**Wine Recommender System Analysis**

| Shane Nelson  *Drexel University*  [sn888@drexel.edu](mailto:sn888@drexel.edu) | Allie Schneider  *Drexel University*  [as5664@drexel.edu](mailto:as5664@drexel.edu) | Francis Villamater  *Drexel University*  [fv48@drexel.edu](mailto:fv48@drexel.edu) | Will Wu  *Drexel University*  [ww437@drexel.edu](mailto:ww437@drexel.edu) |
| --- | --- | --- | --- |

**Abstract** - The goal of this project is to construct a recommender system which can be used by both novice consumers and experienced wine sommeliers alike to provide recommendations for new wines to try. The obtained dataset contains ratings for thousands of wines and the commonly used characteristics to categorize them including, but not limited to, variety, country/province/region, winery, price, designation, and even a user-inputted description. Although it is not immediately clear whether all features are readily usable or helpful as-is, this paper explores the data acquisition, analysis, and pre-processing steps taken to ensure the data is suitable for accomplishing project goals.

**Keywords** - Wine, WineEnthusiast, Recommender System, Python, Machine Learning

**GitHub Repository** - <https://github.com/snelson97/DSCI591_G8SAFW>

1. **Introduction**

Wine is one of the most popularly consumed alcoholic beverages in the world, with multiple different varieties being produced by thousands of different wineries found in many countries around the globe. Although it may be easy for an experienced wine taster to identify and select a wine based on their individual knowledge, the average consumer would almost certainly not consider themself a wine connoisseur and may become overwhelmed with the myriad combinations of region, winery, type, and description of a wine.

Of the different types of recommender systems which exist, a content-based system and a knowledge-based system both stand out as being types of systems which would be particularly relevant. The goal with the content-based system is to utilize features in the data to characterize the qualities of a wine (the “content”), then take user reviews/ratings and identify a single, “target” user for which the system could predict their rating of un-reviewed wines in order to provide recommendations for new wines to try based on the highest predicted ratings. The deliverable of the content-based system is to take wine review data for a target user as input, and provide a list of recommended (un-reviewed) wines as output.

One type of knowledge-based recommender, constraint-based, takes input qualities from a user, (for example description keyword(s), country/region, winery, price, etc.) and provides an output of related wines based on the search criteria and ranked by other user ratings. Alternatively, a case-based knowledge recommender takes an input wine name/ID and provides a list of similar wines to try. The deliverable of the knowledge-based system is to take input search criteria or a specific example of wine and provide relevant, related wines based on similar characteristics. Another potential option is to use clustering to group wines based on shared characteristic combinations and provide recommendations based on wines within the same cluster ranked by average user rating.

Another potential option is to use clustering to group wines based on shared characteristic combinations and provide recommendations based on wines within the same cluster ranked by average user rating.

The system is applicable to a wide range of users from novice wine consumers to well-versed wine sommeliers. A novice user can use this system to help them to find new wines that they enjoy. The system can also aid novice wine consumers in cultivating a better understanding of their palate for wine. Additionally, the recommender system can provide guidance and education that will help the novice wine consumer understand the complexities of wine by offering insights into different grape varieties, regions, and styles. This knowledge empowers novice wine consumers to make more informed decisions and enhances their overall appreciation of wine. Finally, the system can save a novice wine consumer valuable time and energy in their wine selection process. Instead of feeling overwhelmed by an extensive wine list or relying on guesswork, they can rely on the system’s recommendations, which are tailored to their preferences and curated based on expert knowledge.

A well-versed wine sommelier can use this system to expand the breadth of their experiences. The system can serve as a valuable tool for discovering new and obscure wines, expanding the wine sommelier’s knowledge beyond the existing repertoire. The system may also bring to light different perspectives and suggestions by helping the wine sommelier to explore different wine styles and regions. The system may empower wine sommeliers to elevate their wine selection skills, provide personalized recommendations, and deliver exceptional experiences to their clientele.

1. **Dataset**

The Wine Recommender System uses a dataset that was acquired from Kaggle.com. The dataset was scraped from WineEnthusiast in 2017 by a Kaggle user, and includes nearly 120,000 unique wine reviews provided by wine sommeliers. Each of the 120,000 reviews included in this dataset have been reviewed by a wine enthusiast who recorded information about their review. The information recorded from each wine review includes the country the wine was made, a description of the wine, a designation for each wine, a points or score of how much the wine was enjoyed, the price of each bottle of wine, the province and region the wine was made, the reviewer’s name and twitter handle, the title of the wine, the variety of the wine, and the winery that made the wine for a total of 13 attributes.

This dataset has been selected for the use of the Wine Recommender System because of the wide range of information that was recorded for each wine that was reviewed. The inclusion of a description of the wine straight from a wine enthusiast will help the system to understand how wines differ from each other. The points or score of how much the wine was enjoyed by the wine enthusiast gives the system a numeric value for the quality and taste of the wine. The designation of wine variety gives the system the ability to categorize wine. The country, province and region of each wine allows the system to compare wines that were made thousands and thousands of miles apart, as well as compare wines that were made within the same geographic area. Finally, the price of each bottle of wine will allow the system to determine how accessible each wine is to the average consumer.

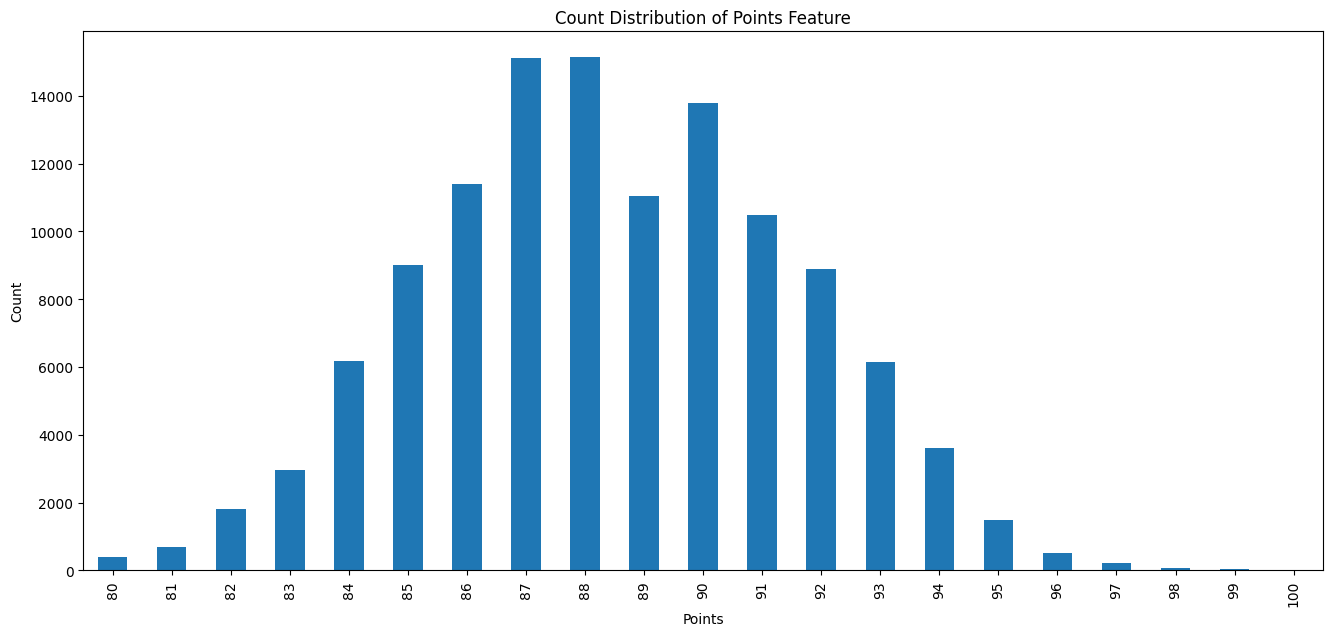
WineEnthusiast (winemag.com) started as a monthly print magazine and has expanded into an acclaimed, multifaceted media brand offering of-the-moment content in the print and digital publishing space[1]. Wine Enthusiast has over 4 million readers and considers itself as the most influential voice in wine and drinks journalism today. WineEnthusiast offers perspectives, stories and insights on wine and drinks. WineEnthusiasts has a global network of editors, writers, and tasters which allow for an accessible but expert view on the world of wine. WineEnthusiast offers 10 annual glossy magazine editions, a website (winemag.com), a biweekly podcast, a wine review buying guide, and virtual and in-person events.

