**Name for each team member:** Stacia Zembal, Matthew Lipinski, Khatoon Alghanim, Chetan Vasudevan, Mariyam Rajeev, Khulan Gankhuyag, Gauthamreddy Manemvenkat

**Text Mining and Predictive Models – Stacia Zembal & Matthew Lipinski**

**Algorithm**

Initially we had hoped to produce a predictive model based on the variety of wine from the dataset. We found that this was going to be a difficult task given the 632 different varieties of wine in the dataset. We then chose to come up with an aggregated model for red and white wines based on the descriptions of each record.

We first ran a function to find the frequent words in all descriptions and then extracted the top 10 red and white words from each description. We then assigned a basic scoring model of the frequency of the red words and white words for each description. For any record with more red words than white, we assigned the record with a red value and the same method holds true for white. We then used kmeans to cluster these values into a single column (utilizing *k*means for this was not necessary as we could have used simple logic, but was helpful in testing other *k* values). This provided us with a single column defining each record as either red or white based upon the frequency of words that correlated to that particular description.

Next, we created a single decision model that utilized the cluster (red/white), points, price and region that has an accuracy of 58%. This is a rather poor result in deciding whether a wine is red or white. Our next attempt was to see if other models produced better results given our data. Not really to any surprise, Random Forest and Naïve Bayes produced a 59% accuracy.

This lead us to thinking our scoring model was the issue and subsequently the words we subjectively had chosen for our red and white extraction on description. We then tried choosing a different set of red and white words for our models. By removing the word blend and replacing with cabernet for the red group and removing oak and fruity and replacing with citrus and green respectively. Next, we ran the same decision tree model which produced results of 66%.

Initial Red & White Word Matching

wm\_red<-c("cherry", "black", "plum", "berry", "red", "blackberry", "tannins", "blend", "dark", "pepper")

wm\_white <- c("ripe", "fresh", "oak", "apple", "white", "peach", "light", "fruity", "bright", "vanilla")

Take 2 Red & White Word Matching

wm\_redT2 <- c("cherry", "black", "plum", "berry", "red", "blackberry", "tannins", "cabernet", "dark", "pepper")

wm\_whiteT2 <- c("ripe", "fresh", "citrus", "apple", "white", "peach", "light", "bright", "vanilla", "green")

Our findings lead us back to the preprocessing. We needed to have a more sophisticated scoring model for our red and white categories to distinguish between red and white models based upon the descriptions. We likely need to implore a weighted model where red or white carries a higher weight than other words.

**Tools Selection**

We utilized RStudio for preprocessing, text analysis, predictive models. We chose RStudio, because it is a tool that we both had the most experience with. All of our programs developed are located in the folder labeled "Small Team's Work" -> "Zembal, Stacia & Lipinski, Matthew".

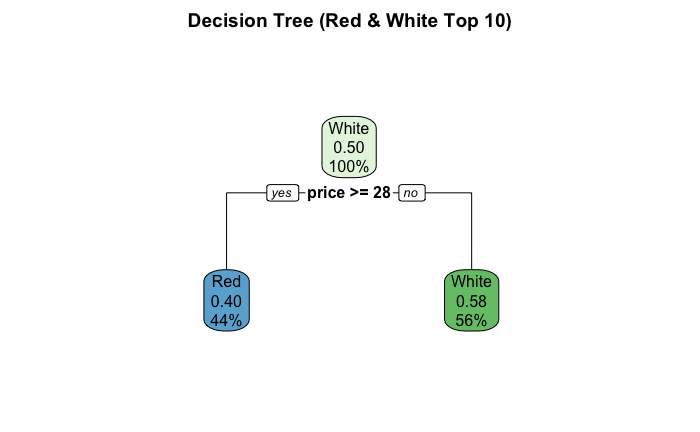
* "WineMagDataClean" - Initial preprocessing of dataset from 150K records to 97K.
* "Stacia\_Matt\_TextMining" - Text mining on preprocessed dataset "Description" column.
* "ModelsWineMagMatthew" - Models created on the red and white wines from text mining.

**Data Mining Results**

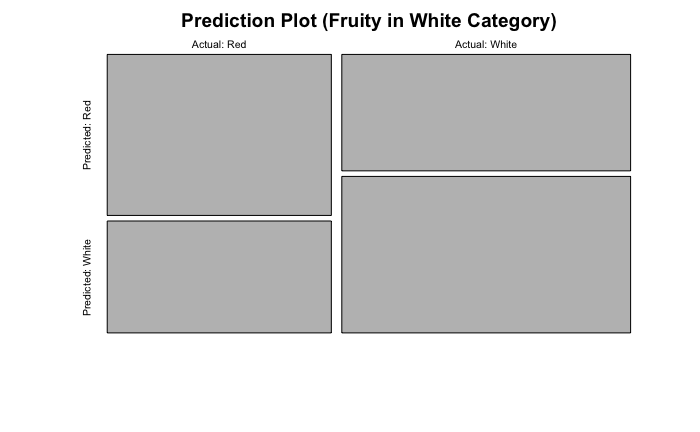
Below is a subset of the initial table that we were used to view the frequency of variety of wine and the count of individual records defined as either red and white based on the scoring method for red and white wines. More noticeable after reflecting on how the model resulted, but we can derive the scoring method for categorizing a wine based on the word frequency was not distinct enough. For example, Cabernet Sauvignon was categorized as a red wine 6,292 times, but also categorized as a white wine 1,980 times. Thus, making it very difficult for our model to correctly predict which wine color (red or white) based on this data.

