.

```
#Printing all the column names
print(data.columns)

#Checking the dataset for any missing values:
missing_values = data.isnull().sum()
print(missing_values)
```

```
Index(['Unnamed: 0', 'country', 'description', 'designation', 'points',
       'price', 'province', 'region_1', 'region_2', 'taster_name',
       'taster_twitter_handle', 'title', 'variety', 'winery', 'Vintage(Year)'],
      dtype='object')
Unnamed: 0                  0
country                    63
description                 0
designation             37465
points                      0
price                    8996
province                   63
region_1                21247
region_2                79460
taster_name             26244
taster_twitter_handle   31213
title                       0
variety                     1
winery                      0
Vintage(Year)            4609
dtype: int64
```

Figure 3

Missing values were handled using imputation methods. For categorical columns, missing values were imputed using mode and numerical columns such as price and Vintage(Year) were imputed using the median and mode, respectively.

```
#Imputation using mode for the categorical values:
categorical_cols = ['country', 'designation', 'province', 'region_1', 'region_2', 'taster_name', 'taster_twitter_handle

for col in categorical_cols:
    data[col].fillna(data[col].mode()[0], inplace=True)

#Filling the missing values with the median/mode where necessary
data['price'].fillna(data['price'].median(), inplace=True)
data['Vintage(Year)'].fillna(data['Vintage(Year)'].mode()[0], inplace=True)

data = data.drop(['Unnamed: 0'],axis =1)
```

Figure 4

The missing values were rechecked to ensure that there were no missing values in the dataset.

```
#Checking the dataset for any missing values and verifying that there are no missing values:
missing_values = data.isnull().sum()
print(missing_values)
```

```
country                 0
description             0
designation             0
points                  0
price                   0
province                0
region_1                0
region_2                0
taster_name             0
taster_twitter_handle   0
title                   0
variety                 0
winery                  0
Vintage(Year)           0
dtype: int64
```

Figure 5

```
#Checking the data information again for verification:
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 14 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   country                129971 non-null  object 
 1   description            129971 non-null  object 
 2   designation            129971 non-null  object 
 3   points                 129971 non-null  int64  
 4   price                  129971 non-null  float64
 5   province               129971 non-null  object 
 6   region_1               129971 non-null  object 
 7   region_2               129971 non-null  object 
 8   taster_name            129971 non-null  object 
 9   taster_twitter_handle  129971 non-null  object 
 10  title                  129971 non-null  object 
 11  variety                129971 non-null  object 
 12  winery                 129971 non-null  object 
 13  Vintage(Year)          129971 non-null  object 
dtypes: float64(1), int64(1), object(12)
memory usage: 13.9+ MB
```

Figure 6

Feature Engineering:

A new column called 'Vintage(Year)' was created by extracting the vintage year from the 'title' column as this is an important criterion for wine analysis.

```
#Extracting the vintage year of the wines from the 'title' column and creating a new column
data['Vintage(Year)'] = data['title'].str.extract(r'(\d{4})')

#Printing the first few lines of the updated dataset
data.head()
```

ɔuntry	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title	variety	winery	Vintage(Year)
Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia	Etna	NaN	Kerin O'Keefe	@kerinokeefe	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicosia	2013
ɔrtugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro	NaN	NaN	Roger Voss	@vossroger	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Quinta dos Avidagos	2011
US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon	Willamette Valley	Willamette Valley	Paul Gregutt	@paulgwine	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainstorm	2013
US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan	Lake Michigan Shore	NaN	Alexander Peartree	NaN	St. Julian 2013 Reserve Late Harvest Riesling ...	Riesling	St. Julian	2013

Figure 7

Data Type Conversions

The 'Vintage(Year)' column, was converted from an object data type to an integer as year is an integer value.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 14 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   country                129971 non-null  object 
 1   description            129971 non-null  object 
 2   designation            129971 non-null  object 
 3   points                 129971 non-null  int64  
 4   price                  129971 non-null  float64
 5   province               129971 non-null  object 
 6   region_1               129971 non-null  object 
 7   region_2               129971 non-null  object 
 8   taster_name            129971 non-null  object 
 9   taster_twitter_handle  129971 non-null  object 
 10  title                  129971 non-null  object 
 11  variety                129971 non-null  object 
 12  winery                 129971 non-null  object 
 13  Vintage(Year)          129971 non-null  object 
dtypes: float64(1), int64(1), object(12)
memory usage: 13.9+ MB
```

Figure 8

```
#Converting the 'Vintage(Year)' column to an integer value
data['Vintage(Year)'] = data['Vintage(Year)'].astype(int)
```

```
#Verifying the data type has been changed
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 14 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   country                129971 non-null  object 
 1   description            129971 non-null  object 
 2   designation            129971 non-null  object 
 3   points                 129971 non-null  int64  
 4   price                  129971 non-null  float64
 5   province               129971 non-null  object 
 6   region_1               129971 non-null  object 
 7   region_2               129971 non-null  object 
 8   taster_name            129971 non-null  object 
 9   taster_twitter_handle  129971 non-null  object 
 10  title                  129971 non-null  object 
 11  variety                129971 non-null  object 
 12  winery                 129971 non-null  object 
 13  Vintage(Year)          129971 non-null  int64  
dtypes: float64(1), int64(2), object(11)
memory usage: 13.9+ MB
```

Figure 9

Detection of Potential Outliers

To identify any outliers present, boxplots were plotted and a few outliers were present.

```
#Selecting only the numerical columns to draw boxplots to identify any outliers in them
numerical_columns = data.select_dtypes(include=['int64', 'float64'])

#Creating the boxplot to check for outliers
plt.figure(figsize=(12, 8))
numerical_columns.boxplot()
plt.title("Boxplot of Numerical Columns")
plt.xticks(rotation=45)
plt.show()
```

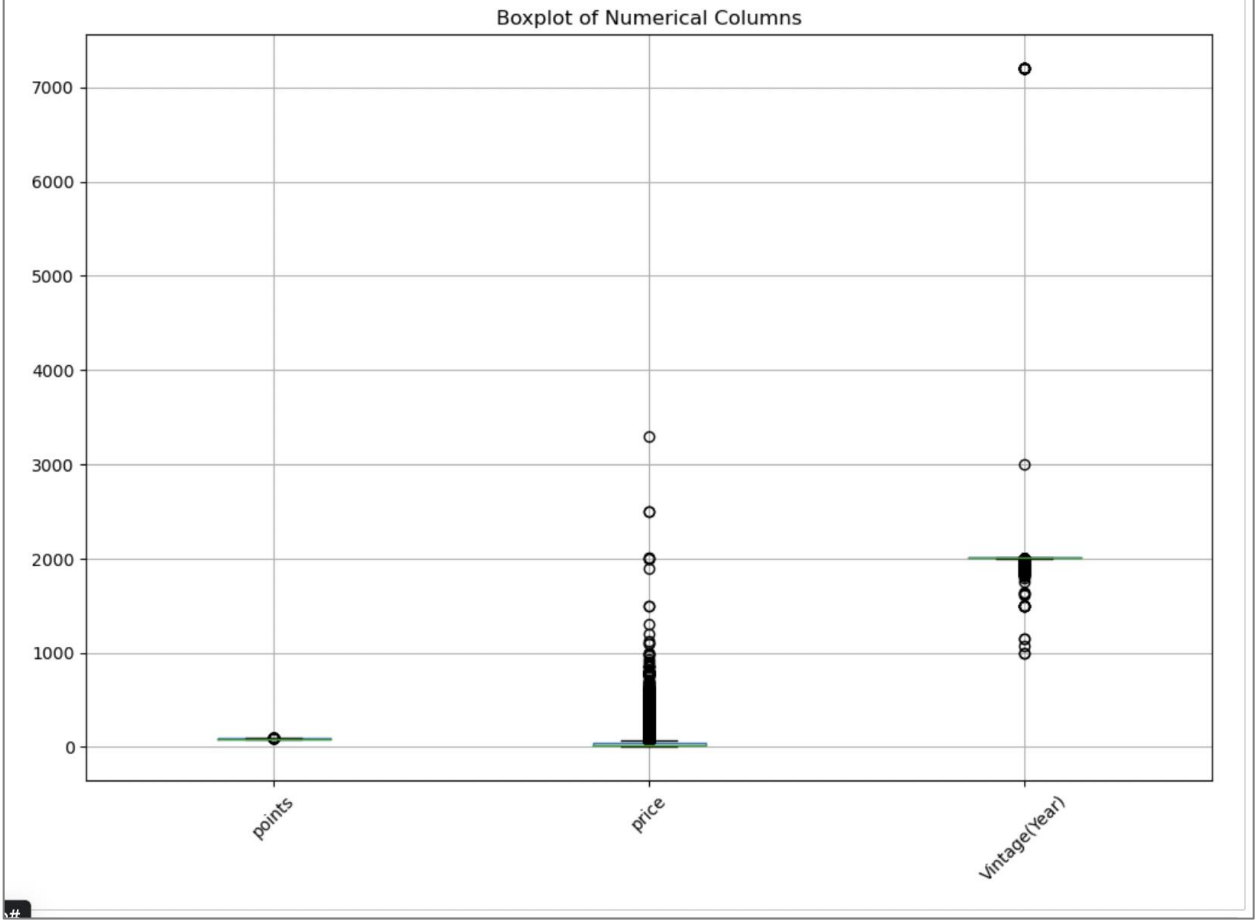

Figure 10

Data Exploration:

Summary statistics were analysed to obtain information about the features.

```
#Printing basic information about our dataset:
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 15 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   Unnamed: 0             129971 non-null  int64  
 1   country                129908 non-null  object 
 2   description            129971 non-null  object 
 3   designation            92506 non-null   object 
 4   points                 129971 non-null  int64  
 5   price                  120975 non-null  float64
 6   province               129908 non-null  object 
 7   region_1               108724 non-null  object 
 8   region_2               50511 non-null   object 
 9   taster_name            103727 non-null  object 
 10  taster_twitter_handle  98758 non-null   object 
 11  title                  129971 non-null  object 
 12  variety                129970 non-null  object 
 13  winery                 129971 non-null  object 
 14  Vintage(Year)          125362 non-null  object 
dtypes: float64(1), int64(2), object(12)
memory usage: 14.9+ MB
```

```
#Displaying the dataset summary statistics:
statistics = data.describe()
print(statistics)
```

```
          Unnamed: 0         points          price
count  129971.000000  129971.000000  120975.000000
mean    64985.000000      88.447138      35.363389
std     37519.540256       3.039730      41.022218
min         0.000000      80.000000       4.000000
25%     32492.500000      86.000000      17.000000
50%     64985.000000      88.000000      25.000000
75%     97477.500000      91.000000      42.000000
max    129970.000000     100.000000    3300.000000
```

Figure 11

To understand the dataset better, the minimum and maximum values were printed.

```
#Finding the minimum value in the 'price' column
min_price = data['price'].min()

#Finding the maximum value in the 'price' column
max_price = data['price'].max()

#Finding the minimum value in the 'points' column
min_points = data['points'].min()

#Finding the maximum value in the 'points' column
max_points = data['points'].max()

print(f"Minimum Price: {min_price}")
print(f"Maximum Price: {max_price}")
print(f"Minimum Points: {min_points}")
print(f"Maximum Points: {max_points}")
```

```
Minimum Price: 4.0
Maximum Price: 3300.0
Minimum Points: 80
Maximum Points: 100
```

Figure 12

The distribution of the variable's points, and price was plotted in an interactive plot for each variety.