Additionally, users will input data to identify information they are looking for in their recommended wine. A user may input information such as: preferred country, preferred price range, relevant keywords for a wine they enjoy, etc.

Our source data is stored in a tabular structured format and is stored in a csv file. Each row corresponds to a single wine review and each column represents a different attribute or feature of the wine. Additionally, users will input numeric and string data to identify the wine characteristics they are looking for.

1. **Exploratory Data Analysis**

Starting with the target attribute, ‘Points’, which is a discrete variable with a range from 80-100 increasing by 1, for a total of 21 levels. The most common value in ‘points’ is ‘88’ at 15,141 (12.7%), followed by ‘87’ at 15,126 (12.7%), and ‘90’ in third at 13,787 (11.6%). When looking at the distribution on a histogram (Figure 1), within this range from 80-100, the distribution is fairly normal with, if anything, a very slight right skew since the median, 88, is marginally less than the mean, 88.44.



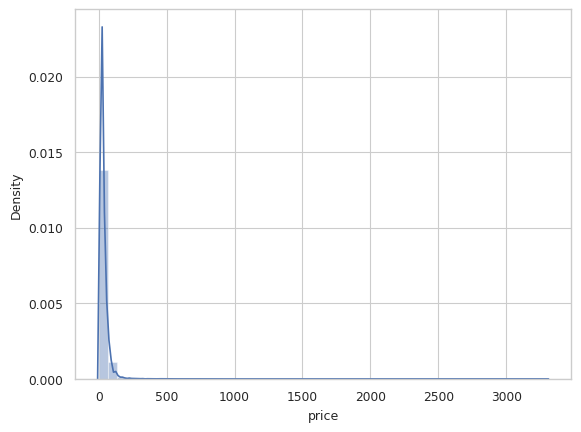
***Figure 1: Distribution of Points***

Analysis of the ‘description’ attribute is limited considering it was used as a free-text field. However, as mentioned above, it contains 118,938 unique values. The attribute ‘designation’ is also a free-text field and categorical with 37,979 unique values. The value ‘Reserve’ is most common at 1,870 (2.2%). However, a quick scan of this attribute will reveal that not all values are written in English. The top five values are ‘Reserve’, ‘Estate’, ‘Reserva’, ‘Riserva’, ‘Estate Grown’, respectively; here we can see three variations of the word ‘Reserve’.

|  | Points | Price ($) |
| --- | --- | --- |
| Count | 118,971 | 110,706 |
| Mean | 88.44 | 35.58 |
| SD | 3.09 | 41.88 |
| Minimum | 80 | 4 |
| 25th %ile | 86 | 17 |
| 50th %ile | 88 | 25 |
| 75th %ile | 91 | 42 |
| Maximum | 100 | 3,300 |

***Table 1: Descriptive Statistics for numerical attributes (SD: Standard Deviation)***

The ‘price’ attribute is visibly heavily skewed to the right as seen in the histogram (Figure 2), which is supported by the fact that



***Figure 2: Distribution of Price***

the mean ($35.36) is more than 10 price points higher than the median ($25) and price ranging from $0-$3,300.

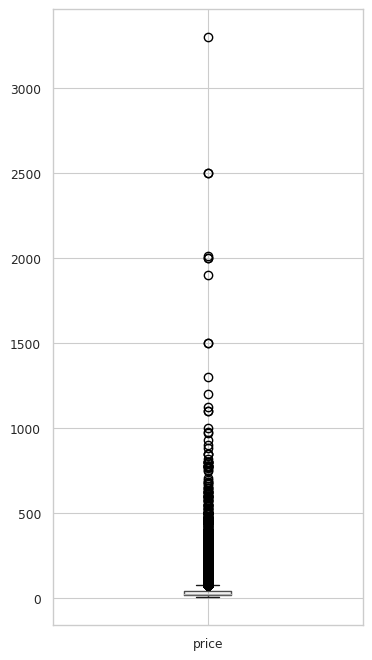
Under our attribute “country”, data is heavily dominated by ‘US’ at 50,298, comprising just over 42% of the entire dataset. The second and third most common values are ‘France’ at 19,771 (17%) and ‘Italy’ at 17,830 (15%), respectively. There are only 42 unique values but the top 3 values already make up almost 74% of the dataset, which indicates that a more specific location feature would be beneficial. The province feature is much better, with 422 unique provinces providing a better level of detail than the country feature, however once again the data is still dominated by California wines (>30%). California is much larger geographically than any other province included in the dataset, and it would be unfair to consider Northern California and Southern California wines to be from the same location, therefore we must continue to explore the region features and their usability. Region\_2 is only populated for provinces of California, New York, Oregon, and Washington and provides a great level of detail compared to just using the province feature, however there is a significant amount of null values for all countries outside of the US and also for some excluded US states. Region\_1 is an extremely specific feature with over 1,200 unique values, however there are still a large number of null values (>10%).

**Missing Values**

We encountered 8,265 missing values in the price column. Since this represents

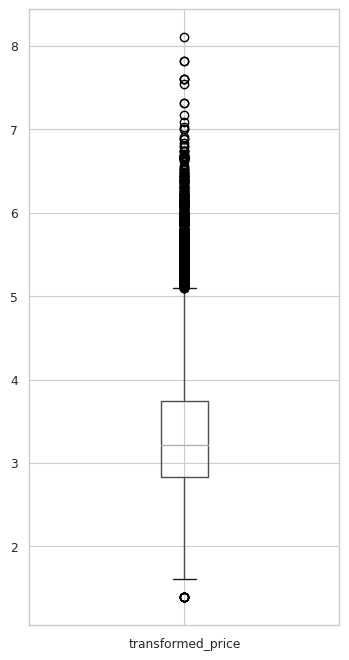
approximately 7% of the total 118,971 rows, we have decided to remove these rows with missing price values. Even after dropping them, we will still have over 110,000 rows available for analysis. Similar to the approach taken with the price column, we have also chosen to drop the missing values in the country and variety columns. By removing the missing values in the country column, any associated missing values in the province column will be automatically eliminated. Furthermore, the variety feature only contains a single missing value (which could be a result from data entry error) and we will be dropping it as well.

**Outliers**

****

***Figure 4: Price before transformation***

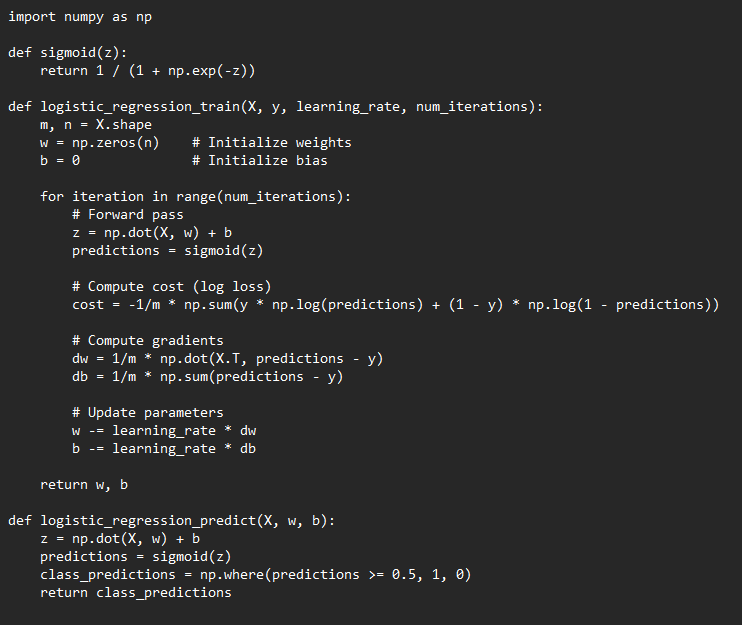
Regarding outliers, our primary concern lies within the price feature. Figure 4 highlights the presence of numerous extreme outliers in the price column, significantly impacting the statistical outcomes of the column. To address this potential issue, we have chosen to employ a logarithmic transformation on the column. This transformation proportionally reduces all values, eliminates outliers, adjusts skewness, equalizes variance, and lowers the standard deviation. Consequently, the emphasis shifts towards relative differences between values rather than absolute differences, which can prove beneficial in the development of our model. Figure 5 showcases a box plot of the price feature subsequent to the applied transformation. The column exhibits a more normal distribution, with the outliers successfully eliminated.

