Predicting Wine Rating

INFO574 Final Report

Zhiqing Sha, Sophie Guo, Fanxi Chen, Yitong Zhao

Table of Content

1. Introduction

The overall goal of this project is to build models to predict the wine rating on a scale of 1-5.

Although similar projects have been done to predict wine rating, our project adds value by exploring a wider set of predictors and comparing results on a more comprehensive set of models. The preliminary input to our models is features including winery name, year of harvest, richness and weight of wine, number of reviews given to the rating, price, type of wine, wine acidity, latitude of the region of wine, and longitude of region. We then built a variety of machine learning models including linear regression, random forest, boosted tree, and neural network to predict numeric ratings.

The rest of the report is organized as the following: Part 2 outlines the problem and background of the project. Part 3 details the data sources and how we manipulated the data. Part 4 lists the models built for the analysis and the reasoning for choosing the model. Part 5 summarizes the results across models. Part 6 discusses the insights from this project and possible improvement.

2. Problem Statement

The purpose of the study is to predict ratings for each name of wine with a set of predictors provided in the dataset. This is a prediction problem with the goal of achieving high accuracy and reducing error. We hypothesize that our model can predict wine ratings, or alternatively, at least one of the coefficients of our predictors is not zero.

Similar work has been conducted in the field. From the outcome perspective, [1] works on predicting quality rating using a classification method, [2] works on text review rather than numeric review. From the model perspective, [3] constructs XGBoost, ExtraTree and an ensemble model of the previous models. Speaking of features, the above studies either have extra predictors in the model, or use a subset of predictors we are interested in. While these are all good analyses, we further explored less common features including acidity, latitude and longitude, and compared results from a more comprehensive set of models.

3. The Data

3.1 Data Source

The data we used consisted of information from two datasets. The first set "Spanish Wine Quality Dataset" [4], which can be accessed from Kaggle, contains common characteristics of red wines in Spanish such as price, year, and type. The dataset also includes the key outcome we are predicting, user rating of each wine. The second set "City" [5] provides basic information on cities in Spain, in which we extract longitude and latitude for each wine. We registered and downloaded the dataset and API for all Spanish cities from the back4pp website for the city dataset. The meaning of features and outcome are provided in the table below.

Variables	**Meaning**
wine	Winery name
winery	Name of the wine
year	Grape harvest year

body	Richness and weight of wine in mouth
num_reviews	Number of users of the reviews
price	Price in euros (€)
type	Wine variety
acidity	Wine's “pucker” or tartness
region	Region of the wines
rating	*User provided rating on wine*
latitude	Latitude of the region
longitude	Longitude of the region

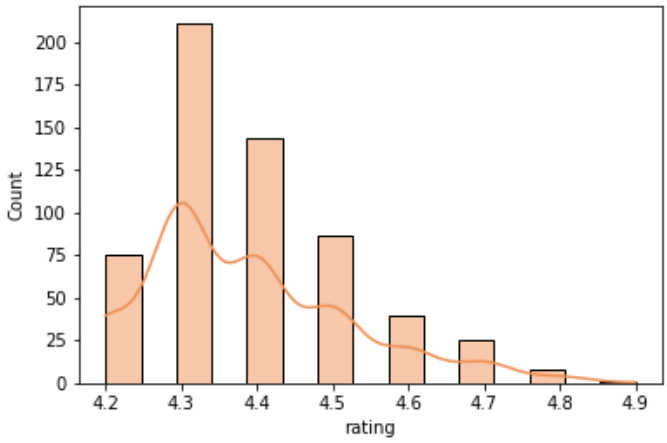
To clean the datasets, we first set text columns to lower case. Then we renamed column "name" to "region" in the City dataset and inner joined the two datasets on the “region” column representing different areas in Spain.