```
# Creating an interactive box plot
fig = px.box(data, x="variety", y="price", title="Price Distribution by Wine Variety")

# Customising the layout
fig.update_xaxes(categoryorder="total ascending")  # Sort x-axis categories
fig.update_xaxes(title_text="Wine Variety")
fig.update_yaxes(title_text="Price")

# Showing the interactive plot
fig.show()
```

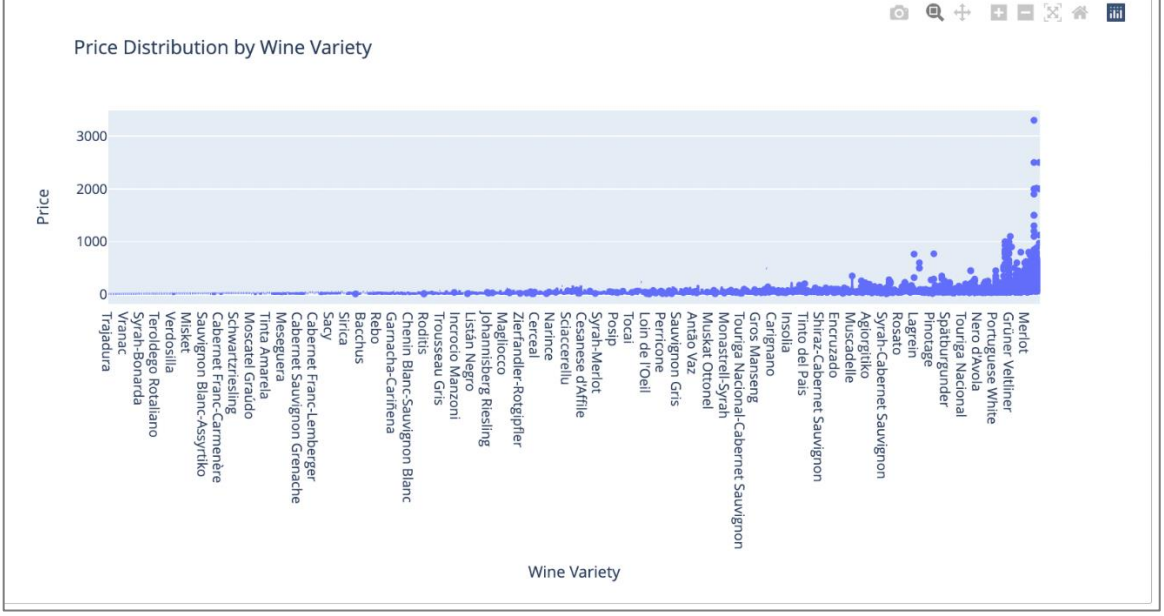

Figure 13

Assumptions and Limitations:

Assumptions

- Missing values were imputed using mode and median assuming that values are missing at random.
- The classification problem assumes that wine prices below $200.00 are considered as value picks and more than $200.00 are premium pick wines.

Data Analysis and Modelling

Regression and Classification Models Development Approach

The regression model was used to predict the price of wine and the classification model was developed to classify wines as ‘value picks’ and ‘premium picks’.

Developing the Regression Model for Sub Problem 1

Two regression models were developed and compared to select the best model.

A correlation matrix was created to understand the linear relationships with price. This shows that points have a relatively stronger correlation.

```
#Identifying which features have a linear relationship with the target variable(price)
correlation_matrix = data.corr()
correlation_with_price = correlation_matrix['price'].abs().sort_values(ascending=False)
print(correlation_with_price)
```

```
price            1.000000
points           0.399231
Vintage(Year)    0.003191
Name: price, dtype: float64
```

Figure 14

Categorical columns were encoded and verified as shown below.

```
#Creating a LabelEncoder instance
label_encoder = LabelEncoder()

#Label encoding categorical variables
data['variety_encoded'] = label_encoder.fit_transform(data['variety'])
data['country_encoded'] = label_encoder.fit_transform(data['country'])
data['winery_encoded'] = label_encoder.fit_transform(data['winery'])
data['region1_encoded'] = label_encoder.fit_transform(data['region_1'])
data['region2_encoded'] = label_encoder.fit_transform(data['region_2'])
data['province_encoded'] = label_encoder.fit_transform(data['province'])
data['taster_name_encoded'] = label_encoder.fit_transform(data['taster_name'])
data['designation_encoded'] = label_encoder.fit_transform(data['designation'])
data['taster_twitter_handle_encoded'] = label_encoder.fit_transform(data['taster_twitter_handle'])
data['title_encoded'] = label_encoder.fit_transform(data['title'])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 24 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   country                        129971 non-null  object
 1   description                    129971 non-null  object
 2   designation                    129971 non-null  object
 3   points                         129971 non-null  int64
 4   price                          129971 non-null  float64
 5   province                       129971 non-null  object
 6   region_1                       129971 non-null  object
 7   region_2                       129971 non-null  object
 8   taster_name                    129971 non-null  object
 9   taster_twitter_handle          129971 non-null  object
 10  title                          129971 non-null  object
 11  variety                        129971 non-null  object
 12  winery                         129971 non-null  object
 13  Vintage(Year)                  129971 non-null  int64
 14  variety_encoded                129971 non-null  int64
 15  country_encoded                129971 non-null  int64
 16  winery_encoded                 129971 non-null  int64
 17  region1_encoded                129971 non-null  int64
 18  region2_encoded                129971 non-null  int64
 19  province_encoded               129971 non-null  int64
 20  taster_name_encoded            129971 non-null  int64
 21  designation_encoded            129971 non-null  int64
 22  taster_twitter_handle_encoded  129971 non-null  int64
 23  title_encoded                  129971 non-null  int64
dtypes: float64(1), int64(12), object(11)
memory usage: 23.8+ MB
```

Figure 15

The p-values for each variable were obtained and variables and the variables listed below were used to train this model.

```
y = data['price']
X = data[['points', 'Vintage(Year)','variety_encoded','country_encoded','winery_encoded','region1_encoded','region2_enc

#Adding the intercept
X = sm.add_constant(X)

#Fitting the linear regression model
model = sm.OLS(y, X).fit()

#Getting the summary of the regression model
summary = model.summary()

#Printing the summary, to display p-values for each variable
print(summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.168
Model:                            OLS   Adj. R-squared:                  0.168
Method:                 Least Squares   F-statistic:                     2182.
Date:                Sat, 09 Sep 2023   Prob (F-statistic):               0.00
Time:                        00:30:52   Log-Likelihood:            -6.5084e+05
No. Observations:              129971   AIC:                         1.302e+06
Df Residuals:                  129958   BIC:                         1.302e+06
Df Model:                          12
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                         -417.2896      4.812    -86.721      0.000    -426.721    -407.858
points                           5.1999      0.034    153.754      0.000       5.134       5.266
Vintage(Year)                   -0.0054      0.002     -2.932      0.003      -0.009      -0.002
variety_encoded                 -0.0081      0.001    -15.292      0.000      -0.009      -0.007
country_encoded                  0.0237      0.009      2.646      0.008       0.006       0.041
winery_encoded                   0.0005      0.000      1.466      0.143      -0.000       0.001
region1_encoded                 -0.0022      0.000     -7.388      0.000      -0.003      -0.002
region2_encoded                  0.1531      0.029      5.251      0.000       0.096       0.210
province_encoded                -0.0056      0.001     -7.473      0.000      -0.007      -0.004
taster_name_encoded              0.5552      0.031     17.837      0.000       0.494       0.616
designation_encoded          -4.308e-05   1.01e-05     -4.269      0.000   -6.29e-05   -2.33e-05
taster_twitter_handle_encoded    0.0468      0.038      1.217      0.224      -0.029       0.122
title_encoded                -6.051e-05   4.56e-05     -1.328      0.184      -0.000    2.88e-05
==============================================================================
Omnibus:                   300223.945   Durbin-Watson:                   1.719
Prob(Omnibus):                  0.000   Jarque-Bera (JB):       7842071673.406
Skew:                          22.170   Prob(JB):                         0.00
Kurtosis:                    1205.549   Cond. No.                     3.45e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.45e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Figure 16

The first model included the following features based on the results of the p-values.

Model with only the p-value variables ¶

```
#Regression Random Forest Regressor Model:
#Selecting the features required for the model (you can experiment with additional features)
selected_features = ['country_encoded','winery_encoded','Vintage(Year)','taster_twitter_handle_encoded','title_encoded'

#Extracting the selected features and the target variable
X = data[selected_features]
y = data['price']


#Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=22)

#Building and training the model using Random Forest Regressor model
rf_model = RandomForestRegressor(n_estimators=100, random_state=22)
rf_model.fit(X_train, y_train)

#Making the predictions using the trained model
y_pred = rf_model.predict(X_test)

#Evaluating the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

#Printing evaluation metrics
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2): {r2:.2f}")

#Visualising the predicted vs. actual prices
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs. Predicted Prices")
plt.show()
```

```
Mean Absolute Error (MAE): 13.88
Mean Squared Error (MSE): 1261.36
R-squared (R2): 0.19
```

Figure 17

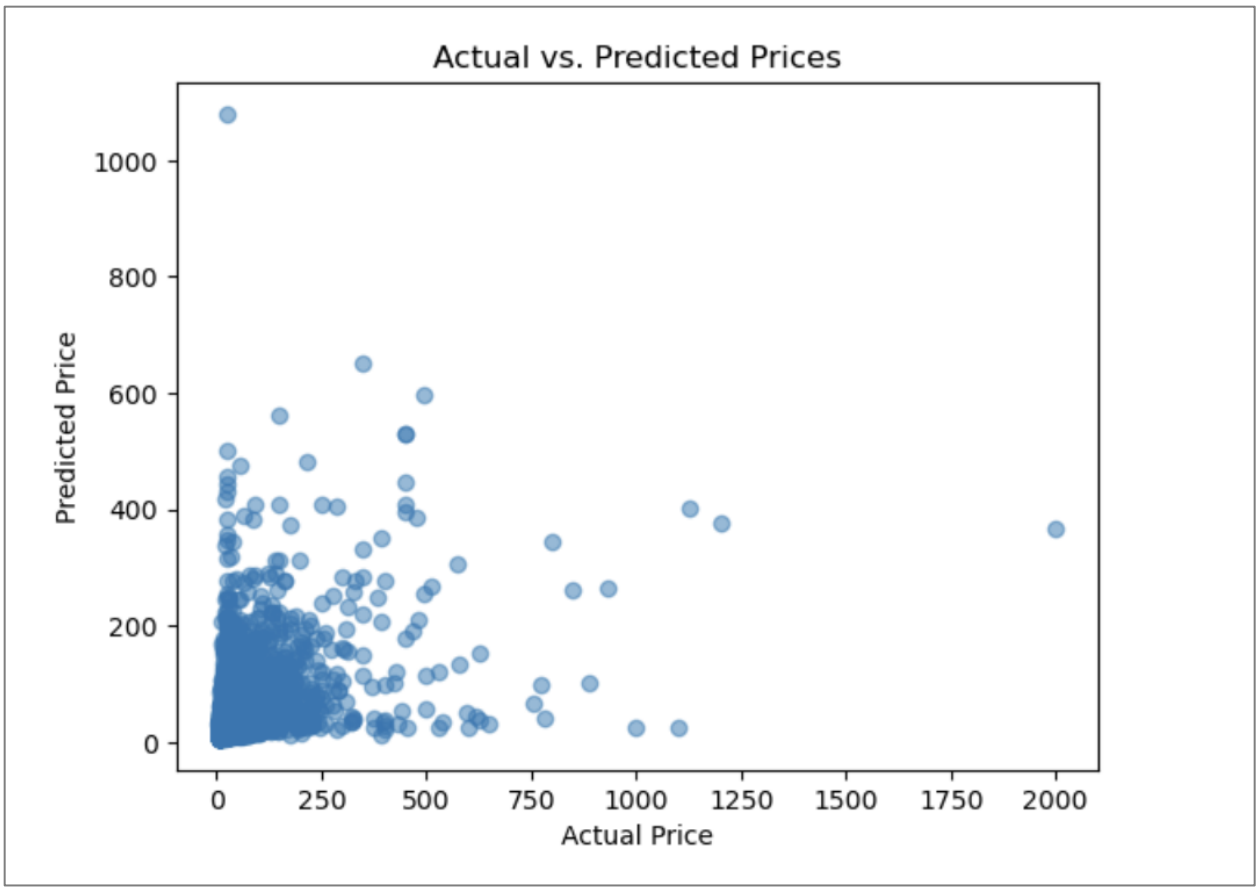

Figure 18

The second model included variables based on some of the p-values and wine knowledge and review information.