|  |  |  |
| --- | --- | --- |
| Variety of Wine | Red Categorization | White Categorization |
| Albarino | 4 | 361 |
| Cabernet Sauvignon | 6,292 | 1,980 |
| Chardonnay | 337 | 8,826 |
| Malbec | 1,522 | 446 |
| Meritage | 164 | 38 |
| Pinot Noir | 5,846 | 3,437 |

Next, we have a few outputs from our decision tree model based on the Initial Red & White Word Matching. We can see that the decision tree only utilized the price as a variable used in tree construction and disregarded the points. In attempts to tune the model utilizing rpart.control, making the tree deeper didn’t increase the accuracy in prediction. Which, we now know is because of our subjective choice of red and white words.

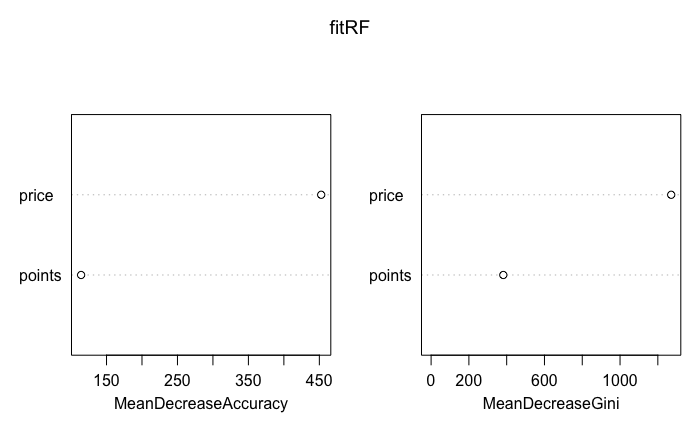


The confusion matrix displayed below shows how model has a difficult time predicting the type of wine (red or white) based upon the predictors price and points.

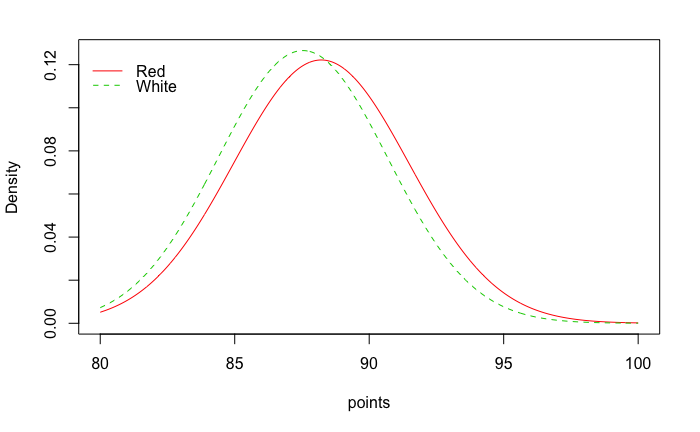
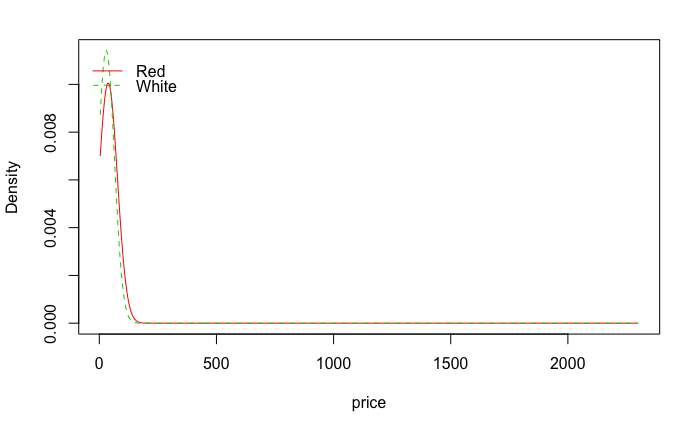


|  |  |  |
| --- | --- | --- |
|  | Predicted Red | Predicted White |
| Actual Red | 6,892 | 4,790 |
| Actual White | 6,422 | 8,632 |

Moving onto the Random Forest, we can see that the Gini placed the price as the most significant predictor. In the end, the Random Forest produced a nearly identical confusion matrix. Again, we know this is because red and white scoring model we implemented.