3.2 Data Description

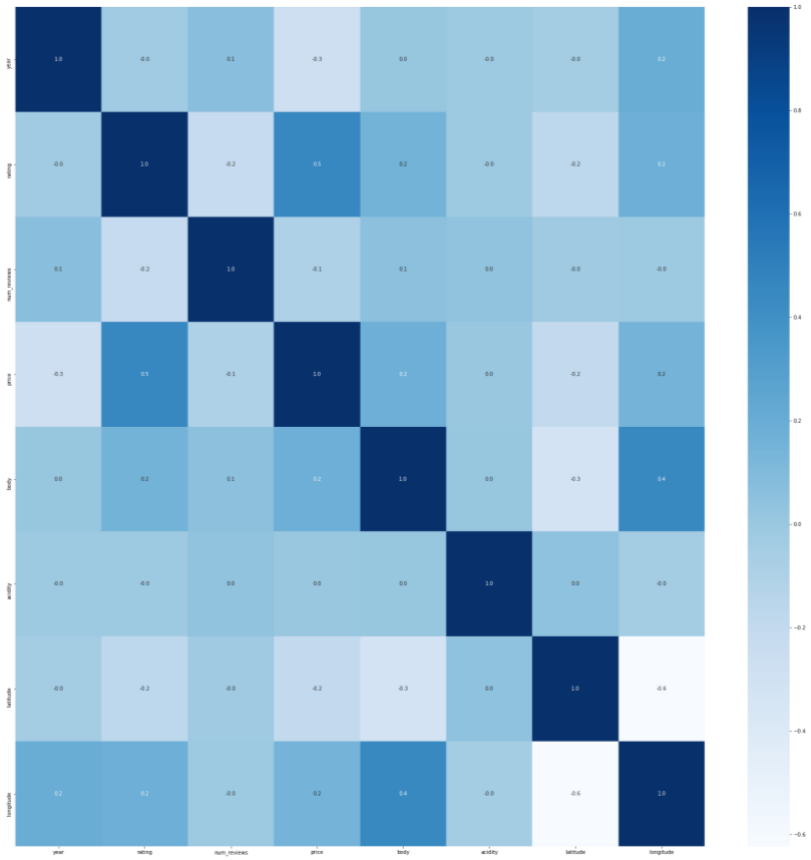
The cleaned dataset has nine features, one outcome variable with 591 rows. Out of the nine features, "winery" and "type" are categorical variables in text, in which dummy-coding produced 153 columns. Among the numeric predictors, "num_reviews" has the widest range with a standard deviation of 1329, followed by "price" with a deviation of 194. Other variables have relatively limited range, with a standard deviation as small as 0.1. Description of the numeric features can be found below.

	year	num_reviews	price	body	acidity	latitude	longitude
count	591.000000	591.000000	591.000000	591.000000	591.000000	591.000000	591.000000
mean	2010.231810	622.529611	100.791675	4.062606	2.989848	-2.735054	37.808174
std	11.561277	1328.712177	194.224480	0.488310	0.100331	1.223458	1.654867
min	1925.000000	25.000000	6.260000	2.000000	2.000000	-6.289100	36.526720
25%	2009.000000	57.000000	31.115000	4.000000	3.000000	-2.463020	36.945080
50%	2014.000000	176.000000	50.000000	4.000000	3.000000	-2.463020	36.945080
75%	2016.000000	667.500000	93.975000	4.000000	3.000000	-2.463020	37.466060
max	2021.000000	16505.000000	2814.160000	5.000000	3.000000	2.294510	41.757800

The outcome variable theoretically ranges from 1-5, while in the dataset it has values from 4.2 to 4.9. The overall shape of the outcome is normal.



To explore the connection between numeric features and the outcome, we created a heap map of correlation shown below. The highest correlation is 0.5, found between the outcome wine rating and the price of wine. This might indicate that price predicts wine rating well.



3.3 Data Preparation

On the merged data, null and improper values ("N.V." in the year column) were dropped, and text (categorical) columns were dummy-coded. There are 134 categories for "winery", and 11 categories for "type", making up 145 dummy columns. Adding up to the pre-existing 8 features, there are 153 columns in total. Duplicated rows and unrelated columns were then

dropped, and the dataset was split into a training set and a testing set with an 8:2 ratio and a random seed = 42 for reproduction. The resulting training set has 472 observations, and the testing set has 119 observations. It should be noted that the outcome variable was not scaled even though its actual range unmatched theoretical range, as we tend to see higher scores (4-5) in real life.

```
# train-test split for dummy-coded data
X = clean[[c for c in clean.columns if c != "rating"]]
y = clean["rating"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
training set: (472, 152)
testing set: (119, 152)
```

Apart from the dummy-coded dataset, we also prepared a version in which scaling was performed on numeric columns. These columns, as mentioned above, include year, num_reviews, price, body, acidity, latitude, longitude. This dataset is specifically prepared for generating feature importance in tree models.

4. Methodology

Model 1: Multiple Linear Regression

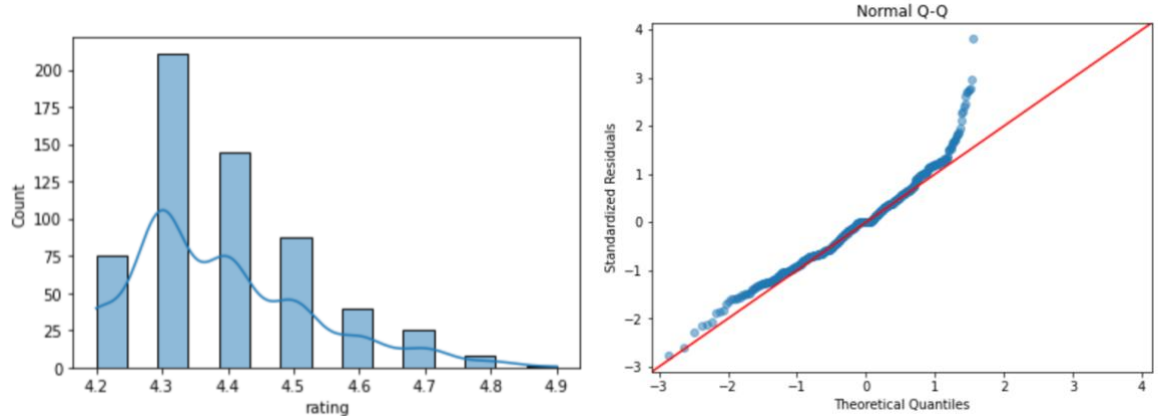
Because of the ease of use and interpretability, as well as the nature of outcome, the first model we decided to use is multiple linear regression. Multiple regression analysis can be used to determine the strength of the relationship between an outcome (dependent variable) and several predictor variables, as well as the significance of each predictor variable in this relationship. This enables the statistical removal of the effects of other predictor variables [6]. The outcome of this model is the rating of the wine. The preliminary prediction variables are price, body, acidity, type, winery, latitude and longitude. Since winery and type are categorical value, we used "winery_age" and "type_red" as the reference case, respectively for the two variables. The features and outcome were fitted to the model with the training set, and a prediction was made on the testing set with the predict() function. With the fitted values, performance was evaluated with MSE and MAPE scores.