Model 2 using varibles based on P-values and wine knowledge

```
#Regression Random Forest Regressor Model:
#Selecting the features required for the model (you can experiment with additional features)
selected_features = ['points','winery_encoded','Vintage(Year)','variety_encoded','country_encoded']

#Extracting the selected features and the target variable
X = data[selected_features]
y = data['price']

#Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=22)

#Building and training the model using Random Forest Regressor model
rf_model = RandomForestRegressor(n_estimators=100, random_state=22)
rf_model.fit(X_train, y_train)

#Making the predictions using the trained model
y_pred = rf_model.predict(X_test)

#Evaluating the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

#Printing the evaluation metrics
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2): {r2:.2f}")

#Visualising the predicted vs. actual prices
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs. Predicted Prices")
plt.show()
```

```
Mean Absolute Error (MAE): 12.48
Mean Squared Error (MSE): 1084.76
R-squared (R2): 0.30
```

Figure 19

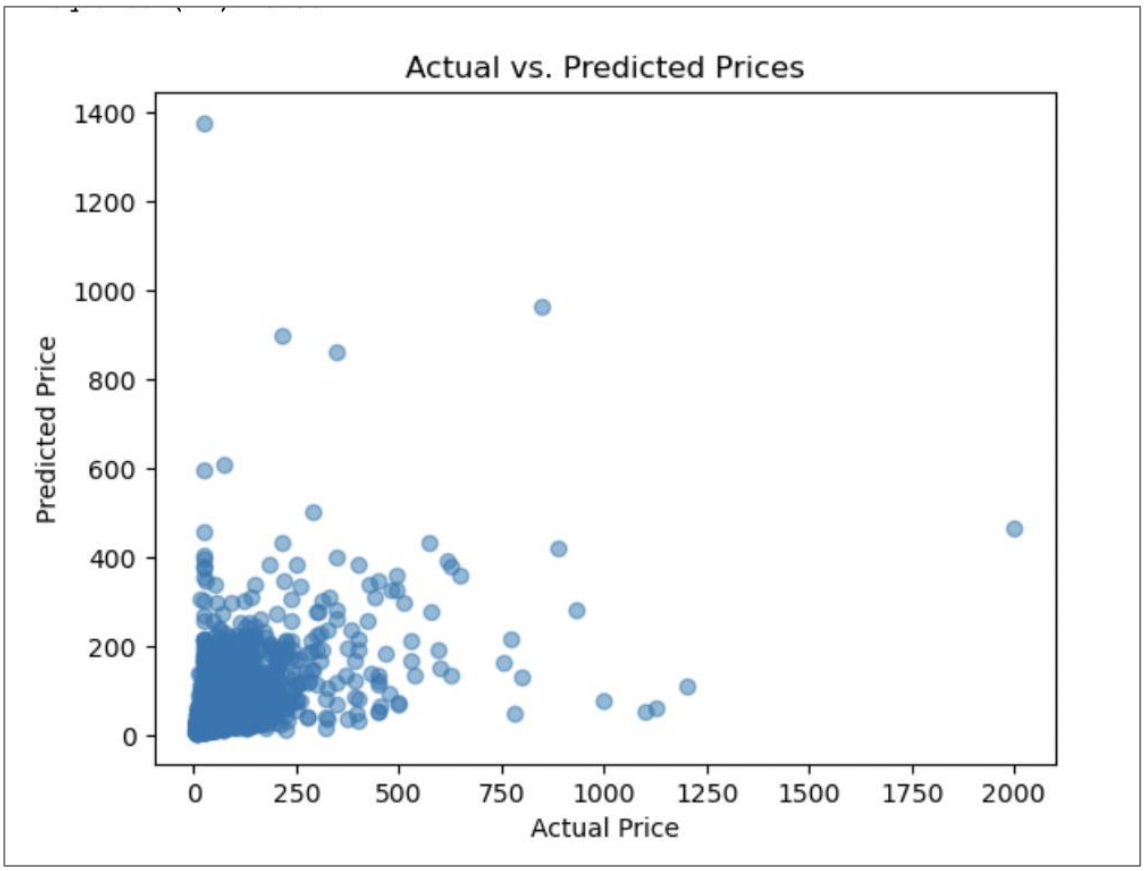

Figure 20

A scatter plot was plotted to understand the relationship between price and points.

```
#Plotting the scatter plot for price vs points
import matplotlib.pyplot as plt

plt.scatter(data['points'], data['price'], alpha=0.5)
plt.xlabel('Points')
plt.ylabel('Price')
plt.title('Scatter Plot: Points vs. Price')
plt.show()
```

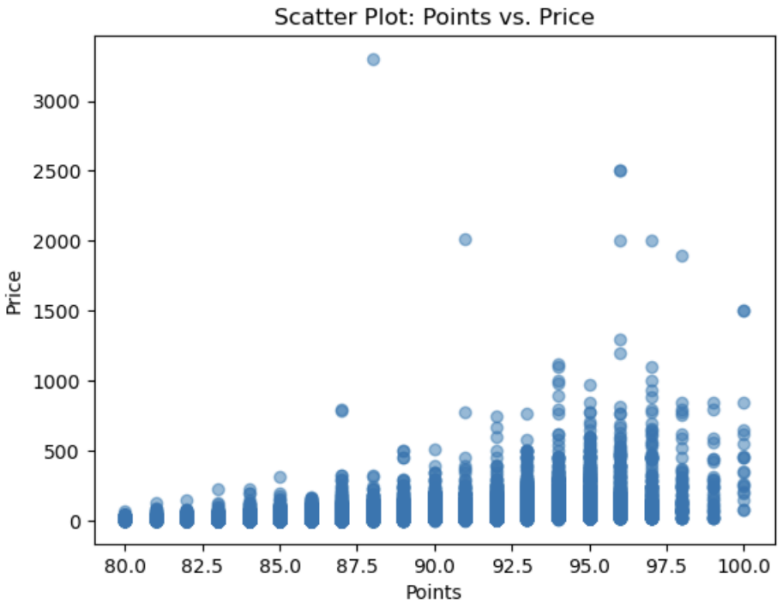

Figure 21

Regression Algorithm Used

The Random Forest Regression model was used for both models and was selected over Linear Regression model since most variables did not show strong linear relationships and it is less sensitive to outliers and proved to be the better model.

Developing the Classification Model for Sub Problem 2

The model included the following features to classify value picks and premium picks. Wines below $200 were considered as value picks and wines above $200 were considered as premium picks.

Sub-question 2: Classification Model

```
#Subquestion 2:
#Defining the criteria for labeling wines as "premium choices" (1) and "value picks" (0)
#We have considered wines with a price above $200 as "premium choices" and those below as "value picks"
data['target'] = (data['price'] > 200).astype(int)

#Selecting the features to build the model
features = ['points', 'price', 'variety']

#Creating a pipeline for preprocessing and modeling
preprocessor = ColumnTransformer(
    transformers=[
        ('num', SimpleImputer(strategy='mean'), ['points', 'price']),
        ('cat', OneHotEncoder(handle_unknown='ignore'), ['variety'])
    ])
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])

X = data[features]
y = data['target']

#Splitting the dataset into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=22)

#Building and training the classification model
pipeline.fit(X_train, y_train)

#Making predictions on the testing set
y_pred = pipeline.predict(X_test)

#Evaluating the built model
accuracy = accuracy_score(y_test, y_pred)
classification_report_text = classification_report(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report_text)
print("Confusion Matrix:\n", confusion)
```

```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     25860
           1       1.00      0.99      1.00       135

    accuracy                           1.00     25995
   macro avg       1.00      1.00      1.00     25995
weighted avg       1.00      1.00      1.00     25995

Confusion Matrix:
 [[25860     0]
 [    1   134]]
```

Figure 22

Predictions were then made based on the developed model and the data was further explored by using various visualisation techniques.

Making Predictions

```
#Defining some sample wine data to make predictions
wine_samples = [
    {'points': 90, 'price': 30, 'variety': 'Chardonnay'},
    {'points': 88, 'price': 60, 'variety': 'Pinot Noir'},
    {'points': 85, 'price': 20, 'variety': 'Merlot'},
]

#Creating a DataFrame for the sample data
sample_data = pd.DataFrame(wine_samples)
# Using the trained model to make predictions for the sample data
predicted_labels = pipeline.predict(sample_data)

for i, (predicted_label, row) in enumerate(zip(predicted_labels, sample_data.iterrows())):
    wine_type = "premium choice" if predicted_label == 1 else "value pick"
    points = row[1]['points']
    variety = row[1]['variety']
    price = row[1]['price']
    print(f"Wine {i + 1}: {variety} ({points} points) with a price of ${price:.2f} is predicted to be a {wine_type}.")
```

```
Wine 1: Chardonnay (90 points) with a price of $30.00 is predicted to be a value pick.
Wine 2: Pinot Noir (88 points) with a price of $60.00 is predicted to be a value pick.
Wine 3: Merlot (85 points) with a price of $20.00 is predicted to be a value pick.
```

Figure 23

All value pick and premium wine picks were printed for each variety.

Displaying all the value pick and premium wines for each variety

In [174]:
```
#Displaying all the value pick and premium wines for each variety:
#Getting the unique wine varieties from the dataset
unique_varieties = data['variety'].unique()

#Defining the categorization criteria for different varieties
criteria = {}

for variety in unique_varieties:
    #Setting the criteria for all varieties
    criteria[variety] = {'Premium': (201, 4000), 'Value': (0, 200)}

#Creating an empty DataFrames to store results
value_picks = pd.DataFrame(columns=data.columns)
premium_wines = pd.DataFrame(columns=data.columns)

#Looping through each unique variety and categorize wines
for variety, variety_data in data.groupby('variety'):
    if variety in criteria:
        premium_criteria = criteria[variety]['Premium']
        value_criteria = criteria[variety]['Value']
        premium_mask = (variety_data['price'] > premium_criteria[1])
        value_mask = (variety_data['price'] >= value_criteria[0]) & (variety_data['price'] <= value_criteria[1])

        #Filtering the wines into value picks and premium wines
        value_picks = pd.concat([value_picks, variety_data[value_mask]])
        premium_wines = pd.concat([premium_wines, variety_data[premium_mask]])

#Displaying the main variety, value pick wines, premium wines, and Vintage year
for variety in unique_varieties:
    if variety in criteria:
        print(f"\nVariety: {variety}")
        print("Value Picks:")
        print(value_picks[value_picks['variety'] == variety][['variety', 'price', 'Vintage(Year)']])
        print("\nPremium Wines:")
        print(premium_wines[premium_wines['variety'] == variety][['variety', 'price', 'Vintage(Year)']])
```

```
Variety: White Blend
Value Picks:
             variety  price Vintage(Year)
0       White Blend   25.0          2013
22      White Blend   19.0          2007
26      White Blend   13.0          2013
32      White Blend   25.0          2011
105     White Blend   14.0          2015
...             ...    ...           ...
129841  White Blend   10.0          2008
129884  White Blend   43.0          2009
129899  White Blend   30.0          2015
129933  White Blend   46.0          2005
129939  White Blend   70.0          2004

[2359 rows x 3 columns]

Premium Wines:
```

Figure 24

To enhance the model and improve convenience for the winery and customers, the model was enhanced to obtain user input and provide results based on this. The user can input the preferred wine variety and obtain value picks and premium pick wine information.