******

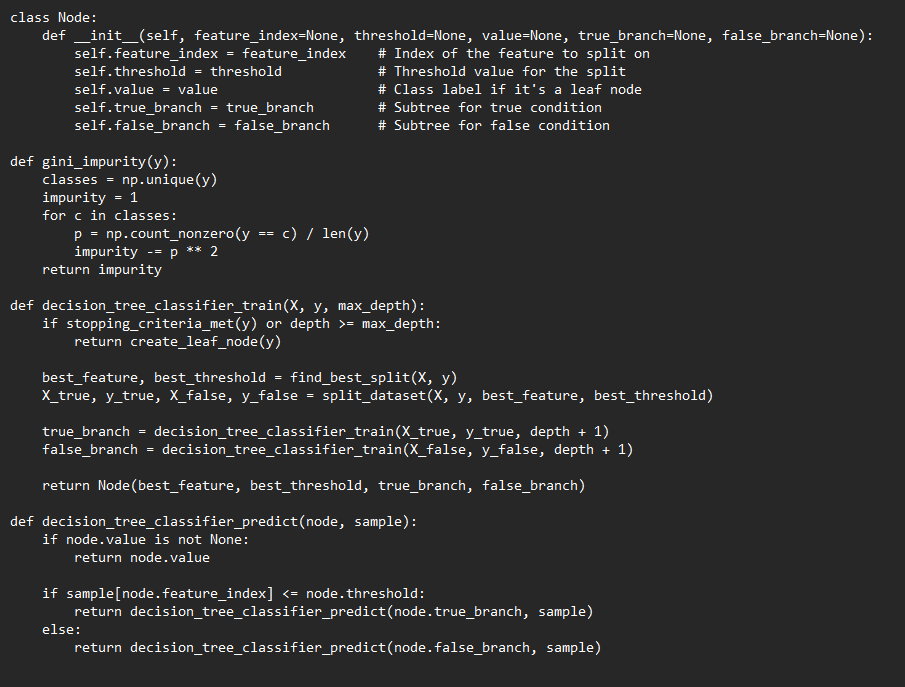
***Figure 5: Price after transformation***

1. **Predictive Models**

**Classification Models**

Logistic Regression is a widely used machine learning method for binary classification tasks, predicting the probability of an observation belonging to a specific category. It utilizes the logistic function to model the relationship between input features and the probability of the outcome. By adjusting model parameters through maximum likelihood estimation, it establishes a decision boundary for class separation. Regularization techniques can be applied to prevent overfitting. Known for its interpretability and ***Figure 6: Logistic Regression Pseudo Code***

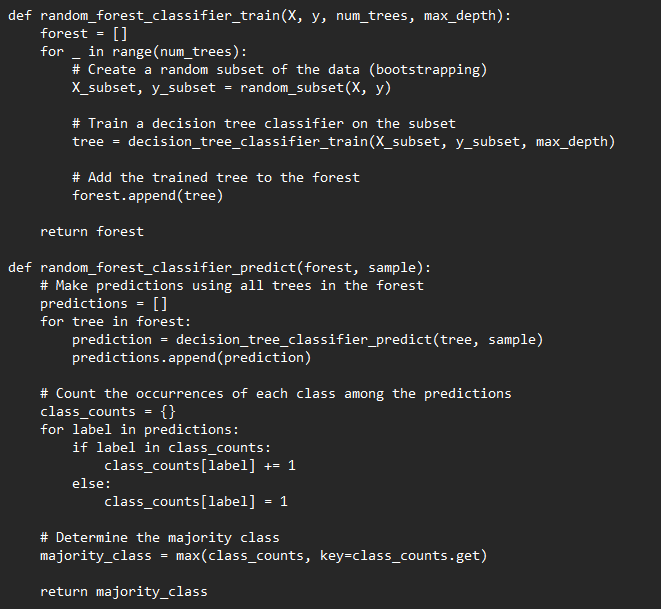
versatility, logistic regression has numerous applications in finance, healthcare, marketing, amongst others. Pseudo code for this type of modeling is shown in Figure 6.



***Figure 7: Decision Tree Classifier Psuedo Code***

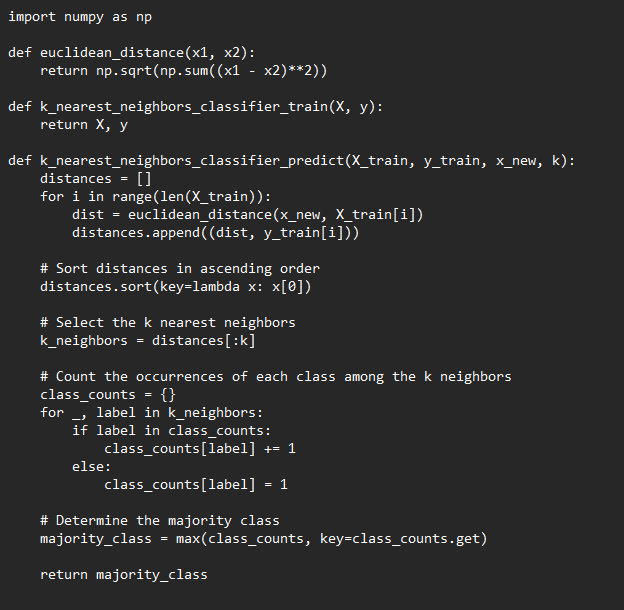
A Decision Tree Classifier is a fundamental machine learning algorithm used for both classification and regression tasks. It works by recursively partitioning the feature space into subsets based on feature values, creating a tree-like structure of decisions. Each internal node represents a feature, each branch represents a decision based on that feature, and each leaf node represents a class label or numerical value. The algorithm selects splits that maximize information gain or minimize impurity at each step. Decision trees are valued for their interpretability and ability to handle complex relationships in data. However, these are prone to overfitting, which can be mitigated using pruning or ensemble methods like Random Forest. Pseudo code for this type of modeling is shown in Figure 7.

A Random Forest Classifier is a machine learning method that builds multiple decision trees during training and combines their predictions for robust classification. Each tree is constructed using a random subset of the training data and a subset of the features, introducing diversity and reducing overfitting. The final prediction is determined through a majority vote or averaging of individual tree predictions. Random Forests excel at handling complex relationships, noisy data, and high-dimensional feature spaces. They provide improved accuracy, reduced overfitting, and valuable insights into feature importance. However, their ensemble nature can make them less interpretable than individual decision trees. Pseudo code for this type of modeling is shown in Figure 8.