OLS Regression Results

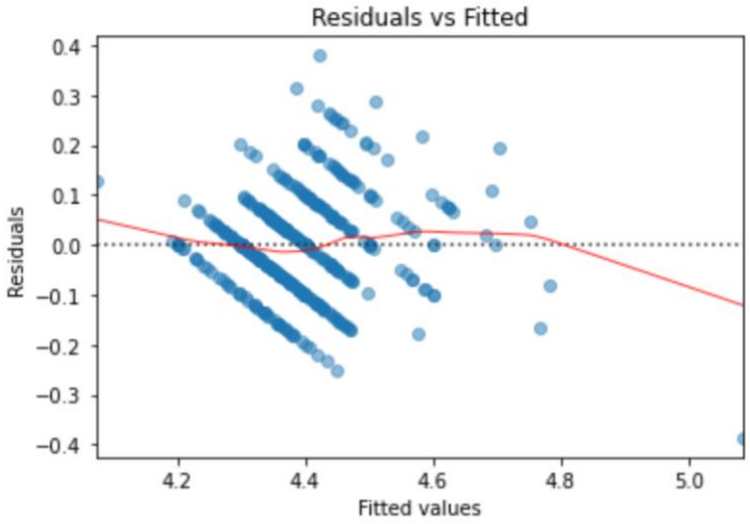
Dep. Variable:	rating	**R-squared:**	0.541
Model:	OLS	**Adj. R-squared:**	0.366
Method:	Least Squares	**F-statistic:**	3.092
Date:	Fri, 16 Dec 2022	**Prob (F-statistic):**	7.71e-17
Time:	18:02:33	**Log-Likelihood:**	449.84
No. Observations:	472	**AIC:**	-637.7
Df Residuals:	341	**BIC:**	-93.12
Df Model:	130		
Covariance Type:	nonrobust		

Several assumptions for linear regression were checked. (1) Normality: we produced a histogram of rating to see if the outcome is normally distributed. As seen in the graphs below,

rating is roughly normal.



(2) Constant variance: we generated a scatter plot of standardized residuals versus predicted values to check homogeneity. There seems to be a trend in the residuals, which implies that the constant variance assumption might be violated.



(3) Linearity: referring to the same plot above, it is expected that points randomly sit around both sides of the zero line. It is not the case here, and we concluded that the data does not have a linear relationship between features and outcome.

Together with the assumption checking, we also performed a test on multicollinearity with VIF (Variance Inflation Factor). What we found is that many predictors have high VIF values, indicating a strong probability of multicollinearity. We hypothesized that this is due to the strong correlation between some winery category or type category and other features, such as a strong relationship between certain winery and body, the richness and weight of wine in mouth because the winery is better at producing wine. With this hypothesis, we tested multicollinearity without winery and type in the model and obtained VIF scores that look much healthier, even though longitude seems to have a relatively high VIF. A refined model is further built based on the listed variables only.

	variables	VIF
0	const	0.000000
1	year	2.257057
2	num_reviews	1.497240
3	price	1.749848
4	body	499.122680
5	acidity	inf
6	latitude	inf
7	longitude	inf
8	winery_abel mendoza monge	1.636347

	variables	VIF
0	const	36119.336057
1	year	1.164458
2	num_reviews	1.027087
3	price	1.147054
4	body	1.402802
5	acidity	1.010652
6	latitude	1.842788
7	longitude	2.289779

(Left) The first nine VIF scores of the predictors; (Right) All VIF scores without winery and type