Printing the premium and value pick prices for each wine variety with user input

```
#Getting the unique wine varieties from the dataset
unique_varieties = data['variety'].unique()

#Defining the categorization criteria for different varieties
criteria = {}

for variety in unique_varieties:
    # Set criteria for all varieties
    criteria[variety] = {'Value': (0, 200), 'Premium': (201, 4000)}

#Prompting the user to enter the wine variety they want to check the prices
user_variety = input("Enter the wine variety: ")

#Checking if the user-specified variety exists in the dataset
if user_variety in unique_varieties:
    print(f"\nVariety: {user_variety}")

    #Filtering the wines based on user-specified variety
    user_variety_data = data[data['variety'] == user_variety]

    #Applying the adjusted categorization criteria to the user-specified variety
    premium_criteria = criteria[user_variety]['Premium']
    value_criteria = criteria[user_variety]['Value']
    premium_mask = (user_variety_data['price'] >= premium_criteria[0]) & (user_variety_data['price'] <= premium_criteri
    value_mask = (user_variety_data['price'] >= value_criteria[0]) & (user_variety_data['price'] <= value_criteria[1])

    #Displaying premium and value pick wines for the specified variety
    print("\nPremium Wines:")
    print(user_variety_data[premium_mask][['price', 'Vintage(Year)']])

    print("\nValue Picks:")
    print(user_variety_data[value_mask][['price', 'Vintage(Year)']])
else:
    print(f"The wine variety '{user_variety}' is not found in the dataset.")
```

Figure 25

```
Enter the wine variety: Pinot Noir

Variety: Pinot Noir

Premium Wines:
        price  Vintage(Year)
357     350.0           2013
3069    205.0           2008
6730    285.0           2007
6731    280.0           2008
8891    250.0           2009
...       ...            ...
120440  279.0           2014
120594  250.0           2008
120595  236.0           2009
121944  360.0           2015
127566  350.0           2008

[123 rows x 2 columns]

Value Picks:
        price  Vintage(Year)
4        65.0           2012
21       20.0           2013
25       69.0           2011
35       50.0           2010
41       22.0           2009
...       ...            ...
129920   48.0           2006
129931  107.0           2005
129936   66.0           2005
129960   48.0           2006
129967   75.0           2004

[13150 rows x 2 columns]
```

Figure 26

Classification Algorithm Used

A Logistic Regression model was used to build this model since it involved a binary classification(value picks and premium picks) and is computationally efficient in handling large datasets.

Predictions

After the model was trained, a few predictions were made as shown below and the data was further explored by using various visualisation techniques.

Making Predictions

```
#Defining some sample wine data to make predictions
wine_samples = [
    {'points': 90, 'price': 30, 'variety': 'Chardonnay'},
    {'points': 88, 'price': 60, 'variety': 'Pinot Noir'},
    {'points': 85, 'price': 20, 'variety': 'Merlot'},
]

#Creating a DataFrame for the sample data
sample_data = pd.DataFrame(wine_samples)
# Using the trained model to make predictions for the sample data
predicted_labels = pipeline.predict(sample_data)

for i, (predicted_label, row) in enumerate(zip(predicted_labels, sample_data.iterrows())):
    wine_type = "premium choice" if predicted_label == 1 else "value pick"
    points = row[1]['points']
    variety = row[1]['variety']
    price = row[1]['price']
    print(f"Wine {i + 1}: {variety} ({points} points) with a price of ${price:.2f} is predicted to be a {wine_type}.")
```

```
Wine 1: Chardonnay (90 points) with a price of $30.00 is predicted to be a value pick.
Wine 2: Pinot Noir (88 points) with a price of $60.00 is predicted to be a value pick.
Wine 3: Merlot (85 points) with a price of $20.00 is predicted to be a value pick.
```

Figure 27

Findings and Discussions

Exploring the Dataset and Visualising Insights

The percentage of different varieties of wines

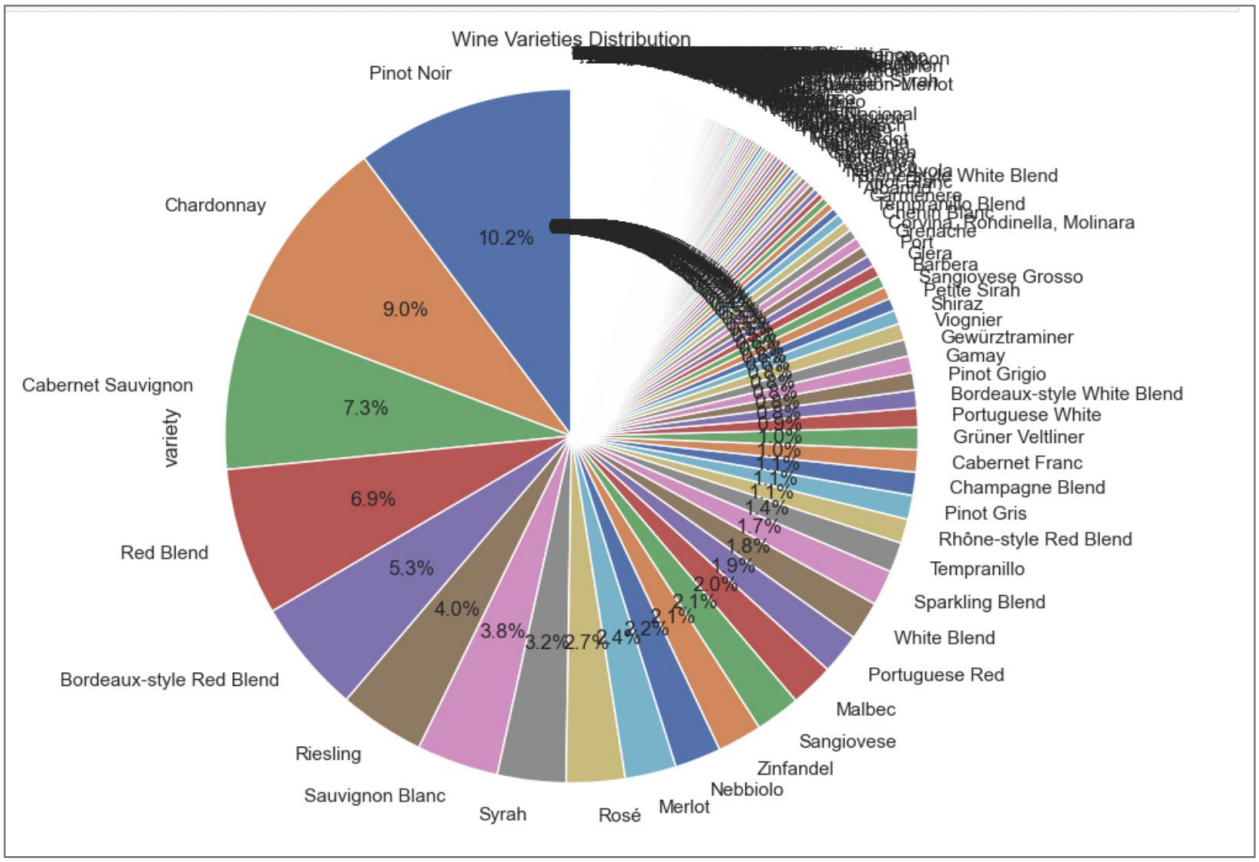

Figure 28

The pie chart explains the distribution of the different wine varieties. Our focus here is to see which wines are currently in stock and are popular. From the pie chart we can clearly see that Pinot Noir and Chardonnay are the clear favourites, making up a combined 19.2% of the total wine sales. These are followed by Cabernet Sauvignon at 7.3% and Red Blend at 6.9%. These four wines combined make up a third of the total wines sold and hence the company should focus on producing and selling these wines as they are the biggest drivers of sales.

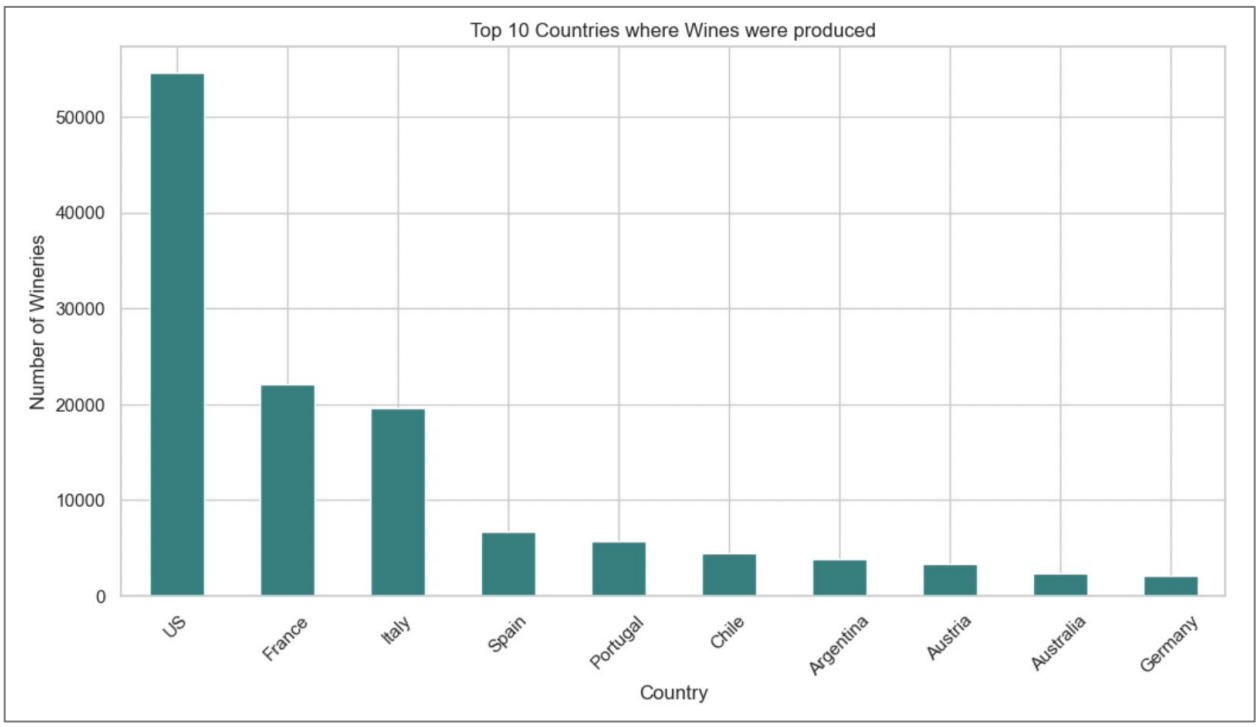

Figure 29

This graph shows the top 10 countries where wines are produced. The US leads the way from the rest with over 50000 wineries. It has more than twice the number of wineries than France and Italy combined and hence the company should focus on putting in more resources and funds towards the wineries in the US to produce more wine.

