Capstone Proposal and Overview

My Capstone project will analyze genetic variant classifications between laboratories and seek to determine patterns in measured data that lead to agreements or disagreements in said classifications.

99.5% of all DNA is shared across all humans; it is the 0.5% that makes all the difference. Genetic variations, or variants, are the differences that make each person's genome unique. A genome is the entire set of genetic material for an organism. The human genome consists of about 3 billion base pairs of DNA across 23 pairs of chromosomes. DNA sequencing identifies an individual's variants by comparing the DNA sequence of an individual to the DNA sequence of a reference genome maintained by the Genome Reference Consortium (GRC).

The average person's genome has millions of variants. Variants occur mostly in DNA sequences outside of genes. Some variants contribute to the differences between humans, such as different eye colors and blood types; other variants have been linked with diseases, but most variants currently have unknown effects. As more DNA sequence information becomes available to the research community, the effects of some variants may be better understood.

The genetics community has a defined structure to categorize genetics variants.

- Pathogenic - a sequence variant that is previously reported and is a recognized cause of the disorder.
- Likely Pathogenic – a sequence variant that is previously unreported and is of the type which is expected to cause the disorder.
- VUS (Variant of Unknown Significance) – a sequence variant that is previously unreported and is of the type which may or may not be causative of the disorder.
- Likely Benign – a sequence variant that is previously unreported and is probably not causative of disease.
- Benign – a sequence variant is previously reported and is a recognized neutral variant.

The data I am working with is provided by three clinical labs and collected and curated by ClinVar. From their website, "ClinVar is a freely accessible, public archive of reports of the relationships among human variations and phenotypes, with supporting evidence. ClinVar thus facilitates access to and communication about the relationships asserted between human variation and observed health status, and the history of that interpretation. ClinVar processes submissions reporting variants found in patient samples, assertions made regarding their clinical significance, information about the submitter, and other supporting data. The alleles described in submissions are mapped to reference sequences, and reported according to the HGVS standard. ClinVar then presents the data for interactive users as well as those wishing to use ClinVar in daily workflows and other local applications. ClinVar works in collaboration with interested organizations to meet the needs of the medical genetics community as efficiently and effectively as possible."

The dataset I am using is found on Kaggle as "Genetic Variant Classifications". This data set contains 65188 observations each with 45 variables measured per record and 1 label variable (this analysis' target variable) 'CLASS'. The 'CLASS' variable is a binary representation of the outcome of lab agreement or conflict. An agreement is encoded as 0 and a conflict is encoded as 1. Each record represents an allele- a variant form of a given gene. These variants are (usually manually) classified by clinical laboratories using the previously defined categorical spectrum. Variants that have conflicting classifications (from laboratory to laboratory) can cause confusion when clinicians or researchers try to interpret whether the variant has an impact on the impact in a given patient.

There is an additional variable known as the 'CLASS' feature, which indicates whether or not all of the participating labs agreed on their variant classification type. Therefore the focus of this analysis will be on the binary classification, while examining relative variables that may be linked to each classification case.

The binary variable of interest, 'CLASS' will be analyzed across several feature variables (columns) to determine patterns and relationships between classifications and other measured quantities. Both binary cases- classification agreements and disagreements- will be analyzed.

The application of this analysis aims to benefit ClinVar directly as well as the broader genetics community and its understandings of genetic variant classifications. The primary question I seek to answer is- which feature or features have a significant correlation to the target variable (the classification), and how well do various ML models perform in using them to predict the classification variable on unseen data. A deeper dive into this capstone may look into the specific values of these features to determine if there exists an even deeper link to the classification agreements/disagreements based upon specific values.

Data Wrangling and Processing

The dataset I am using for my capstone project is from Kaggle and therefore relatively clean already. However, there was still quite a bit of data wrangling and processing to be done. The overarching goal for the wrangling and processing is to filter out the features that are going to be unhelpful in the future analysis.

An info call on the dataframe reveals that there are 65188 total records with a total of 46 columns (variables) measured. In a data science sense, there are 45 feature or predictor variables and 1 target variable- the 'CLASS' variable. Figure 1 shows the breakdown of this target variable.