***Figure 8: Random Forest Classifier Pseudo Code***

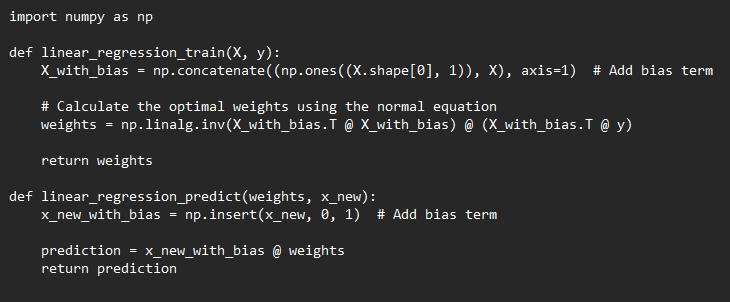
The k-Nearest Neighbors (k-NN) algorithm is a non-parametric and instance-based classification method used for pattern recognition and regression. In k-NN, an observation's class or value is determined by the majority class or the average of the k nearest data points in the feature space. The distance metric, such as Euclidean distance, plays a crucial role in measuring similarity. k-NN is simple to understand and implement, suitable for both small and large datasets, and flexible in handling multi-class problems. However, it can be sensitive to the choice of k, may struggle with high-dimensional data, and requires careful preprocessing to normalize features for accurate distance calculation. Pseudo code for this type of modeling is shown in Figure 9.



***Figure 9: k-NN Pseudo Code***

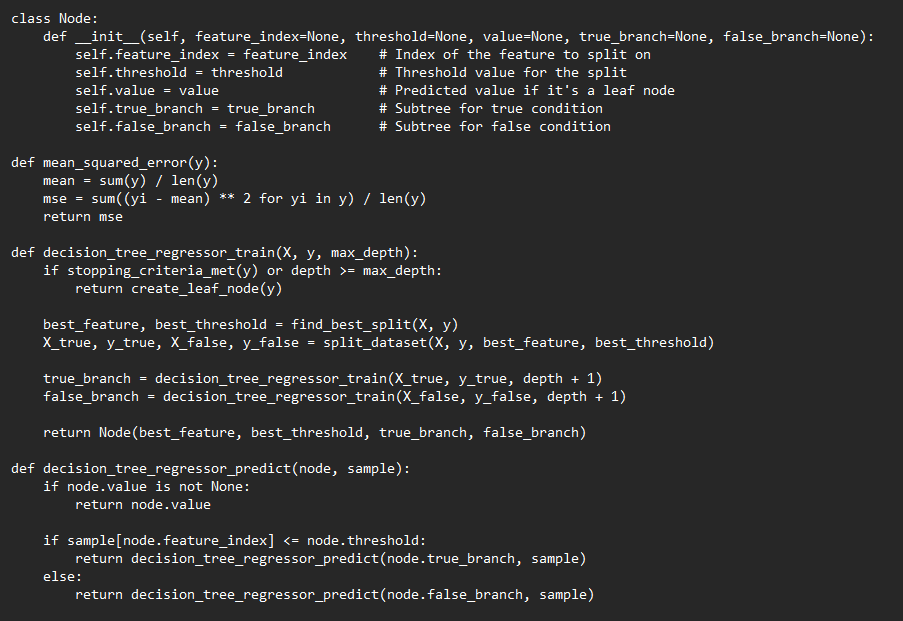
**Regression Models**

Linear Regression is a fundamental and widely-used statistical method employed in machine learning for predicting a continuous numeric outcome. It establishes a linear relationship between input features and the target variable by finding the best-fitting line that minimizes the sum of squared differences between the predicted and actual values. The model's coefficients represent the slope and intercept of this line, providing insights into the feature influences on the target. Linear Regression is interpretable, computationally efficient, and suitable for capturing simple relationships in data. However, it assumes a linear relationship between variables and may not perform well when dealing with complex non-linear patterns. Pseudo code for this type of modeling is shown in Figure 10.



***Figure 10: Linear Regression Pseudo Code***

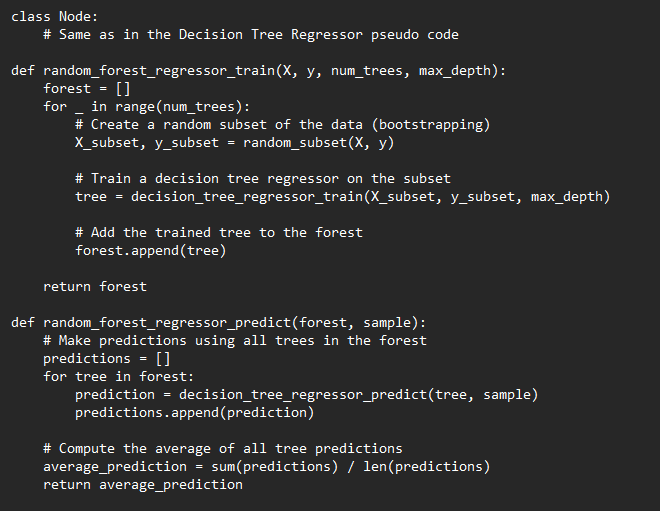
A Decision Tree Regressor is a machine learning algorithm used for regression tasks. It works similarly to the classification version but predicts continuous numeric values instead of categorical labels. The algorithm partitions the feature space into segments based on feature values, creating a tree-like structure of decisions. Each internal node represents a feature, each branch represents a decision based on that feature, and each leaf node represents a predicted numeric value. Decision trees in regression minimize variance by fitting different constant values to each leaf, often the mean of the target values in that region. Decision Tree Regressors are adept at capturing non-linear relationships and handling noisy data. However, they can be sensitive to outliers and overfitting, which can be mitigated using pruning or ensemble methods like Random Forest. Pseudo code for this type of modeling is shown in Figure 11.



***Figure 11: Decision Tree Regressor Pseudo Code***

A Random Forest Regressor is an ensemble learning algorithm designed for regression tasks. It builds multiple decision trees during training and combines their predictions to produce a more accurate and stable final prediction. Each tree is constructed using a random subset of the training data and a subset of the features, introducing diversity and reducing overfitting. The final prediction from the random forest is typically the average of the individual tree predictions. Random Forest Regressors are effective at capturing complex relationships, handling noisy data, and providing insights into feature importance. They tend to have better generalization performance compared to a single decision tree. However, like any ensemble method, Random Forests might be less interpretable than individual decision trees. Pseudo code for this type of modeling is shown in Figure 12.

All four methods for our classification models performed very similarly. We pose that our classification was simply not complex enough to fully utilize the capabilities of the Decision Tree and Random Forest classifier. In this case, if we were to leave our classification unchanged, it would be optimal to utilize a logistic regression for its simplicity and expedited train time.



***Figure 12: Random Forest Regressor Pseudo Code***

1. **Evaluations**

Metrics used for evaluating our machine learning model classification performance are a confusion matrix, accuracy, precision, recall, F1-score, and Receiver Operating Characteristics (ROC) Curve and Area Under the Curve (AUC). A confusion matrix clearly defines true positives, false positives, false negatives, and true negatives. From those, we can obtain the remaining metrics which help quantify the model’s correctness, ability to capture positive instances and exclude negative ones as well as the strengths and weaknesses of the model. ROC-AUC is a good visualization for a model’s trade off between true positive rate and false positive rate.

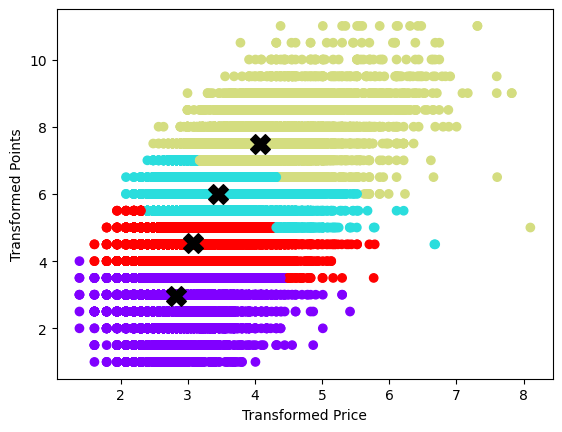
Our ground truth would be checking to see if the wines that are marked as having a score of greater than 88 points were actually scored that way. In this context, we’d like to minimize false positives as much as possible. This is because we’d like to avoid recommending any “bad” wines as “good”; according to our definition of a “good” wine having a score of greater than 88 points.