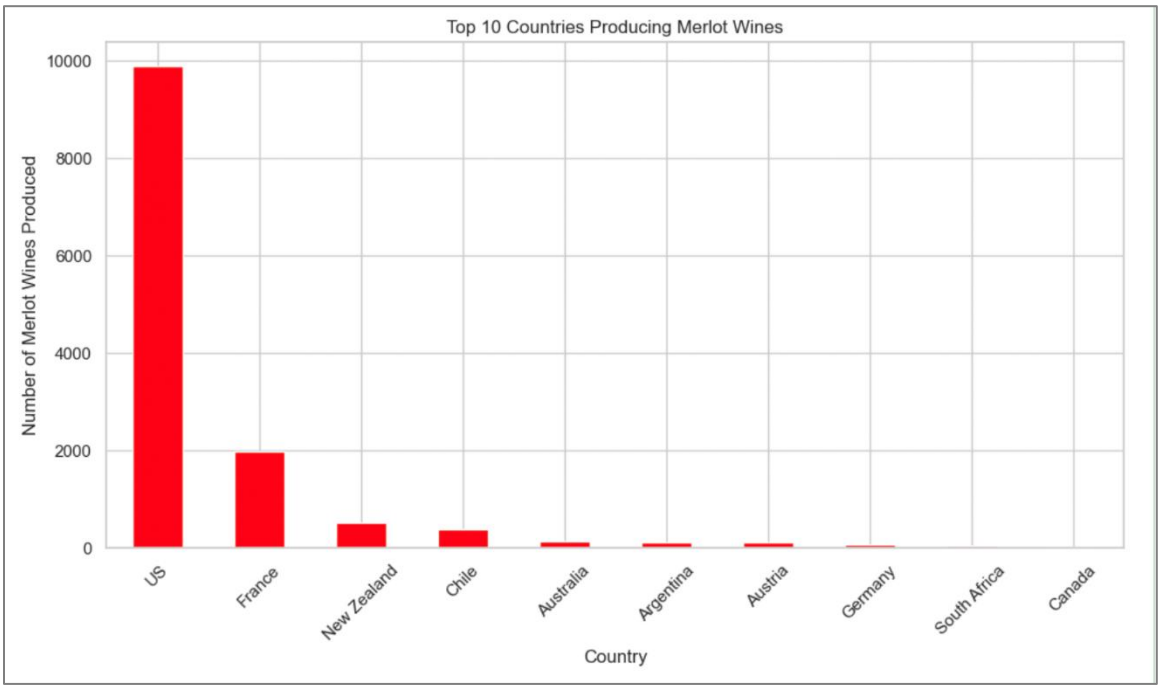

Figure 280

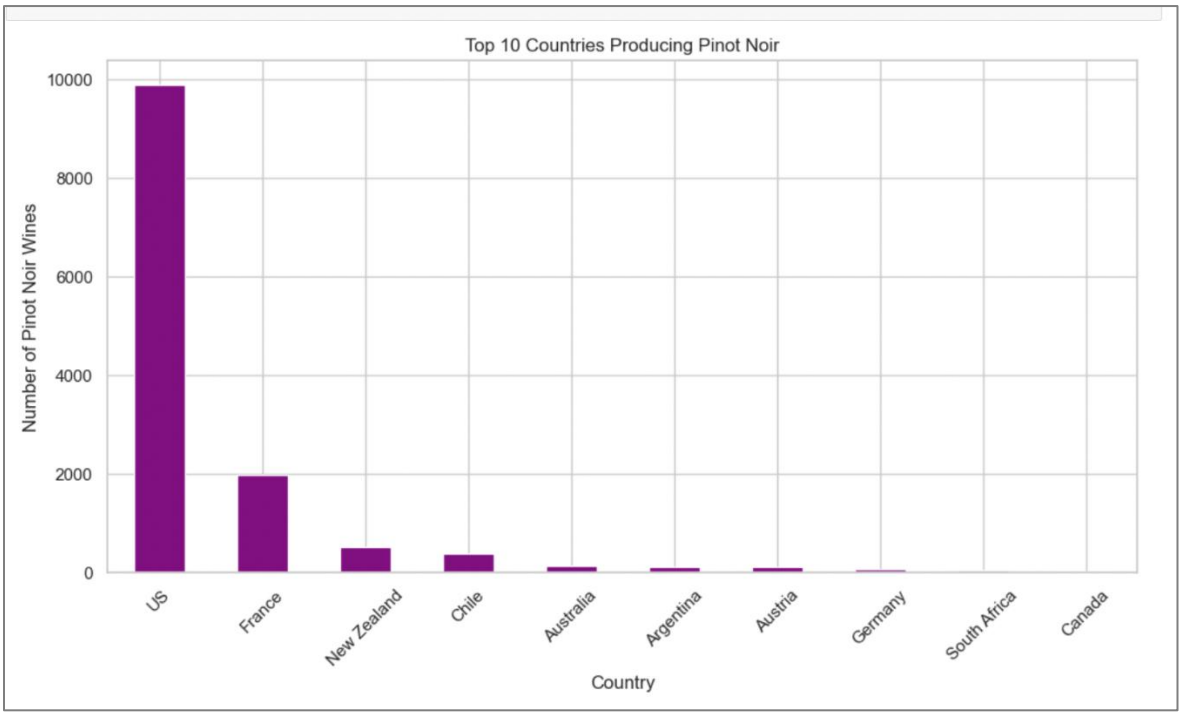

Figure 31

The above two graphs reiterate the fact that US is the leading producer of both Merlot and Pinot Noir wines followed by France and New Zealand.

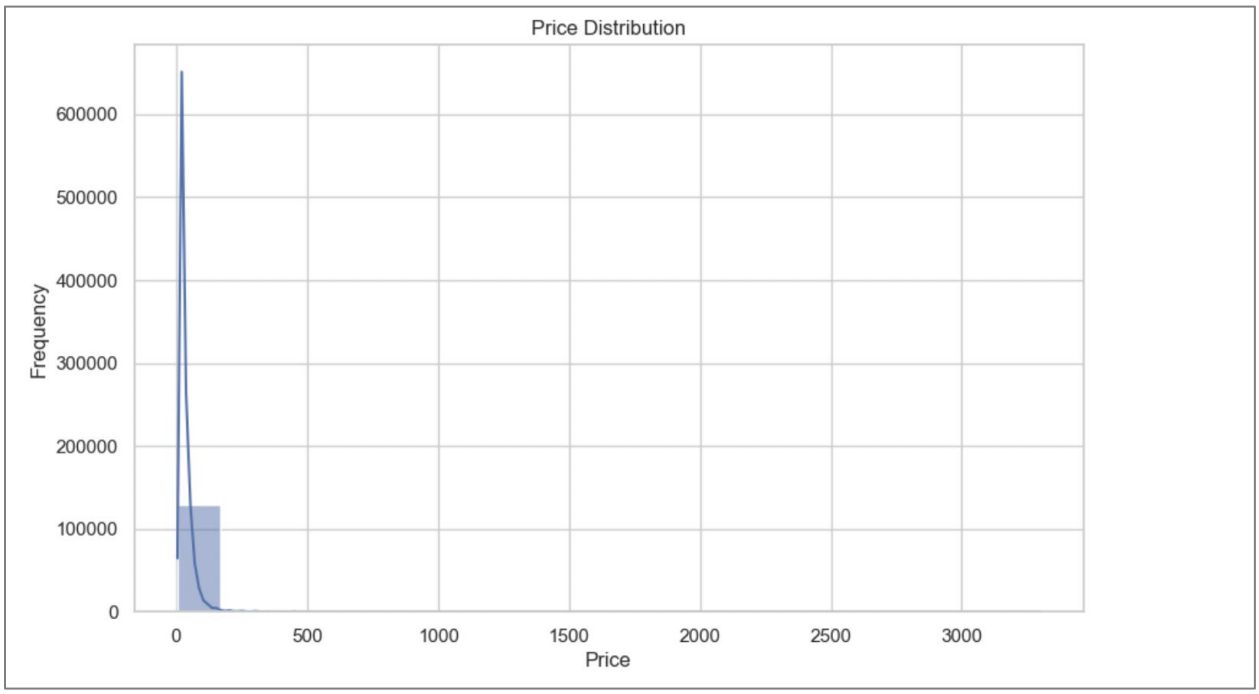

Figure 32

The price distribution graph shows that most wines fall around the price range of 0 to $100. There are more than 600,000 wines that fall under this price range, and this could be since most customers prefer wines in the affordable range for the average consumer. A graph like this is essential while taking pricing decisions as it provides a clear idea about the market forces.

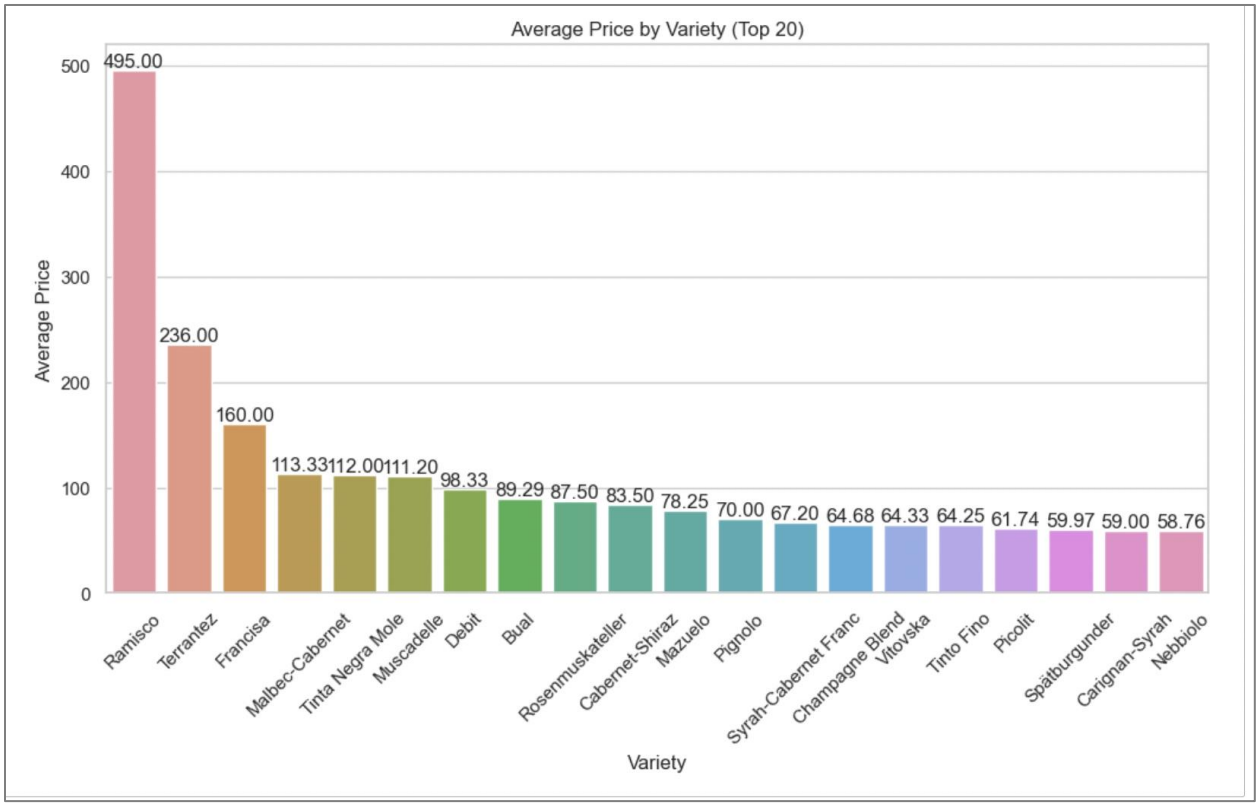

Figure 33

This graph shows the average price of the top 20 wine varieties(descending order). Ramisco wines are usually close to $495, while wines like Nebbiolo and Carignan-Syrah are close to $60. This should give the company an idea of what kind of wines to stock and produce more and which to stock and produce less as the consumer demand for wines will drop as the prices rise.

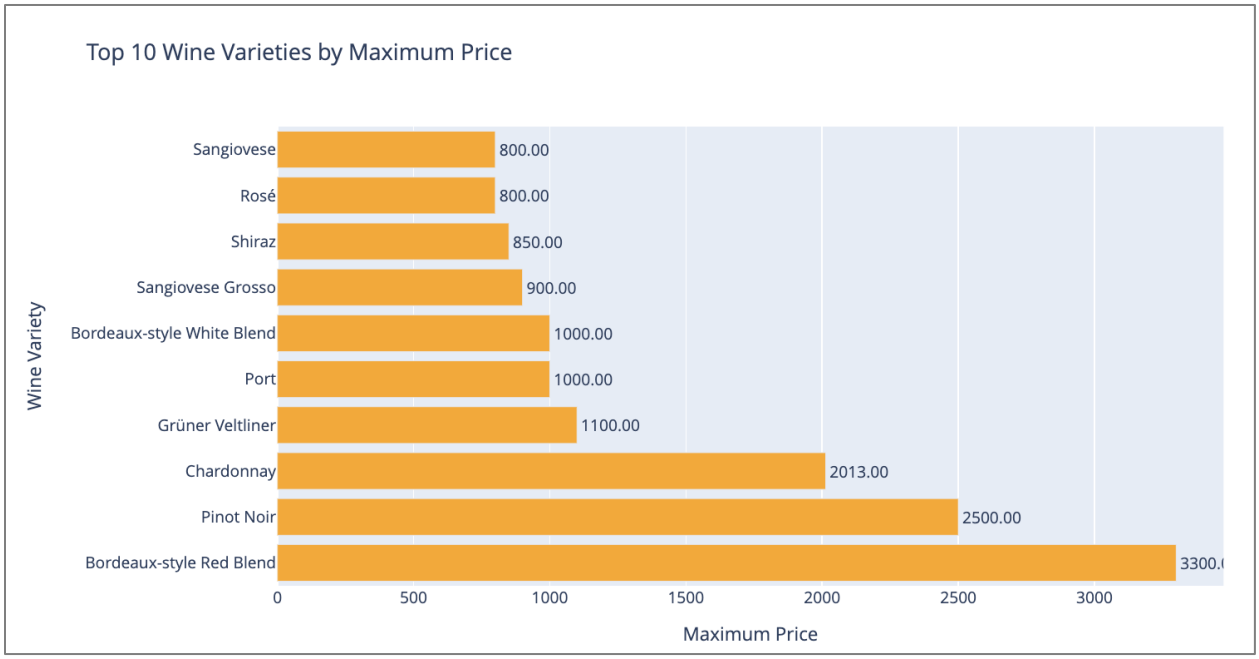

Figure 34

Figure 3529

The above graphs show the top 10 varieties of wine based on maximum and minimum price. While the most expensive wines range from $800 to the Bourdeaux-style Red Blend at $3300, the cheaper options cost around $4.