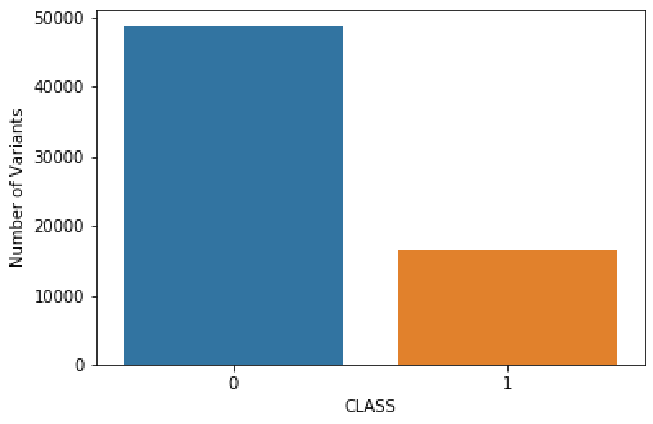


Figure 1. Breakdown of lab agreements (0) and disagreements (1). There are approximately three times more agreements than disagreements

Looking into the dataframe, I immediately see that there are several null entries between the feature variables. Furthermore, the number of null entries between all features is inconsistent (meaning some features have more or less null entries than others). A visual representation of this observation can be seen below in Figure 2, where yellow marks a null entry in a given feature variable. Notice how some feature variables have a solid yellow bar-indicating that it contains *mostly* null entries.

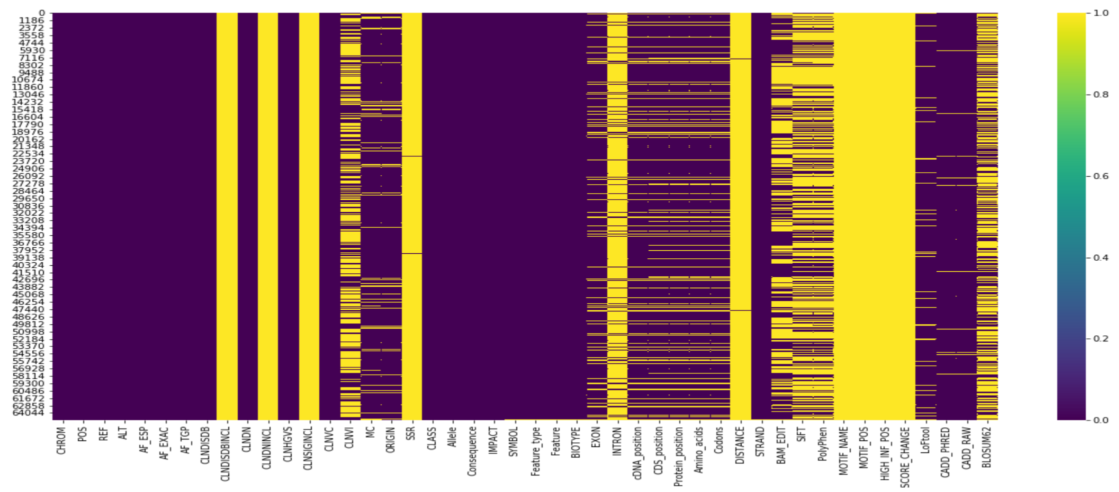


Figure 2. Visual of the null values in every feature variable. Ten of the 45 feature variables are nearly completely filled with null entries, and even more variables mostly contain null values.

As in any data science endeavor, the most comprehensive analysis includes every feature variable and every entry. Unfortunately, this result is not always achievable. For instance, there are several feature variables that have less than 90% or even 80% of the maximum non-null entries, and more that have even less. Inspecting these variables that lack all of the entries, I find that there is no immediately obvious way to intelligently fill all or even most of these missing entries. There are simply too many unique values to safely fill missing entries. This brings about another issue to deal with that will be explained shortly.

Re-broadening the scope of the wrangling- the most obvious variables that I need to cut are those with several null entries. I extend this to define a threshold of non-null entries that a feature variable must have in order to remain in the analysis process. This threshold must filter out enough of the unimportant variables, while simultaneously retaining most of the dataframe information. However, I cannot keep (for example) all variables with > 50% of maximum number of non-null entries. Due to the fact that I will have to drop *all* observations (rows) that contain *one* null entry, I need to be careful about setting the filter threshold too low. This would result in a large cut of data after the null-entry dropping on the remaining dataframe.