1. **Modeling**

**Clustering Model**

We used KMeans to build a clustering model on the transformed price and transformed points features. We decided to use only these two features as they are the only numeric features in our dataset. Including categorical features that have been transformed to numerical results in undistinguishable clusters, thus we have chosen to only use the original numeric features, transformed price and transformed points.

We performed the elbow method to determine the optimal number of clusters. We found that four clusters is optimal. Figure 13 shows these four clusters on a graph.

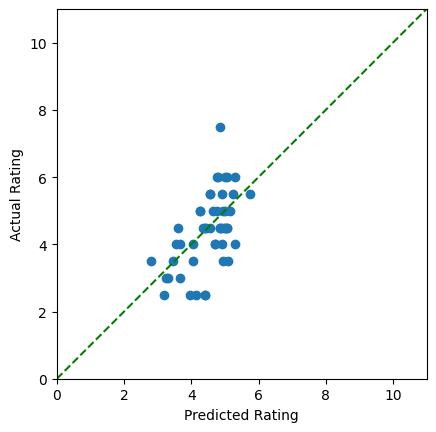


***Figure 13: Clustering Model***

**Cosine Similarity**

Cosine similarity finds a valuable application in building personalized recommendations. Calculating the cosine similarity between a target user’s vector and other users’ vectors, the system identifies users with similar preferences. Similar users serve as neighbors and interactions are aggregated to provide personalized recommendations for items not yet interacted with by the target user. Alternatively, at the item level, cosine similarity helps discover items similar ot those already engaged with by the user, allowing the system to generate suggestions aligned with the individual’s preferences.

We decided to use price, variety, country, and province. These variables provided enough variety in it’s recommendations, limiting bias of popular items, without including any other features that may dominate the calculation. Our cosine similarity model performed relatively well, as seen in Figure 14, depicting predicted ratings against actual ratings. Objectively, our model had an MSE of 0.90.

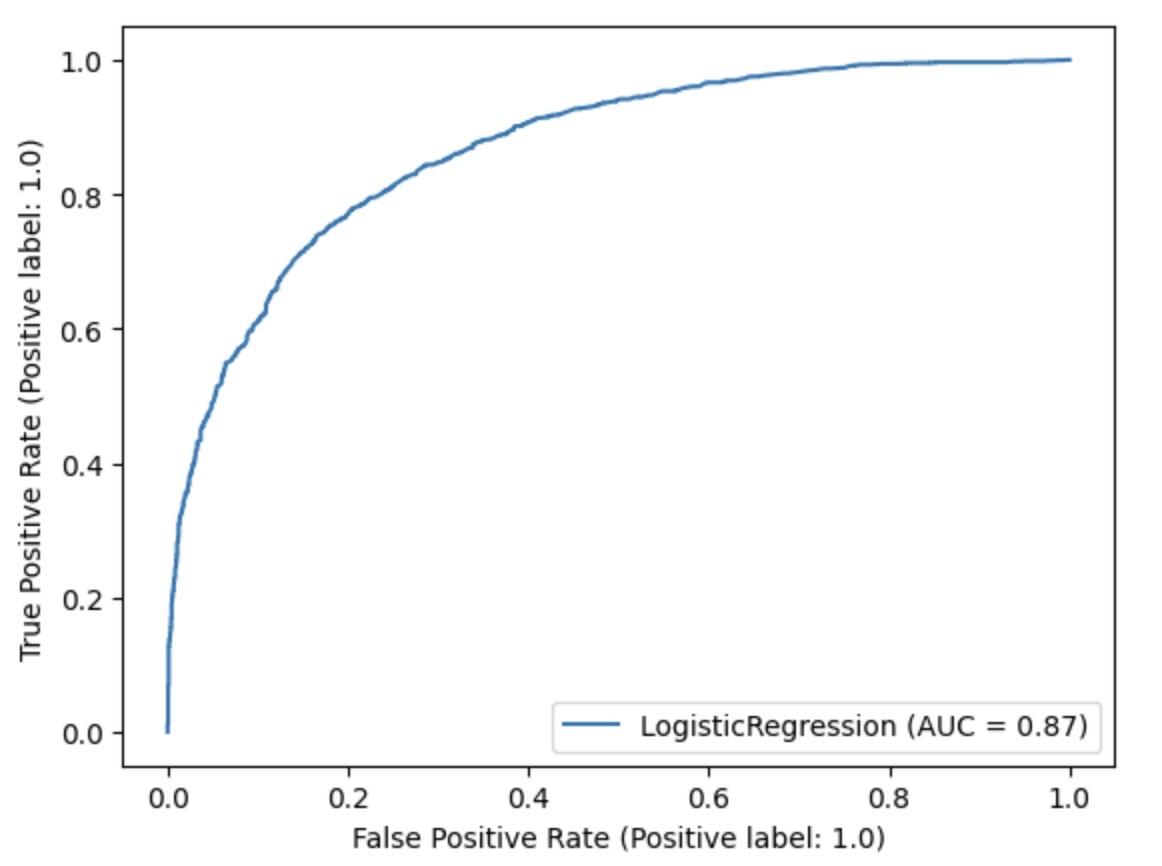


***Figure 14: Cosine Similarity Predictions***

**Classification Models**

We performed four different classification models including: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and K-Nearest Neighbors. In each of these classification models, we encoded the points feature to 0 and 1, where 1 represents points greater than or equal to 88, and 0 represents points less than 99. We filtered our dataset on our target user of ‘Roger Voss’. Our X variable, or user input, is variety and location, both of which have been one hot encoded, and price which has been log transformed. Our Y variable, or our target, is outputted as a 1 or a 0, 1 representing ‘Yes, recommend’, and 0 representing ‘No, do not recommend’. We performed an 80/20 train/test for Roger Voss’ 18,068 reviews and used these train and test sets for our four different classification models to compare results.

In the logistic regression, we performed no hyperparameter tuning and our model resulted in an accuracy of 0.78, a precision of 0.83, a specificity of 0.79, and an F1-score of 0.8. The ROC curve of this model is shown in Figure 15, and the area under the curve (AUC) of this model is 0.87. The confusion matrix of this model is shown in Table 2.