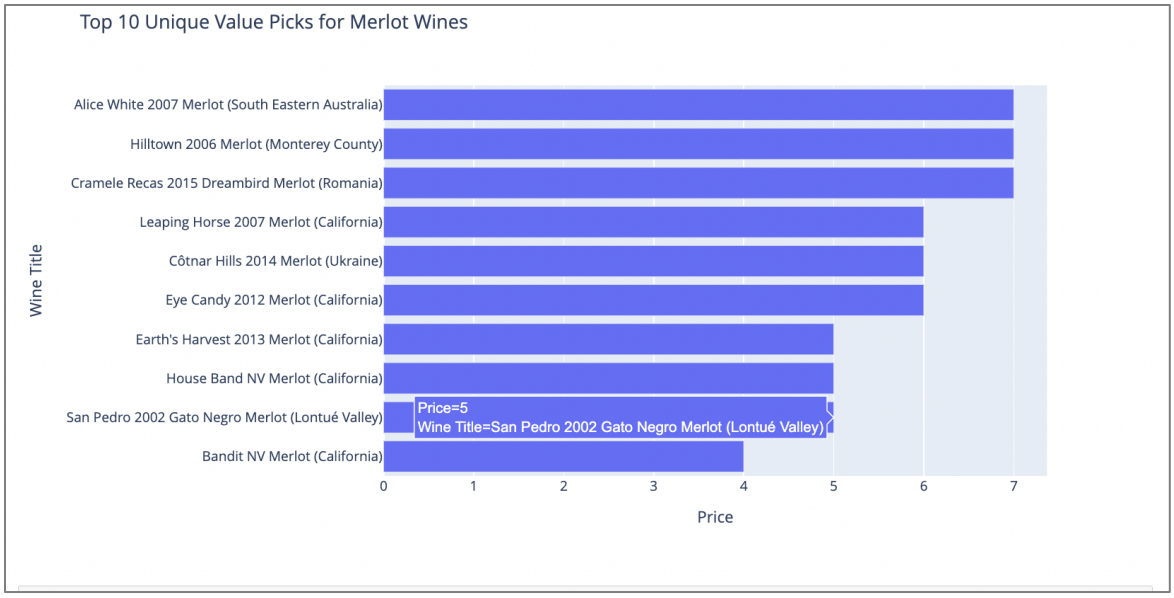

Figure 36

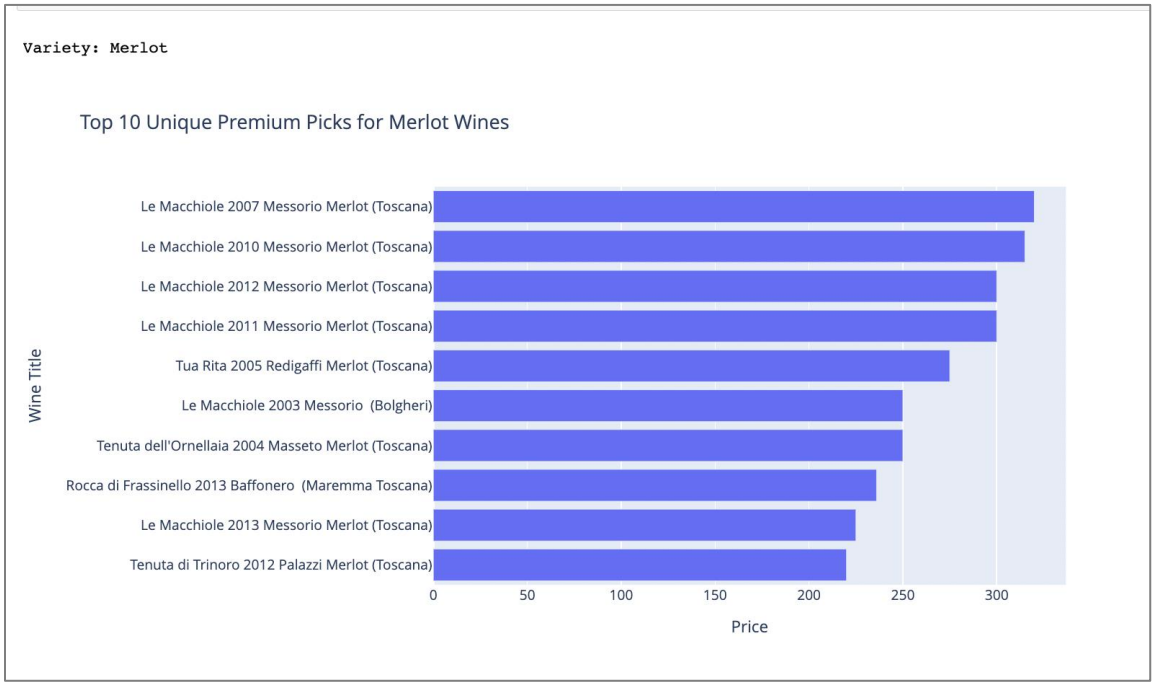

Figure 37

The above two graphs represent the top 10 premium and value picks for Merlot wines. While the value picks range from $4 to $7, the premium ones come in between $200 and $350.

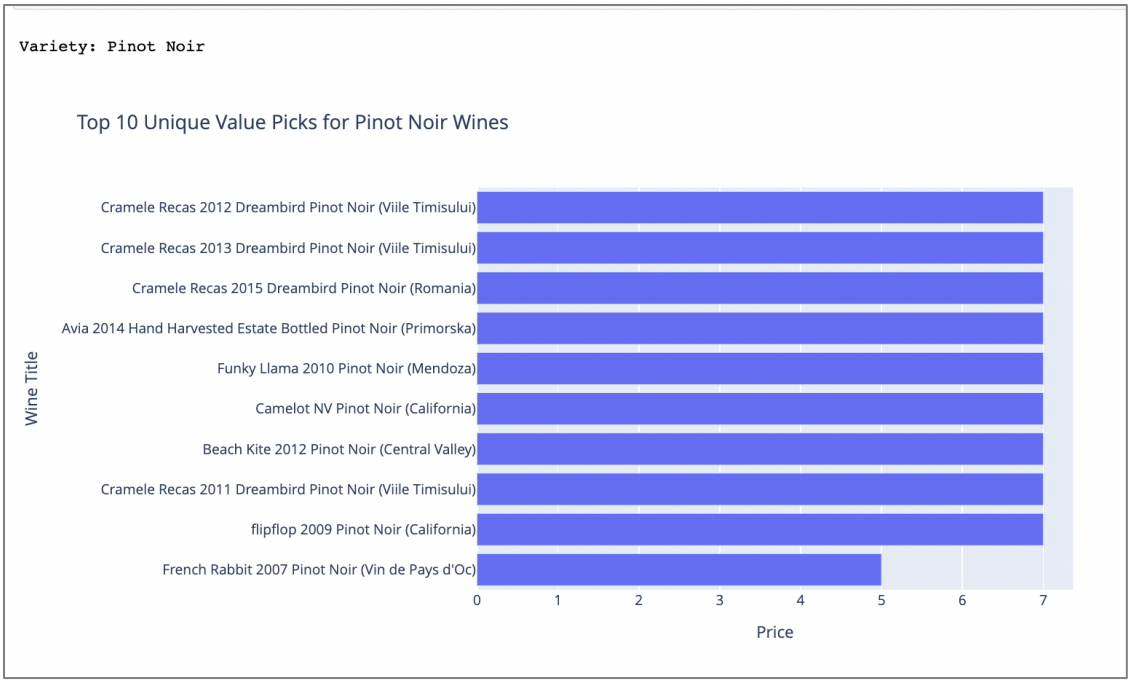

Figure 38

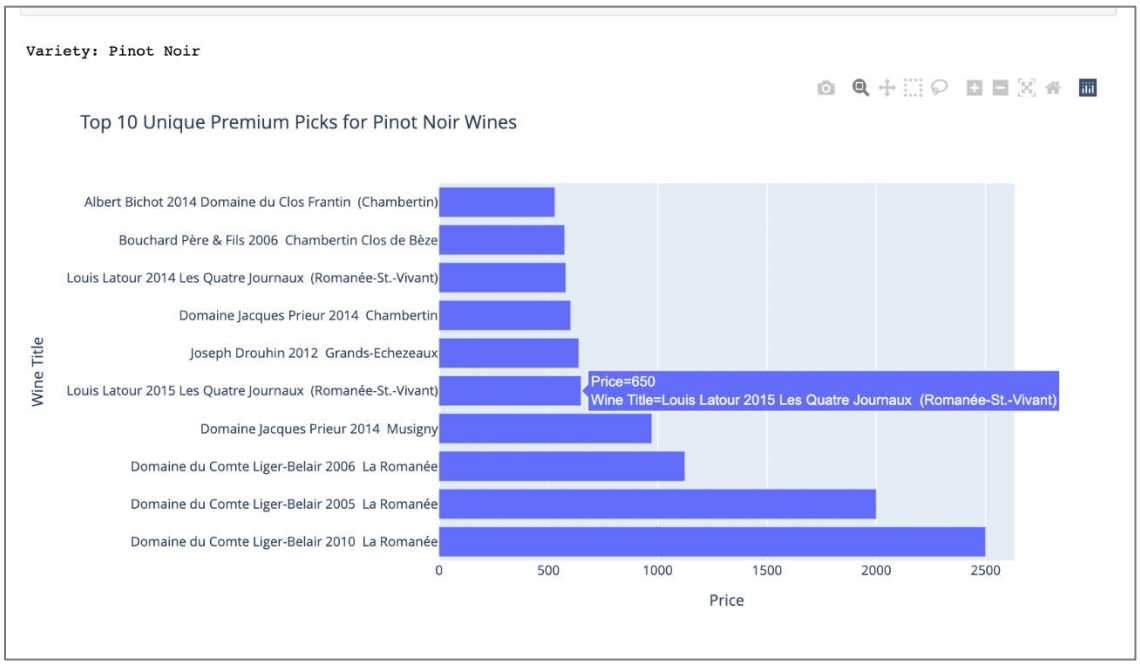

Figure 39

The above two graphs shows the top 10 premium and value picks for Pinot Noir wines. While the value picks are mostly priced at $7, the premium ones range from just over $500 upto $2500.