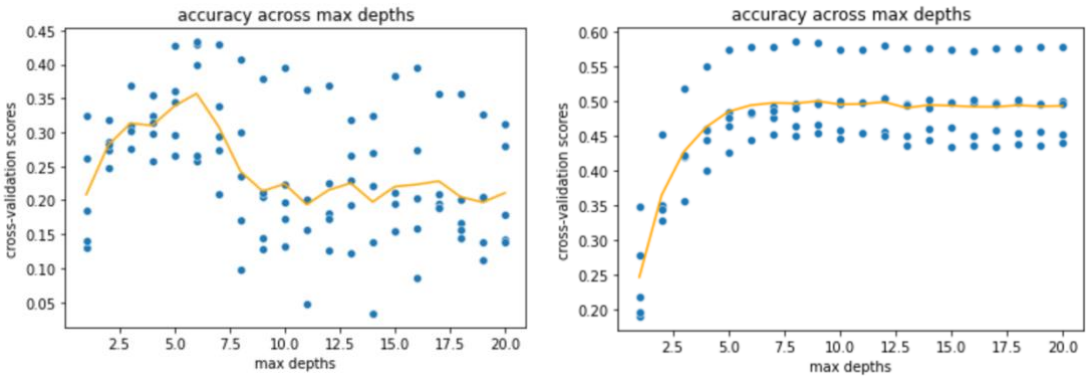
The new model was built with seven numeric predictors only. Although the R-squared is relatively lower, we trust this model better since the assumption that the model has low, or no multicollinearity is satisfied.

OLS Regression Results

Dep. Variable:	rating	**R-squared:**	0.308
Model:	OLS	**Adj. R-squared:**	0.298
Method:	Least Squares	**F-statistic:**	29.54
Date:	Fri, 16 Dec 2022	**Prob (F-statistic):**	1.00e-33
Time:	18:02:35	**Log-Likelihood:**	353.04
No. Observations:	472	**AIC:**	-690.1
Df Residuals:	464	**BIC:**	-656.8
Df Model:	7		
Covariance Type:	nonrobust		

Model 2: Decision Tree & Random Forest

From the assumption checking of multiple linear regression, we saw that several assumptions were not satisfied. Therefore, we decided to turn to Decision Tree and the Random Forest for their abilities to handle data in non-linear shape, cope with multiclass and manage outliers. Similar to the previous model, we train the models with the training set, and then perform prediction on the testing set to gain performance result. To tune the depth parameter, we put the training data into a decision tree model with for-loop to find optimal max depth with cross-validation scores. The cross validation was set to a common 5-fold with seed 33 for replication, and the depth with the highest score (accuracy by default) was selected. The depth identified for decision tree was 6, and for random forest was 9. Then we put these max depths into the corresponding model to make predictions with test data.



(Left) tunning max depth for Decision Tree; (Right) Tunning max depth for Random Forest

Following predictions on test set, we used MSE (mean squared error) and MAPE (mean absolute error percentage) to test the accuracy of predictions by comparing the errors between predicted values and true values. Finally, we also evaluated feature importance by using the feature importance function in both models. What is special about this analysis is that while the models were built on raw data, feature importance was produced on the scaled version to avoid overestimation of large-scale predictors such as price and num_reviews, hence a higher precision on ranking ability to reflect prediction power is expected.

Model 3: Deep Neural Network

Like tree models, Neural Network can handle data without normality or linearity well. Considering our data does not look linear and outcome is just roughly normal, the third model we used in our analysis is deep neural networks. Our primary tool with deep neural networks is the Keras module of Google TensorFlow. Details of each step are described below.

(1) Data normalization of predictors' variables. Normalization before data input can effectively improve the convergence speed and effect. Therefore, in the first step, we use z-scores to normalize the predictors.

(2) Prepare training and test datasets just as in previous models, including separating features (dummy-coded) and outcome, and performing a train-test split with seed 42.

(3) Import the data into the to-be-established neural network, adding two hidden layers. The parameters including number of layers and number or nodes were set based on the default values recommended by Keras. “relu” was used as the activation function as this is a regression problem.

```
model = Sequential()
model.add(Dense(25, input_dim=x.shape[1], activation='relu')) # Hidden 1
model.add(Dense(10, activation='relu')) # Hidden 2
model.add(Dense(1)) # Output
```

(4) Each iteration, the network is optimizing based on standard MSE (mean square error).

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

(5) Set up 1000 epochs to train the neural network. Add Early Stopping to the training process of the neural network. The purpose of adding this step is to stop the

training before overfitting and to obtain the training model with the best MSE score. Details about the parameters are provided below.

```
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3,
                        patience=5, verbose=1, mode='auto',
                        restore_best_weights=True)
model.fit(x_train,y_train,validation_data=(x_test,y_test),
          callbacks=[monitor],verbose=2,epochs=1000)
```