***Figure 15: Logistic Regression ROC Curve***

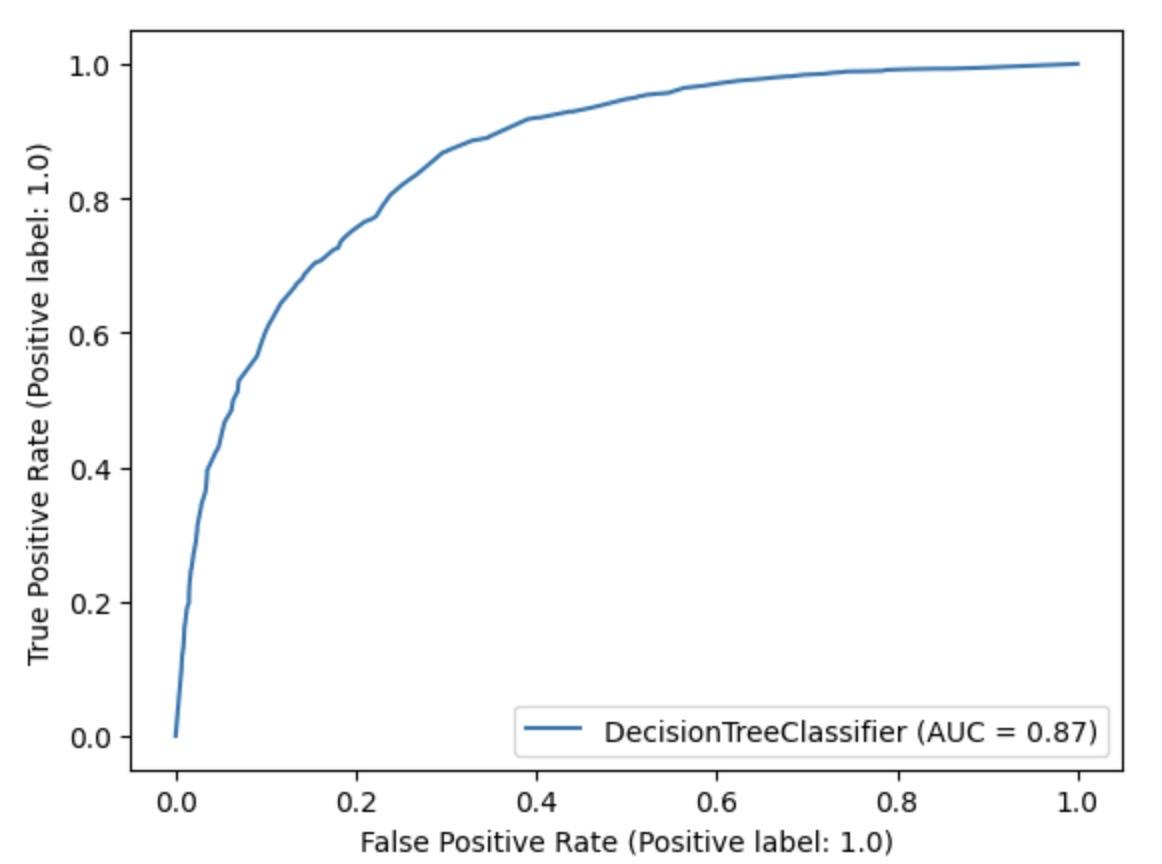
Logistic Regression is a widely used classification algorithm that models the probability of a binary outcome. In our case, the binary outcome indicates whether a user would enjoy a wine or not. The logistic regression model learns the relationships between the input features (such as price, variety, and location) and the binary outcome (enjoyment prediction) by finding the best-fitting S-shaped logistic curve.The logistic regression model showed a consistent and balanced performance across various evaluation metrics. It accurately predicted whether a user would enjoy a wine with reasonable accuracy, precision, specificity, and an F1-score.

|  | Actual Positive | Actual Negative |
| --- | --- | --- |
| Predicted Positive | 1529 | 306 |
| Predicted Negative | 448 | 1214 |

***Table 2: Logistic Regression Confusion Matrix***

In the decision tree classifier model our hyperparameter tuning consisted of a criterion of entropy, a max\_depth of 30, and min\_samples\_split of 200. Our model resulted in an accuracy of 0.78, a precision of 0.82, a specificity of 0.78, and an F1-score of 0.79. The ROC curve of this model is shown in Figure 16, and the AUC of this model is 0.87. The confusion matrix of this model is shown in Table 3.

The Decision Tree Classifier is a non-linear algorithm that splits the data into subsets based on the values of input features. It creates a tree-like structure where each internal node represents a feature, each branch represents a decision based on that feature, and each leaf node represents a class label (in this case, wine enjoyment).The decision tree classifier achieved similar results to the logistic regression model. It effectively captured decision boundaries within the feature space, enabling it to provide reliable predictions. Its accuracy, precision, specificity, and F1-score were all in line with the logistic regression model.



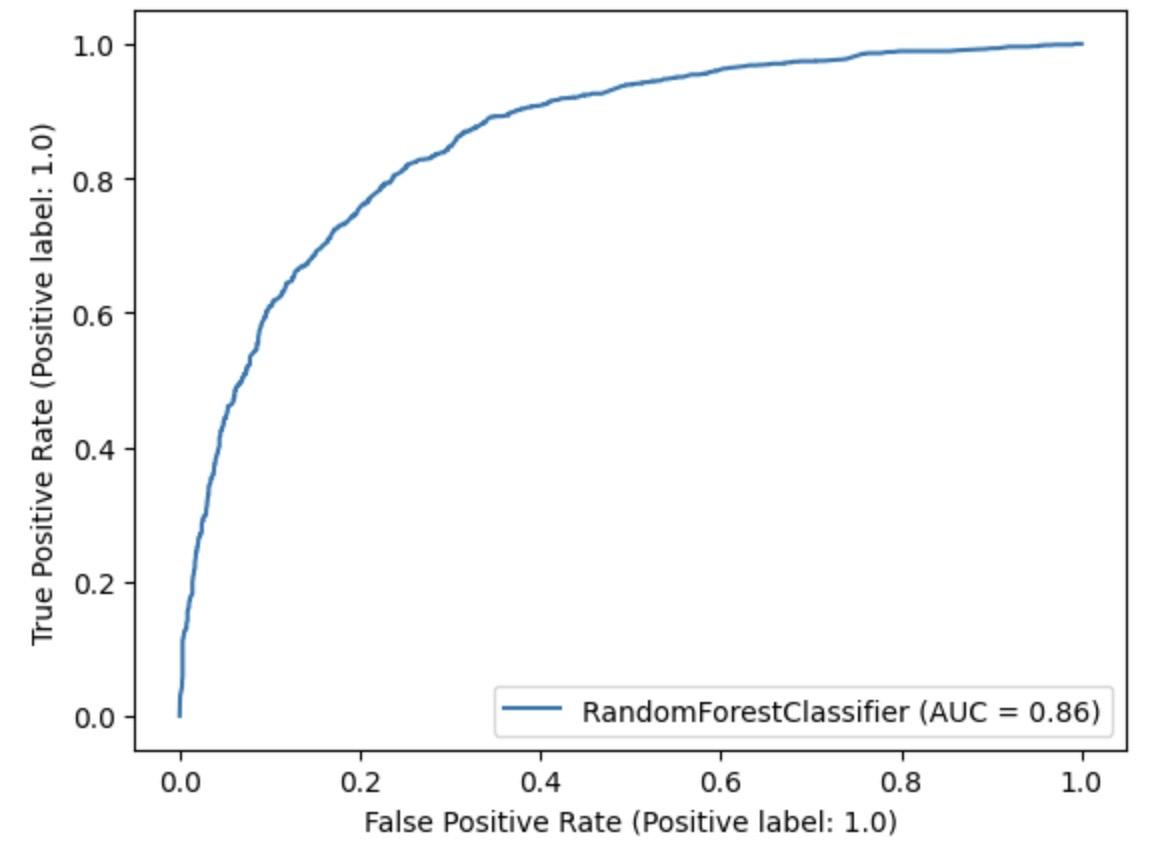
***Figure 16: Decision Tree Classifier ROC Curve***

|  | Actual Positive | Actual Negative |
| --- | --- | --- |
| Predicted Positive | 1498 | 337 |
| Predicted Negative | 438 | 1224 |

***Table 3: Decision Tree Classifier Confusion Matrix***

In the random forest classifier model our hyperparameter tuning consisted of a criterion of entropy, a max\_depth of 25, min\_samples\_split of 75, and n\_estimators of 200. This model resulted in an accuracy of 0.78, a precision of 0.83, a specificity of 0.79, and an F1-score of 0.80. The ROC curve of this model is shown in Figure 17, and the AUC of this model is 0.86. The confusion matrix of this model is shown in Table 4.

The Random Forest Classifier is an ensemble method that creates multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. Each tree is built on a random subset of the data and a random subset of features. The random forest classifier delivered results consistent with the other models. By aggregating predictions from multiple decision trees, it provided robust and reliable predictions, as reflected in its accuracy, precision, specificity, and F1-score.