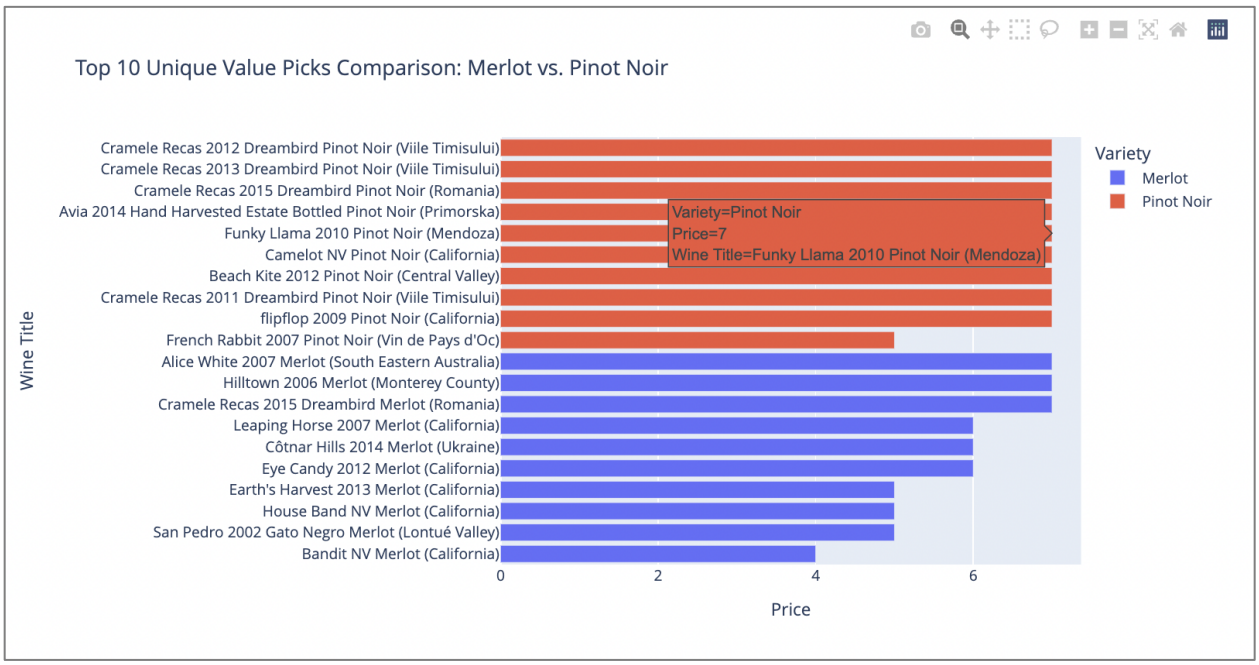

Figure 40

Merlot and Pinot Noir are one of the most popular red wines. The above graph shows a comparison of the top 10 unique value picks for both these wine varieties. While we can see that 9 of the top 10 Pinot Noir wines are priced at $7, Merlot has even cheaper options. As a company which needs to take the decision on which wine to produce more of, visualisations like this help to show that having a wider variety of Merlot wines could target a larger market demographic due to its wider range of price points.

Data Visualization

The following data visualizations are regarding the Linear Regression model.

Model –1:

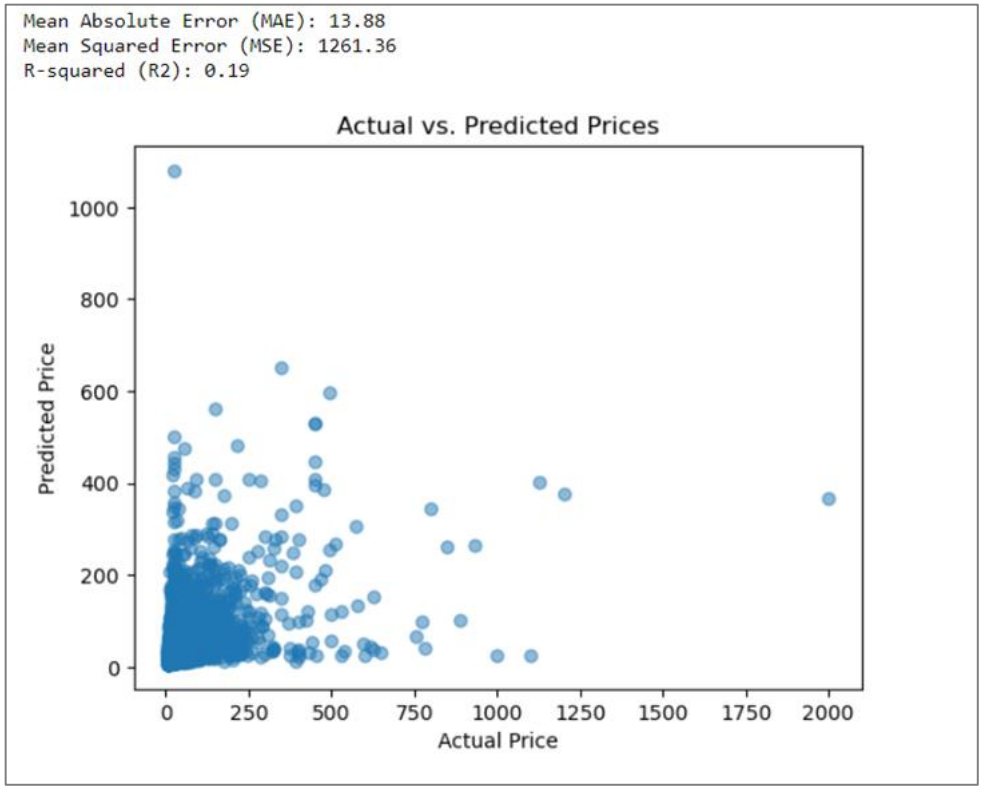

Figure 41

Scatter plots are a good way of comparing actual and predicted prices. The above graph for Model 1 shows a positive correlation between the two prices. However, as we can see, the correlation is very low. It's difficult to draw a line of best fit through the points. Ideally, a model that is highly correlated would produce a scatter plot wherein all the points would be closely placed next to each other such that a straight line could be drawn through them explaining the overall trend. That's not the case here. This shows that the variables we have taken do not predict the price of the wines accurately.

Model – 2:

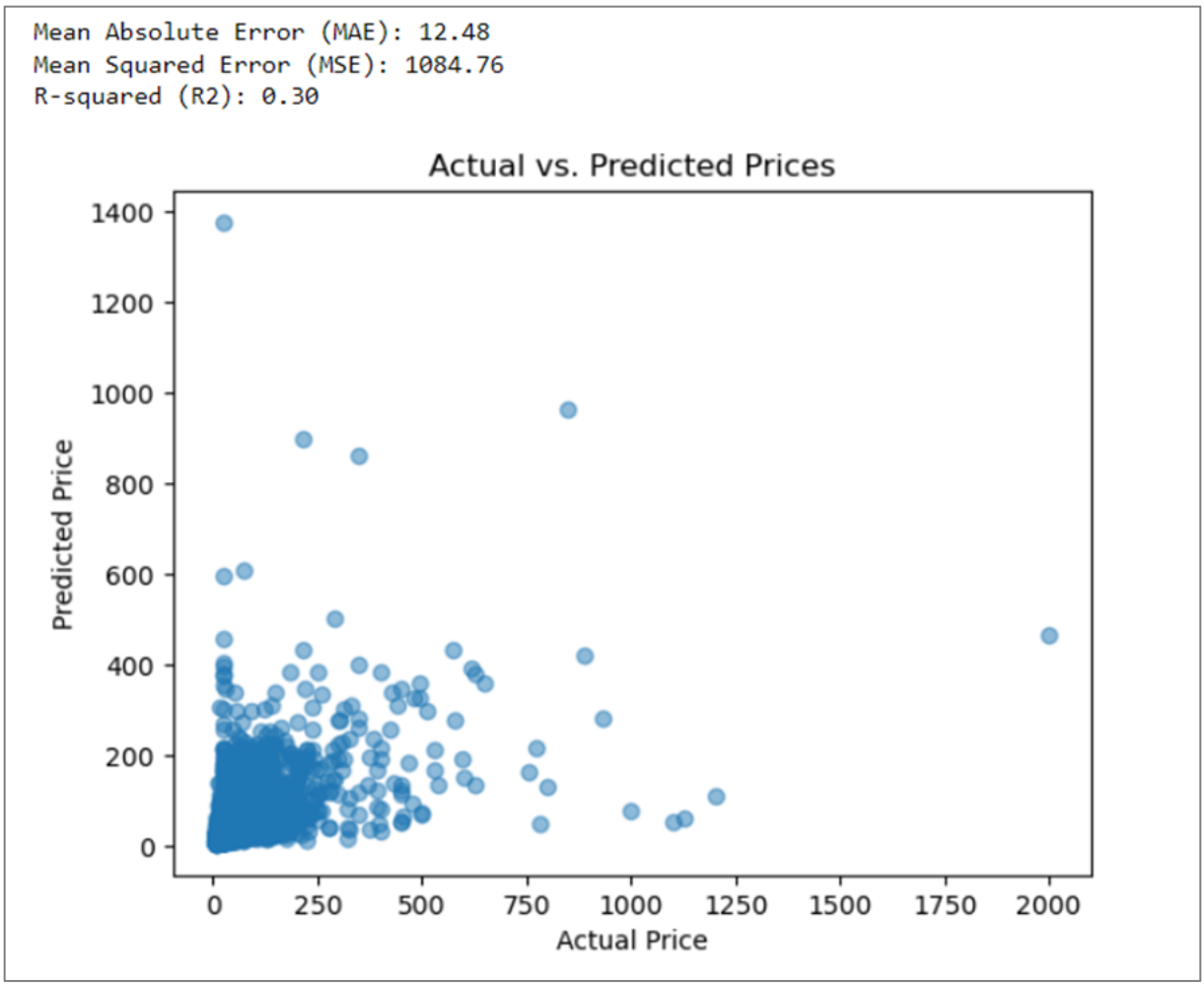

Figure 42

The above scatter plot for Model 2 is quite similar to that of Model 1. We can again see that there is a positive correlation between the two prices. However, the correlation is very low and it's difficult to draw a line of best fit through the points. Even though visually there is not much difference between the two models, Model 2 actually does a slightly better job at predicting the prices, as explained previously in the report.

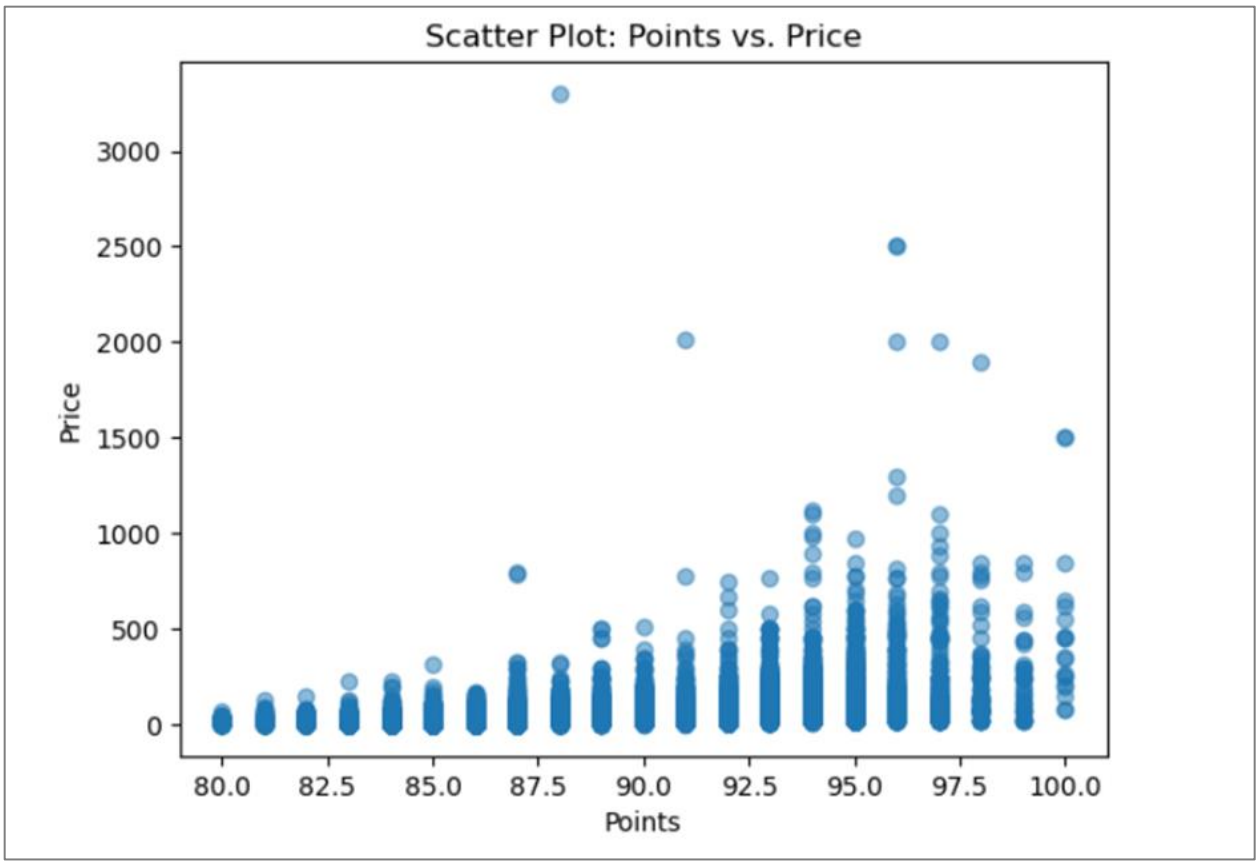

Figure 43

This scatter plot graph between points and price, gives us a good representation of how as the points of the wines increase, their prices increase as well. We can clearly see the general trend of the graph rising upward. This suggests that there is a positive correlation between the price of the wine and its points. The graph also indicates certain outliers where the price is significantly higher than the trend.