The epoch training details, and results of neural networks are shown below. Since the output display is lengthy, only partial result will be shown).

```
15/15 - 0s - loss: 0.0224 - val_loss: 0.3437 - 50ms/epoch - 3ms/step
Epoch 60/1000
15/15 - 0s - loss: 0.0213 - val_loss: 0.3342 - 61ms/epoch - 4ms/step
Epoch 61/1000
15/15 - 0s - loss: 0.0211 - val_loss: 0.3297 - 54ms/epoch - 4ms/step
Epoch 62/1000
15/15 - 0s - loss: 0.0205 - val_loss: 0.3336 - 70ms/epoch - 5ms/step
Epoch 63/1000
15/15 - 0s - loss: 0.0203 - val_loss: 0.3349 - 49ms/epoch - 3ms/step
Epoch 64/1000
15/15 - 0s - loss: 0.0197 - val_loss: 0.3226 - 66ms/epoch - 4ms/step
Epoch 65/1000
15/15 - 0s - loss: 0.0193 - val_loss: 0.3233 - 51ms/epoch - 3ms/step
Epoch 66/1000
15/15 - 0s - loss: 0.0188 - val_loss: 0.3200 - 50ms/epoch - 3ms/step
Epoch 67/1000
15/15 - 0s - loss: 0.0184 - val_loss: 0.3204 - 59ms/epoch - 4ms/step
Epoch 68/1000
15/15 - 0s - loss: 0.0185 - val_loss: 0.3262 - 63ms/epoch - 4ms/step
Epoch 69/1000
15/15 - 0s - loss: 0.0178 - val_loss: 0.3119 - 55ms/epoch - 4ms/step
Epoch 70/1000
15/15 - 0s - loss: 0.0174 - val_loss: 0.3085 - 74ms/epoch - 5ms/step
Epoch 71/1000
15/15 - 0s - loss: 0.0169 - val_loss: 0.3071 - 69ms/epoch - 5ms/step
Epoch 72/1000
15/15 - 0s - loss: 0.0166 - val_loss: 0.3119 - 55ms/epoch - 4ms/step
Epoch 73/1000
15/15 - 0s - loss: 0.0166 - val_loss: 0.3101 - 53ms/epoch - 4ms/step
Epoch 74/1000
15/15 - 0s - loss: 0.0159 - val_loss: 0.3023 - 66ms/epoch - 4ms/step
Epoch 75/1000
15/15 - 0s - loss: 0.0160 - val_loss: 0.3091 - 65ms/epoch - 4ms/step
Epoch 76/1000
15/15 - 0s - loss: 0.0158 - val_loss: 0.2998 - 71ms/epoch - 5ms/step
Epoch 77/1000
15/15 - 0s - loss: 0.0155 - val_loss: 0.3077 - 52ms/epoch - 3ms/step
Epoch 78/1000
15/15 - 0s - loss: 0.0151 - val_loss: 0.3003 - 49ms/epoch - 3ms/step
Epoch 79/1000
15/15 - 0s - loss: 0.0148 - val_loss: 0.3049 - 52ms/epoch - 3ms/step
Epoch 80/1000
15/15 - 0s - loss: 0.0147 - val_loss: 0.3008 - 53ms/epoch - 4ms/step
Epoch 81/1000
Restoring model weights from the end of the best epoch: 76.
15/15 - 0s - loss: 0.0144 - val_loss: 0.3011 - 57ms/epoch - 4ms/step
Epoch 81: early stopping
<keras.callbacks.History at 0x7f748e325f70>
```

(6) Input predictors' values from the test dataset into the trained neural network model to predict the prediction accuracy based on the test dataset and obtain the MSE,

MAE, and MAPE scores. The code and result will be shown in the next "Result" section.

5. Results

5.1 Multiple Linear Regression 1

We obtained an R-square value of 0.541 for our linear regression model, which indicates that 45.1% of the data fit the regression model. We also evaluated performance with two error metrics. As we learned, the lower the value, the better the model's performance. The mean squared error for this model was 0.0253, which was excellent. Our MAPE was 2.3996%. A MAPE of less than 5% is considered to have acceptable forecast accuracy, so this model is considered acceptable for the prediction purpose.

Speaking of predictors, we found that year, num_reviews, and price have a p-value lower than 5%, indicating a high significance in this model. Even though the coefficients were small, if we consider the fact that ratings were in a limited range of 4.2-4.9 and relatively large range of these predictors, these three features are very important in predicting the rating. The below picture shows the shortened version of result from regression as there are many dummy variables.

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0996	0.188	0.531	0.596	-0.270	0.469
year	0.0017	0.001	2.346	0.020	0.000	0.003
num_reviews	-1.477e-05	4.47e-06	-3.303	0.001	-2.36e-05	-5.97e-06
price	0.0004	4.37e-05	9.591	0.000	0.000	0.001
body	0.0780	0.126	0.619	0.537	-0.170	0.326
acidity	-0.0663	0.296	-0.224	0.823	-0.648	0.515
latitude	0.0652	0.041	1.573	0.117	-0.016	0.147
longitude	0.0182	0.037	0.496	0.620	-0.054	0.090
winery_abel mendoza monge	0.0762	0.142	0.536	0.592	-0.203	0.356
winery_allende	0.0681	0.119	0.571	0.568	-0.167	0.303
winery_alonso & pedrajo	0.0319	0.162	0.197	0.844	-0.286	0.350
winery_alta alella	0.1075	0.213	0.504	0.614	-0.312	0.527
winery_altanza	0.0508	0.139	0.366	0.715	-0.222	0.324
winery_altos de rioja	-0.0113	0.140	-0.081	0.936	-0.286	0.264
winery_alvaro palacios	0.1066	0.142	0.753	0.452	-0.172	0.385
winery_artadi	-0.0078	0.138	-0.057	0.955	-0.279	0.263
winery_atalaya	0.0336	0.093	0.361	0.719	-0.149	0.217

5.2 Multiple Linear Regression 2

We obtained an R-square value of 0.308 for this linear model with a subset of predictors from the previous. The model witnessed a slight change in coefficients, while significance remains roughly the same.