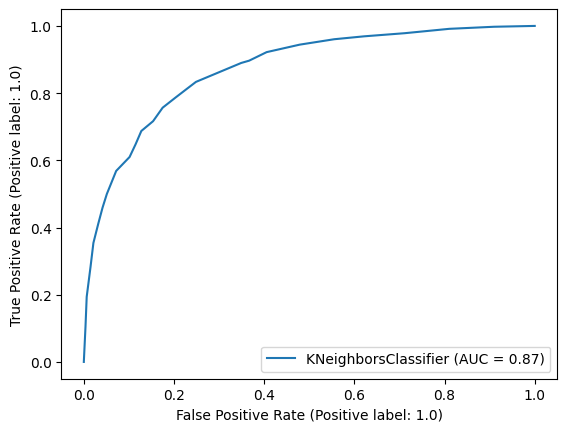
***Figure 17: Random Forest Classifier ROC Curve***

|  | Actual Positive | Actual Negative |
| --- | --- | --- |
| Predicted Positive | 1514 | 321 |
| Predicted Negative | 457 | 1205 |

***Table 4: Random Forest Classifier Confusion Matrix***

In the final classification model we performed, K-Nearest Neighbors, the hyperparameter tuning consisted of n\_neighbors of 27. This model resulted in an accuracy of 0.79, a precision of 0.81, a specificity of 0.77, and an F1-score of 0.80. The ROC curve of this model is shown in Figure 18, and the AUC of this model is 0.87. The confusion matrix of this model is shown in Table 5.

K-Nearest Neighbors (KNN) is a simple yet effective algorithm that classifies data points based on the class labels of their K nearest neighbors. The prediction is made by majority voting among the K neighbors.The K-Nearest Neighbors model displayed competitive performance in predicting wine enjoyment. It leveraged the similarities between data points to make predictions, and its accuracy, precision, specificity, and F1-score were on par with the other models.



***Figure 18: K-Nearest Neighbors ROC Curve***

|  | Actual Positive | Actual Negative |
| --- | --- | --- |
| Predicted Positive | 1500 | 382 |
| Predicted Negative | 345 | 1270 |

***Table 5: K-Nearest Neighbors Confusion Matrix***

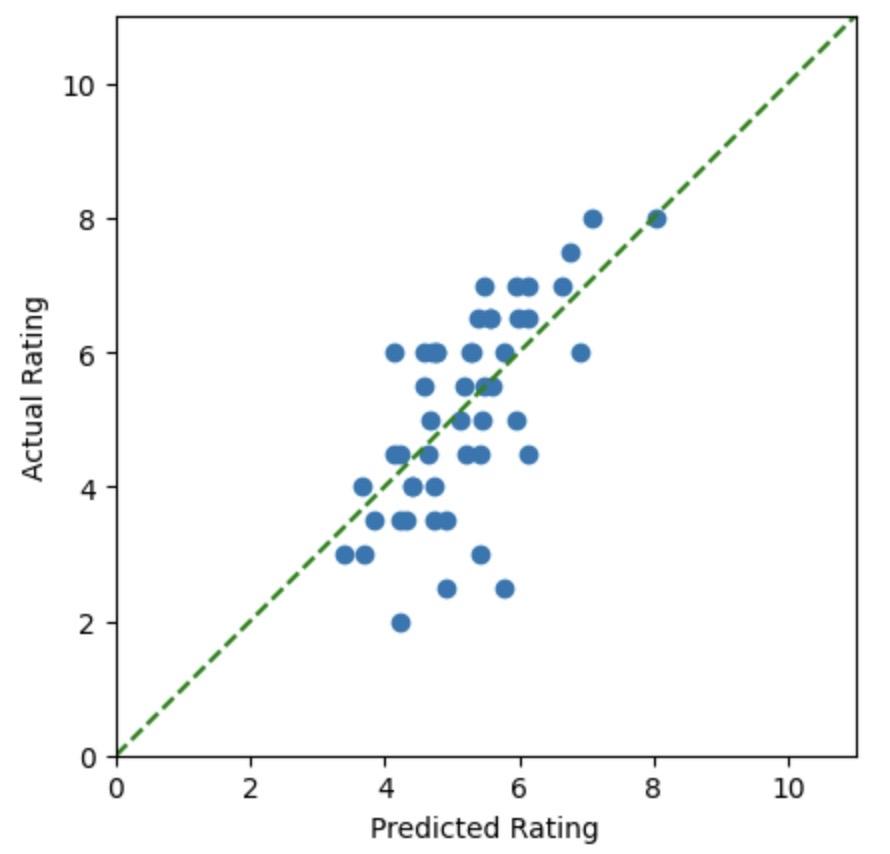
**Regression Models**

The regression models we have performed are similar to the classification models, except here, we are predicting a numeric rating that would be made for all wines instead of just a Yes/No recommendation. The regression models we have performed include: Linear Regression, Decision Tree Regressor, and Random Forest Regressor.

The train and test sets in these regression models are the same train and test sets as the classification models with the same input content variables (price, variety, and location) to predict the numeric target feature of points. In these regression models, a numeric prediction of the points feature is made for all un-reviewed wines in the database. Recommendations are then provided based on the highest predicted points by the model.

In the linear regression model, we did not perform any hyperparameter tuning. This model resulted in an R-squared value of 0.56 and a root mean square error (RMSE) value of 1.03. Figure 19 shows a scatter plot of the results of this model.

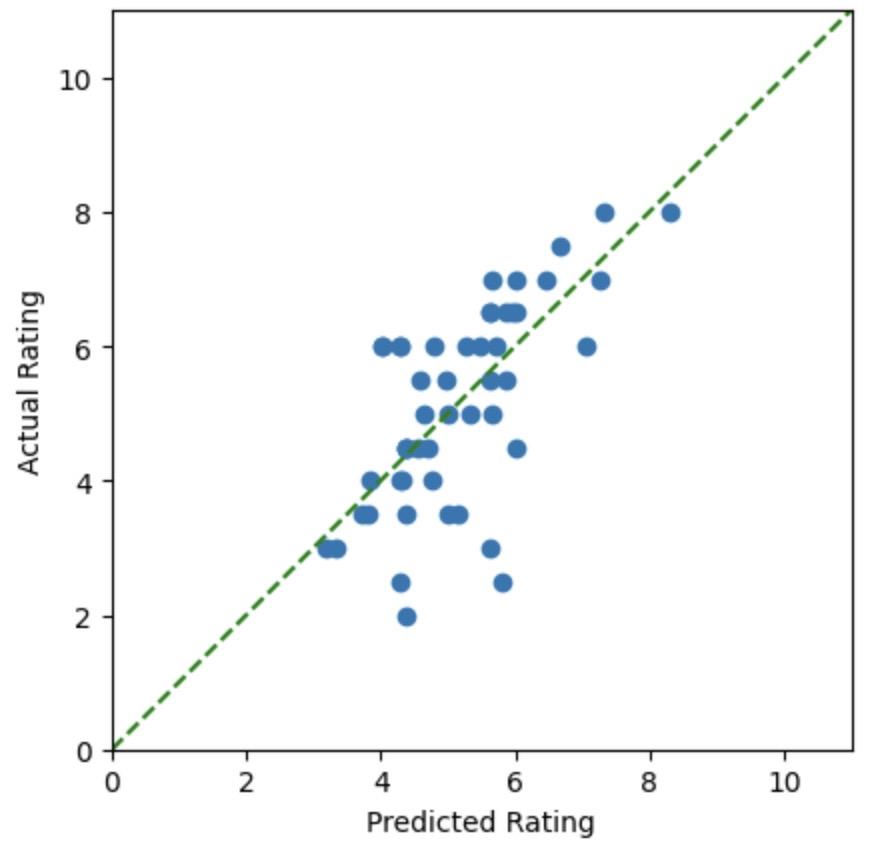
Overall, the linear regression model did not perform as well as we had hoped. The model provided insights into predicting wine ratings, but it had its limitations. An R-squared value of 0.56 indicates that approximately 56% of the variance in wine ratings was captured by the model. An RMSE of 1.03 suggests the average prediction error of the model. While it offers a basic understanding of how input features affect wine ratings, it might not capture the complexities of the data fully.