We also tried Naïve Bayes to see if decision trees were a poor model, but again the model produced really low prediction accuracy of 59%. Below are the outputs of our Naïve Bayes model.



|  |  |  |
| --- | --- | --- |
|  | Predicted Red | Predicted White |
| Actual Red | 5,997 | 3,530 |
| Actual White | 7,317 | 9,892 |

After reviewing the outputs of our models on the “Initial Red and White Word Matching”, we had good confidence the reason our models couldn’t perform was due to our scoring method of red and white wines. We had either incorrectly chosen words that didn’t associate with red and white correctly or didn’t have a rigorous enough scoring method utilizing those words. With this in mind to see if our theory was correct, we tried changing some of the words in our red and white matching categories. We then created “Take 2 Red and White Word Matching”. We created a decision tree utilizing the same training and test values and produced a model that has a 66% accuracy.

Perhaps we were too focused on utilizing the work we had done on our text mining and analysis and should have focused on aggregating other attributes, creating new features, or moving forward with the attributes we started with initially. However, we wanted to display the work and knowledge we gained from our text mining and analysis. We definitely learned a lot from working on this project despite our models’ unfavorable results.

**Comparison of Algorithms**

As we discussed in detail in our “Data Mining Results” section, our models had consistent accuracy across the decision tree, random forest, and Naïve Bayes. We now know that our scoring method for determining the wine’s color based on the description was our downfall. We proved this by creating a second red and white word matching categories that produced a better accuracy. However, we also know that our text mining skills have much room for improvement as this was both of our first attempts at such an analysis.

We found this dataset on Kaggle and one user did a similar analysis with text. The one thing they did differently is they were more specific to the variety of wine. For example, they focused on a specific variety of wine e.g. Pinot Noir and Cabernet Sauvignon. If we were to go back and also get more specific about the type of wine, we most likely could have produced better models that focused on the frequency of words used to see if the points and price were correlated to the description.

Another use developed a function to determine if the descriptions were similar to one another using cosine similarity. This is a rather novel investigation. From her model, you provide a description and from that description it matches to other like descriptions. Utilizing this model could then be a recommendation engine based upon a description. You would find other wines that meet your description and then try that wine to see if it is something that you enjoy.

Going into this project, we set the bar very high and probably attempted too much initially. Reflecting on what we did over the semester now, we should have focused our efforts on smaller accomplishments and built on top of those. Attempting to build a model to predict the variety of wine, 632 varieties, is something no one has accomplished on Kaggle yet. We know that a more rigorous scoring model for red and white wines would improve our models, but we also should have focused on subsets of varieties. We could create an additional feature that aggregates the variety of wine into more categories than red and white and would potentially make a more decisive and faster processing.

In closing, we are disappointed with the outcome of our models, but we are able to take a lot of lessons learned away from this project. We can safely say this project pushed our boundaries, felt we learned the most from our project work, and inspired us to get involved with more Kaggle competitions.

**CART & Bagging Using Python: Rajeev George, Mariyam & Vasudevan, Chetan**

**Algorithm**

Utilized CART and Bagging for points that was binned into two categories less than or greater than 90 points.

**Tools Selection**

The tool selected was Python. All of the programs developed are located in the folder labeled "Small Team's Work" -> "Rajeev George, Mariyam & Vasudevan, Chetan".

**Data Mining Results**

Results produced the following:

|  |  |
| --- | --- |
| AUC Type | Result |
| Baseline AUC | 55.53% |
| Pruned AUC | 59.00% |
| Bagged AUC | 64.21% |

**Comparison of Algorithms**

The Bagged method provided the greatest area under the curve.

**Decision Tree: Gankhuyag, Khulan & Alghanim, Khatoon**

**Algorithm**

Created a decision tree based on all attributes except for description and designation. Target was price which was binned into two prices consisting of equal to and greater than $50 and less than $50.

After cleaning the data, we remove two columns "Description and Designation" by using RStudio. Then, we use the dataset to predict using 70% of the sample and 30% for the testing. We set target as price and other attribute as parameter. After that, we convert the price to binary by splitting the data: if the price => 50 then it is 1. otherwise it is 0.

**Tools Selection**

The tool selected was RStudio. All of the programs developed are located in the folder labeled "Small Team's Work" -> "Gankhuyag, Khulan & Alghanim, Khatoon".

**Data Mining Results**

We couldn't get the Confusion Matrix and ROC curve for the decision Tree because we had problem with the DT and we were out of time.

**Comparison of Algorithms**

Unknown.

**Linear Regression: Manemvenkat, Gauthamreddy**

**Algorithm**

Created a linear regression based upon price and points.

**Tools Selection**

The tool selected was Python. The program developed are located in the folder labeled "Small Team's Work" -> "Manemvenkat, Gauthamreddy".

**Data Mining Results**

Accuracy of 75.09%.

**Comparison of Algorithms**

Unknown.