Similar to the interpretation in the previous linear model, we see that *year*, *num_reviews* and *price* are significantly powerful in the prediction. Since we excluded *winery* and *type* in this model, the left variables are expected to take a higher share of prediction power than in the previous model.

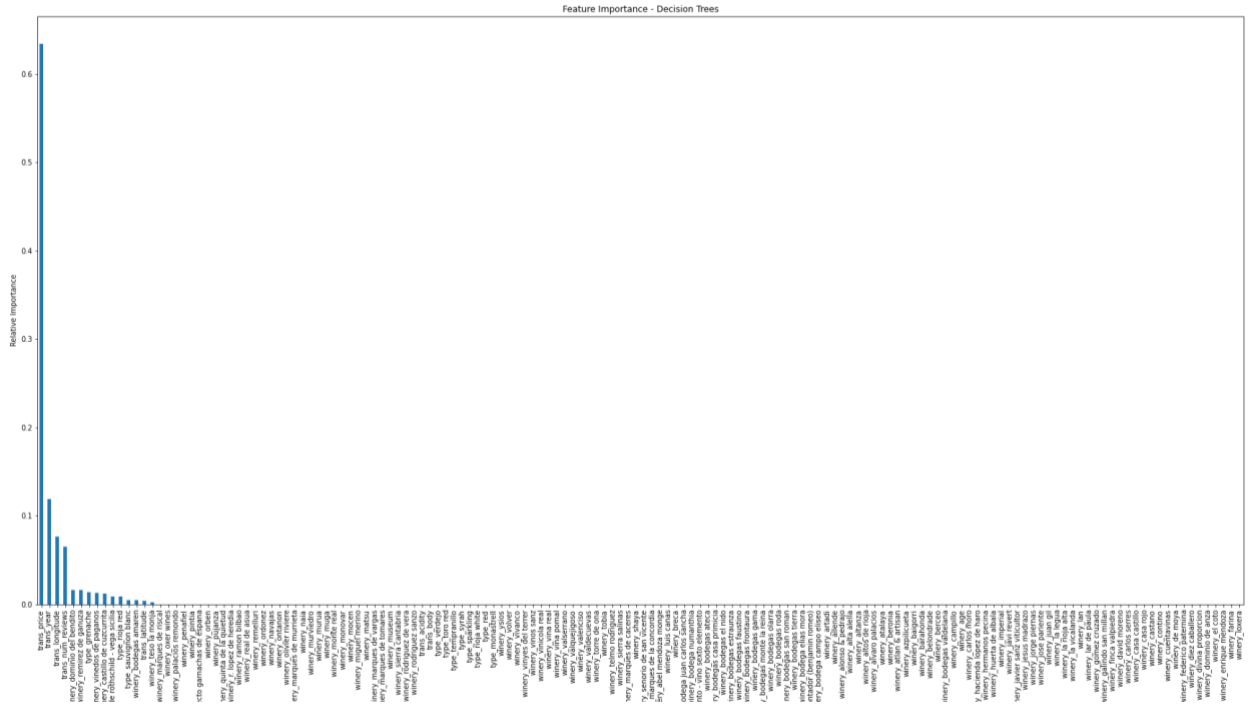
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.8485	1.010	0.840	0.401	-1.137	2.834
year	0.0017	0.001	3.288	0.001	0.001	0.003
num_reviews	-1.623e-05	3.89e-06	-4.173	0.000	-2.39e-05	-8.59e-06
price	0.0005	3.72e-05	12.405	0.000	0.000	0.001
body	0.0188	0.013	1.417	0.157	-0.007	0.045
acidity	-0.0035	0.052	-0.067	0.947	-0.106	0.099
latitude	-0.0035	0.006	-0.576	0.565	-0.015	0.008
longitude	0.0014	0.005	0.288	0.773	-0.008	0.011

Omnibus:	28.055	**Durbin-Watson:**	2.200
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	31.437
Skew:	0.621	**Prob(JB):**	1.49e-07
Kurtosis:	3.242	**Cond. No.**	4.15e+05

5.3 Decision Tree

In this model, we got the max depth = 6, and the fitted tree model predicted test outcome with an MSE of 0.14851 and MAPE around 2.23615%. These errors are slightly higher than in linear regression, but we believe decision tree produces a more reliable result as it does not assume a linear shape of data.