***Figure 19: Linear Regression Model***

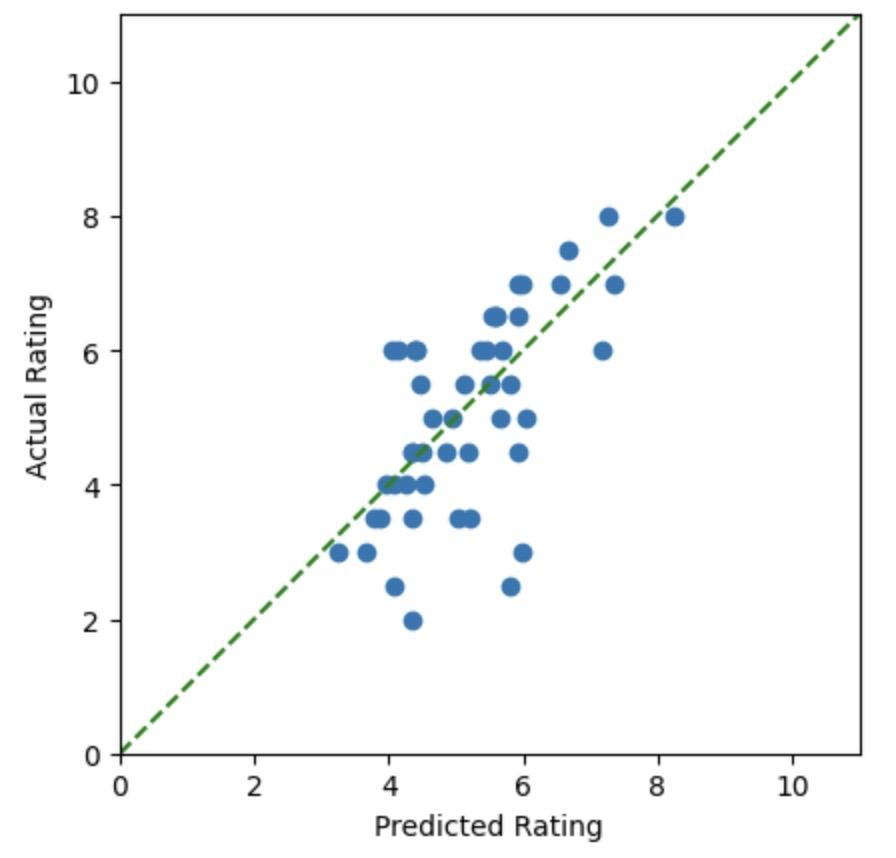
In the decision tree regressor model, we performed hyperparameter tuning consisting of a max\_depth of 30 and a min\_samples\_split of 175. This model resulted in an R-squared value of 0.55 and an RMSE value of 1.03. Figure 20 shows a scatter plot of the results of this model.

The decision tree regressor model performed similarly to the linear regression model. The decision tree regressor exhibited similar limitations to the linear regression model. An R-squared value of 0.55 and an RMSE of 1.03 indicate performance comparable to linear regression. While the decision tree model might capture non-linear relationships between features and ratings better than linear regression, it might still miss some nuances in the data.



***Figure 20: Decision Tree Regressor Model***

In the random forest regressor model, we performed hyperparameter tuning consisting of n\_estimators of 200, max\_depth of 25, and min\_samples\_split of 50. This model resulted in an R-squared value of 0.58 and an RMSE value of 1.0. Figure 21 shows a scatter plot of the results of this model.

The random forest regressor showed slight improvements compared to the linear regression and decision tree regressor models. An R-squared value of 0.58 and an RMSE of 1.0 indicate better performance in capturing the variance in wine ratings and reducing prediction errors. The ensemble nature of random forests allows it to capture complex relationships better than individual decision trees.

***Figure 21: Random Forest Regressor Model***

1. **Results and Discussion**

In this project, we started with a dataset acquired from Kaggle, which consisted of nearly 120,000 unique wine reviews provided by wine sommeliers. The dataset encompassed 13 attributes, including information about the country, description, designation, points, price, province, region, reviewer details, title, variety, and winery of each wine.

During the exploratory data analysis, we observed that the dataset was dominated by wines from the United States, accounting for approximately 42% of the entire dataset, followed by France and Italy. The wine descriptions and designations were rich and varied, providing valuable insights into the qualities and characteristics of each wine. The points attribute, which represented the enjoyment score, showed a fairly normal distribution with a slight right skew. The price attribute, on the other hand, exhibited a significant right skew, indicating a wide range of prices for the wines in the dataset.

To address missing values, we decided to remove rows with missing values in the price, country, province, and variety columns. This allowed us to retain over 110,000 rows for analysis, ensuring a substantial dataset for our Wine Recommender System.

Regarding outliers, we focused primarily on the price feature. We employed a logarithmic transformation on the price column, which reduced all values, eliminated outliers, adjusted skewness, equalized variance, and lowered the standard deviation. This transformation resulted in a more normalized distribution of prices and improved the overall statistical outcomes of the column.

We performed numerous predictive modeling techniques and determined that the classification models performed the best. In terms of performance, all four classification models demonstrated similar results. They all provided accurate and balanced predictions, indicating their effectiveness in distinguishing between wines that users would enjoy and those they might not. The key takeaway is that any of these models could be a suitable choice for your Wine Recommender System, as they consistently delivered reliable recommendations. When choosing among these models, we might consider factors beyond performance. Logistic Regression is straightforward and interpretable, making it suitable for explaining the reasons behind predictions. Decision Trees provide insight into decision boundaries, while Random Forests reduce overfitting. K-Nearest Neighbors leverages similarity among data points. Ultimately, our choice could depend on factors such as interpretability, computational efficiency, and the ease of integrating the model into your recommender system's architecture. In summary, each model brings its own strengths to the table, and based on their similar performance, you have flexibility in selecting a model that aligns with your system's requirements and design considerations.

1. **Conclusion**
   1. **Summary / Recommendations**

In conclusion, our Wine Recommender System, built upon the pre-processed dataset, has the potential to assist a wide range of users, from novice wine consumers to well-versed wine sommeliers. For novices, the system offers personalized recommendations based on their preferences, allowing them to discover new wines, cultivate their palate, and gain a better understanding of the complexities of wine. For sommeliers, the system serves as a valuable tool for expanding their wine repertoire, exploring new and obscure wines, and delivering exceptional experiences to their clientele.

Overall, the Wine Recommender System that will be built in this project has the potential to revolutionize the way individuals discover and select wines, catering to both novices and experts alike, and fostering a deeper appreciation and enjoyment of this popular beverage.

* 1. **Future Work**

The insights gained from the dataset, coupled with the data preprocessing techniques applied, have laid the foundation for the development of an effective Wine Recommender System. Future work could involve implementing and evaluating different recommendation algorithms, such as content-based and knowledge-based approaches, to provide more accurate and tailored wine recommendations. Additionally, integrating collaborative filtering techniques could further enhance the system's ability to suggest wines based on the preferences of similar users.

1. **References**
2. Zackthoutt. (2017, November 27). Wine reviews. Kaggle. https://www.kaggle.com/datasets/zynicide/wine-reviews
3. About Us. Wine Enthusiast. (2021, May 5). https://www.winemag.com/about-us/
4. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.
5. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.
7. Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). Classification and regression trees. CRC press.
8. Scikit-learn: Machine learning in Python. (<https://scikit-learn.org/stable/modules/tree.html>)
9. Scikit-learn: Machine learning in Python. (https://scikit-learn.org/stable/modules/ensemble.html#random-forests)
10. Scikit-learn: Machine learning in Python. (<https://scikit-learn.org/stable/modules/neighbors.html>)
11. Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to Linear Regression Analysis. John Wiley & Sons.
12. Scikit-learn: Machine learning in Python. (<https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares>)
13. Scikit-learn: Machine learning in Python. (https://scikit-learn.org/stable/modules/tree.html#regression)
14. Scikit-learn: Machine learning in Python. (<https://scikit-learn.org/stable/modules/ensemble.html#random-forests>)