Model Evaluation

Regression Model

The first sub problem was handled using a random forest regression model to predict prices based on the different variables available in the dataset. To find out which variables affected the price of the wine significantly, the p-values were analysed. All textual variables were encoded into numerical values. Once we did that, we calculated the OLS regression summary, which gave us

the p-values. From our results, we could clearly notice one thing: all variables had a P-value of zero, except the variables 'country_encoded','winery_encoded','Vintage(Year)','taster_twitter_handle_encoded','title_encoded', which had P-values of 0.008, 0.143, 0.003, 0.224 and 0.184 respectively. The p-value represents the probability of observing a test statistic as extreme as, or more extreme than, the one calculated from the sample data, assuming that the null hypothesis is true. And since, only 5 variables had a P-value > 0, those were the only 5 variables we could work with. Furthermore, we realized that variables like 'taster_twitter_handle_encoded' and 'title_encoded' wouldn't affect the price of a wine to a great extent, we still wanted to include these variables as we wanted to make our model as precise as possible using the data we had.

With the above results, we realized that none of our variables were actually highly correlated to the price of the wine. Hence, we decided to create 2 regression models, one with variables that had a P-value other than zero and the other which included other variables as well that we found through our research could affect the price of a wine. We wanted to compare the two models and see which was more reliable in predicting the price of the wine. Our first model, which included the variables: 'country_encoded', 'winery_encoded', 'Vintage(Year)', 'taster_twitter_handle_encoded', 'title_encoded'.

The result was as follows:

```
Mean Absolute Error (MAE): 13.88
Mean Squared Error (MSE): 1261.36
R-squared (R2): 0.19
```

Figure 44

On the other hand, our second model, included the variables: 'points', 'winery_encoded', 'Vintage(Year)', 'variety_encoded', 'country_encoded'.

The result was as follows:

```
Mean Absolute Error (MAE): 12.48
Mean Squared Error (MSE): 1084.76
R-squared (R2): 0.30
```

Figure 45

Mean Absolute Error (MAE):

Model 2 has a lower MAE (12.48) compared to Model 1 (13.88). A lower MAE indicates that Model 2's predictions are, on average, closer to the actual values compared to Model 1. Therefore, Model 2 is performing better in terms of MAE.

Mean Squared Error (MSE):

Model 2 has a lower MSE (1084.76) compared to Model 1 (1261.36). A lower MSE indicates that Model 2's predictions have smaller errors, and the errors are squared, which makes Model 2's predictions more accurate than those of Model 1 in terms of MSE.

R-squared (R2):

Model 2 has a higher R-squared (0.30) compared to Model 1 (0.19). R-squared measures the proportion of the variance in the dependent variable that is explained by the independent variables. A higher R-squared indicates that Model 2 explains a larger proportion of the variance in the data compared to Model 1. Therefore, Model 2 provides a better fit to the data according to R-squared.

In summary, based on these evaluation metrics, Model 2 outperforms Model 1 in terms of MAE, MSE, and R-squared. It has lower errors, explains a larger proportion of the variance in the dependent variable, and provides better predictive performance.

The regression model predicts the price based on points, the winery, year, variety and country. Prices for Pinot Noir, Merlot and Chardonnay are displayed below.

```
#Defining the features for the predictions:
predictions = {
    'points': [84, 87, 87],
    'winery_encoded': [11608, 12956, 13018],
    'Vintage(Year)': [2012, 2012, 2012],
    'variety_encoded': [440, 125, 326],
    'country_encoded': [22, 31, 40],
}

#Creating a DataFrame from the new data
predictions_df = pd.DataFrame(predictions)

#Using the trained model to make predictions from our model:
predictions = rf_model.predict(predictions_df)

#Defining a dictionary to map variety_encoded values to variety names
variety_encoded_to_name = {440: 'Pinot Noir', 125: 'Chardonnay', 326: 'Merlot'}

#Printing the predicted prices
for i, predicted_price in enumerate(predictions):
    variety_encoded = predictions_df['variety_encoded'].iloc[i]
    variety_name = variety_encoded_to_name.get(variety_encoded, 'Unknown')
    print(f"Predicted Price for Sample {i+1} ({variety_name}): ${predicted_price:.2f}")
```

```
Predicted Price for Sample 1 (Pinot Noir): $21.55
Predicted Price for Sample 2 (Chardonnay): $20.64
Predicted Price for Sample 3 (Merlot): $21.59
```

Figure 46

Relation to the Main Problem and Sub Problem -Regression Model

The Random Forest Regression model predicts the price of wines, and it is essential for optimising inventory curation and improving customer satisfaction. The predicted prices can help wineries to understand their monthly cost and revenue and understand how they should stock/group their inventory wines based on pricing. By categorising wines based on predicted prices, the winery can stock wines based on their budgets and preferences and would enhance customer experience. The winery can also make informed decisions about reordering stock and promotional strategies.

Classification Model

The model was evaluated using the classification report and confusion matrix. An accuracy of 1.00 was obtained. Precision values, Recall, F1-score and Support indicate positive results for the model. Precision values indicated that all the positive predictions made were accurate and recall indicates that the model identified almost all positive instances and the F1-score indicates a good

balance between precision and recall values. The confusion matrix values show that the model only made one false positive prediction and no false negatives.

```
#Evaluating the built model
accuracy = accuracy_score(y_test, y_pred)
classification_report_text = classification_report(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report_text)
print("Confusion Matrix:\n", confusion)
```

```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     25860
           1       1.00      0.99      1.00       135

    accuracy                           1.00     25995
   macro avg       1.00      1.00      1.00     25995
weighted avg       1.00      1.00      1.00     25995

Confusion Matrix:
 [[25860     0]
 [    1   134]]
```

Figure 47

Output

The user is able to input the preferred wine variety and obtain the list of premium picks and value pick wines. This would be very practical and easy to use as this saves time and provides accurate information to enhance customer experience.

```
Enter the wine variety: Pinot Noir

Variety: Pinot Noir

Premium Wines:
        price  Vintage(Year)
357     350.0           2013
3069    205.0           2008
6730    285.0           2007
6731    280.0           2008
8891    250.0           2009
...       ...            ...
120440  279.0           2014
120594  250.0           2008
120595  236.0           2009
121944  360.0           2015
127566  350.0           2008

[123 rows x 2 columns]

Value Picks:
        price  Vintage(Year)
4        65.0           2012
21       20.0           2013
25       69.0           2011
35       50.0           2010
41       22.0           2009
...       ...            ...
129920   48.0           2006
129931  107.0           2005
129936   66.0           2005
129960   48.0           2006
129967   75.0           2004

[13150 rows x 2 columns]
```

Figure 48

Relation to the Main Problem and Sub Problem -Classification Model

The Logistic Regression model addresses the main problem and sub problem by classifying the wines into value picks and premium choices to optimise the wine selection process and enhance the customer experience.

This model helps to identify what wines are considered as value picks and premium choices based on points and price for each variety of wine. Since customers have different wine preferences and budgets, this would help the winery to cater to a broader range of customers and make quick recommendations and make strategic decisions regarding pricing and promotions.

Recommendations

Here are some recommendations based on our analysis:

1. USA is a leading source of wine production and has over 50,000 wineries. The company should focus on putting in more resources and funds towards the wineries in the US to produce more wine. It is followed by France and Italy.
2. The company should price their wines between 0 to $100. There are more than 600,000 wines that fall under this price range. In order to get a competitive hold on the market share, pricing wines in this range would allow them to target a larger consumer base.
3. Pinot Noir, Chardonnay, Cabernet Sauvignon and Red Blend make up a third of the total wines sold and hence the company should focus on producing and selling these wines as they are the biggest drivers of sales.
4. Pinot Noir wines are good Premium wines to recommend to consumers while Merlot wines are good value pick options for the regular consumer. Other premium recommendation would be: Ramisco and Terrantez. Other value pick suggestions could be: Chardonnay and Cabernet Sauvignon.

Conclusion

Through our intensive analysis of wine data, we conducted regression analysis to predict wine prices and employed a logistic regression model to classify wines into "Value Picks" and "Premium Picks." Our results and findings gave us valuable insights into the wine industry based on which the findings offer valuable insights for the business in the wine industry.

Regarding our regression analysis, Model 2 was more accurate than Model 1. It had a lower MSE and MAE, indicating a superior ability to predict wine prices. Moreover, the higher R-squared (R2) value also showed that Model 2 explains a greater proportion of price variance.

Secondly, our logistic regression model successfully classified wines into "Value Picks" and "Premium Picks" based on various attributes. This classification can significantly enhance our inventory curation, ensuring that customers have access to wines that align with their preferences and budget constraints.

The integration of the regression and classification models allows predicting wine prices and classifying them accurately. This holistic approach empowers us to optimize our product offerings, improve customer experiences, and tailor our inventory to meet diverse demands. It is crucial to maintain ongoing model monitoring and exploration of alternative modelling techniques to ensure sustained business success in the dynamic wine market.

References

Maurel, C, Pierrot, F, Cheriet, F, Amadieu, A 2019, *Inventories in the wine industry: From sector and financial determinants to strategic behaviors*, 13, Annual AAWE Conference, Jul 2019, Vienne, Austria, 4 p. ffhal-02789122f, viewed 08 Sep 2023 <https://ideas.repec.org/p/hal/journl/hal-02789122.html#>.

Navarro, M, Pedraja-Inglesias, M & Marta 2010, *Are there different profiles of wine tourists? An initial approach*, International Journal of Wine Business Research,20101109,Emerald Insight, viewed 09 Sep 2023, <https://research.ebsco.com/c/6kr4lr/details/hvrhgwingf?limiters=FT1%3AY&q=wine%20customers%20have%20different%20tastes>.

Gustafson, R, Lybbert, J, Sumner, A 2016 ,*Consumer sorting and hedonic valuation of wine attributes: exploiting data from a field experiment*, Agricultural Economics,2016,FSTA - Food Science and Technology Abstracts, viewed 09 Sep 2023, <https://research.ebsco.com/c/6kr4lr/details/uui6ffa5zv?limiters=FT1%3AY&q=wine%20price%20customers>.

Hagan, M 2020, *Wine History: Exploring Wine's History and Origins*, Usual, viewed 09 Sep 2023, <https://usualwines.com/blogs/knowledge-base/history-of-wine>.

Smith B 2019, Getting More Out of Wine: wine experts, wine apps and sensory science, Current Opinion in Food Science 2019, 27:123-129, viewed 09 September 2023, < https://www.sciencedirect.com/science/article/pii/S2214799319300165?via%3Dihub >.