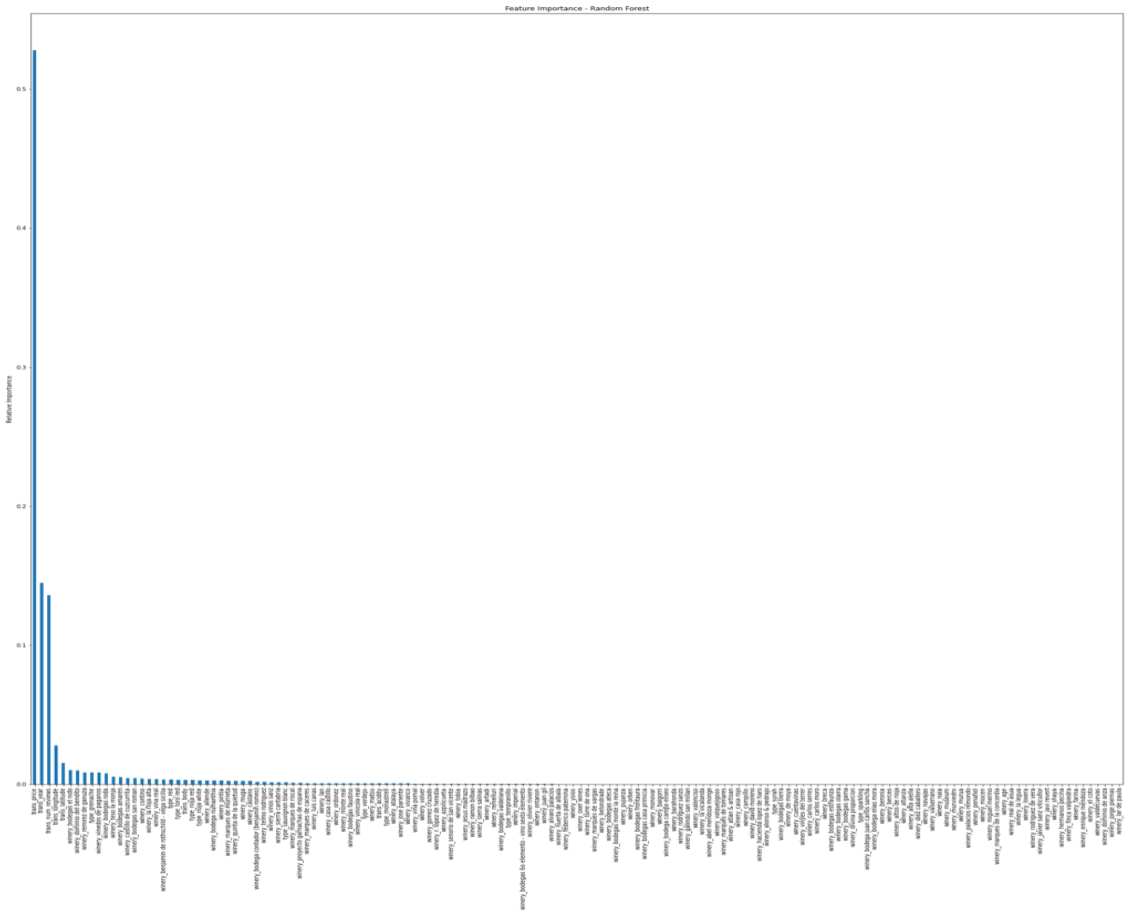
We also created a plot of feature importance to identify top predictors. We found that *price*, *number of reviews*, *year*, and *longitude* are the most relevant to the ratings. In other words, they have a strong impact on the prediction of rating outcome. More specifically, price is the most significant factor, with an importance score over 0.6 according to the graph.



5.4 Random Forest

The prediction on test set with max depth 9 obtained an MSE of 0.0102 and MAPE around 1.8007%, which are the best scores among fitted models.

The feature importance scores indicated that several factors: *price*, *year*, *the number of reviews*, *longitude*, *latitude*, *some wineries categories*, and *some types* have leading predicting power on wine rating. Compared to the decision tree, a few more factors are involved as the leading factors in the prediction model. We think the lower errors and higher number of top predictors match the fact that random forest has a more flexible decision boundary than decision tree and it de-correlates the trees to improve accuracy.



5.5 Neural Network

The neutral network reached the early stopping at the 81st epoch. The model was trained and yielded an MSE score of 0.2998, a MAE score of 0.1839, and a MAPE score of 5.4202. Compared to previous models, neural network produced a medium MSE, but a highest mean absolute percentage error, meaning that its prediction is the furthest from actual values in terms of percentage.

```
from sklearn import metrics

# Predict
pred = model.predict(x_test)

# Measure MSE error.
score = metrics.mean_squared_error(pred,y_test)
print("Final score (MSE) by deep neural network: {}".format(score))
```

```
4/4 [==============================] - 0s 3ms/step
Final score (MSE) by deep neural network: 0.299828658973319
```

```
# Predict
pred = model.predict(x_test)

# Measure MAE error.
score = metrics.mean_absolute_error(pred,y_test)
print("Final score (MAE) by deep neural network: {}".format(score))
```

```
4/4 [==============================] - 0s 6ms/step
Final score (MAE) by deep neural network: 0.18388938554445228
```

```
# Predict
pred = model.predict(x_test)

# Measure MAPE error.
score = np.mean(np.abs((y_test-pred)/y_test)) * 100
print("Final score (MAPE) by deep neural network: {}".format(score))
```

```
4/4 [==============================] - 0s 5ms/step
Final score (MAPE) by deep neural network: 5.420169816814108
```

5.6 Model Comparison

Overall, all four models produced low error scores. We believe this is because ratings are in the range of 4.2 to 4.9, in which prediction cannot be too deviated from actual value to yield a high error. However, we can still identify the best model with the lowest error even though the scale is tiny. Across the models, we see that random forest produced the highest accuracy, reflected by the lowest MSE 0.0102 and MAPE 1.8007%. Identified top predictors in this model are *price*, *year*, *the number of reviews*, *longitude*, and *latitude*.

	Linear Regression 1	**Linear Regression 2**	**Decision Tree**	**Random Forest**	**Neural Network**
MSE	0.0253	0.0216	0.014	0.0102	0.2998
MAPE	2.3996%	2.3080%	2.1328%	1.8007%	5.4202%
R^2	0.541	0.308			

6. Conclusion

The analysis results confirmed our hypothesis that our model can predict wine ratings well

with several predictors, including *price*, *year of harvest*, *the number of reviews*, *longitude*, *latitude*, etc. The final model we chose is random forest with a depth of 9, which produced the best performance result. Surprisingly, we did not see much importance of acidity and body in the prediction, which we expected to have at least moderate influence on the outcome. Thinking about their definition, we believe these variables are very subjective to individual definitions, and hence hard to summarize any pattern out.

In future work, there could be some improvements to make. First, we may adopt some techniques to decrease the number of dummy variables. In our case, we have more than a hundred dummy variables, which may increase multicollinearity and weaken the performance on prediction precision. To fix the problem, we may search for studies about winery type, location, production focus, or even owner, to learn about the differences between wineries, in order to pick a useful subset to fit in our model. Other techniques like PCA can be used to achieve the same goal. Alternatively, we may merge some wineries together based on pre-existing knowledge about them and treat them as one category.

In addition to dealing with dummies better, to produce a better prediction and more accurate model, we may need to explore more datasets and extract the useful features that help to predict the wine rating. For example, in future work, we can combine the datasets with more features that are related to per wine, such as chemical properties, which have not been studied a lot with market features like price. We can perform proper tests and analyses to examine whether these related features also potentially influence the wine rating or not.

Talking about the models we built, there is also room for enhancement. Some tunning parameters in our model were not iteratively tested to identify the best values. For instance, the value of the number of trees to use in random forest (n_estimators), the number of layers and number of nodes in neural networks were selected simply based on the recommended or default provided in the package. While the resulting prediction accuracy is high, there could be space for even better performance. To tune these parameters, cross-validation can be used. Furthermore, different sets of features can be fitted in the model to yield increasing accuracy, such as interaction terms, or higher power terms.

To further increase the prediction accuracy, outlier analysis might be performed. Specific scores like cook's distance and leverage score can be calculated to see if points were outside of the normal range. Even though tree models and neural network can handle outliers fairly well, their influence can be further limited by exclusion of outliers from the training model.

Finally, more data can be collected to perform a more thorough analysis. The outcome variable in the current dataset was concentrated in the 4.3 rating, and even though the range of rating is 1-5, none of the ratings was under 4. To predict a full range of ratings, data in the relatively lower score range should be collected and combined with what we currently have, in order to generalize the results better in real life.

We met many challenges and learned how to apply knowledge to utilization in this project. We hope that our analysis would help people to know about the wine market better such as customer needs, product information, etc. One day, we may be able to pick a good wine based on the analysis information.

7. References

[1] Predicting Wine Quality with Several Classification Techniques https://towardsdatascience.com/predicting-wine-quality-with-several-classification-techniques-179038ea6434

[2] Sentiment Analysis with Wine Reviews https://dscrashcourse.com/practice/wine/

[3] Spanish wine: Cleaning, EDA & Modeling) https://www.kaggle.com/code/mahmoudelhusseni/spanish-wine-cleaning-eda-modeling/notebook

[4] Spanish Wine Quality Dataset https://www.kaggle.com/datasets/fedesoriano/spanish-wine-quality-dataset

[5] City Dataset https://www.back4app.com/database/back4app/list-of-all-cities-in-finland

[6] https://www.sciencedirect.com/topics/economics-econometrics-and-finance/multiple-regressionanalysis#:~:text=Multiple%20regression%20analysis%20allows%20researchers,of%20other%20predictors%20statistically%20eliminated

[7] https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class_04_3_regression.ipynb