I set a threshold of > 50000 non-null entries. This is around 76% of the maximum observations, but keep around 70% of the original number of feature variables. This is after dropping 8 variables that were roughly 90% null-entries. At this point my dataframe contains 30 feature variables and, after dropping all observations that contain any null-entry, 44000 observations. Overall, this is a great retention of the bulk of the data and I shed a lot of the unhelpful weight of the data.

This concludes the wrangling process of my analysis. However, I still must process the data so that I can focus on solely those features that have the greatest impact on the binary variable 'CLASS'.

As previously and briefly mentioned, there were feature variables that contain several unique values. Not all of these feature variables were cut during the wrangling process, and will be counterproductive in the machine learning (ML) portion of the analysis for the following reason: if there are proportionally too many values in a given variable many ML models can *overfit* this data, which can lead to large computation times or a poorly-predicting model. Therefore, the unique values need a threshold filter as well. Initially I choose a number of unique values per feature to be that of approximately 10% (4000) of the number of current observations (44000). I say initially, because this number may need to be lowered or raised based upon future analysis. Applying this unique filter, I get a dataframe with 21 predictor variables (and again the one target variable 'CLASS').

The data wrangling and processing cut about half of the data feature variables by applying 2 filtration thresholds. Thus, in the end I have a dataframe with 22 variables, each with 44572 observations. This proves to still be quite a lot of data to work with as I move forward in the analysis.

Data Visualization

The data visualization portion of my analysis, much like a lot of any data science projects, can be considered an extension of data processing. Examining trends in the data that prove to be both helpful and unhelpful, and exploring the helpful insights in more depth.

The overarching goal of this capstone project is to find trends in the predictor variables of the data as they relate to the target variable 'CLASS'. The data wrangling and processing methods described previously just sought to trim the unhelpful feature variables and data from the dataframe; there was not much attention given to the binary variable. The data visualization aspect of this capstone project will proceed with specific attention given to the target variable.

Specifically, my analysis will proceed by looking at how each of the remaining 21 feature variables relate to the target variable. To this end, I had a distinct way in which I wanted to initially visualize these potential trends.

I set out to visualize the count of the unique values of each feature- how many times did a specific value occur in a given feature variable. I decided that the best way to accomplish this visualization was with a contingency table of a predictor variable and the target variable casted as a seaborn heatmap. A contingency table can be easily configured through Panda's cross tabulate function.

I constructed a function containing an algorithm that took my cleaned and processed dataframe and cross tabulated a predictor variable and the 'CLASS' target variable. Once the data was in this cross tabulated form, the seaborn heatmap could be produced.

I iterated through the entirety of the dataframe's feature variables to cross tabulate them with the target variable and then plot a heatmap of the table to visualize. The way my contingency table was constructed had values corresponding to the number of occurrences of the *top 5* most frequently occurring values of the feature variable. The heatmap was split into rows (0 and 1) of the binary classification target variable to view the counts in each classification. This singular visualization proved to not be as effective as I had previously hoped. While I did see some values dominating certain feature variables, the counts were not as enlightening as say a *percentage* might be.

My next visualization tactic had this in mind. Instead of looking at pure counts of values in each feature variable as they relate to the target variable, I would instead look at percentages of values. I wanted to look at the values in terms of two percentages: the percentage of the value count in regards to the *total* number of observations in the feature variable (i.e. this value occurred 14% of the time in this feature variable) and the percentage of value count in regards to how the values split between the target binary variable (i.e. this value appeared 60% in agreement classification and 40% in disagreement classification). Again, these visualizations would both be heatmap and thus the algorithmic intensity and computation to achieve them did not increase too much.

Viewing these 3 heatmaps per feature variable proved to be very useful. Each heatmap provided more insight into how the values of each feature variable broke down with the feature variable and how they related to the binary classification. As stated beforehand, the visualization process served to be more data processing in preparation for applying ML algorithms. Below will be a description of the processing resulting from the visualization.

After inspecting all 21 feature variables, I was able to cut even more variables that I decided were going to be unhelpful in the ML process. My first observation was in the heatmaps corresponding to the feature variables 'BIOTYPE' and 'Feature_Type'. The visualization showed me that these feature variables only had *one* unique value each and therefore would not provide any help in a binary classification prediction. These two feature variables were dropped from the dataframe. Furthermore, the feature variables 'ORIGIN' and 'CLNVC' displayed only two unique values each, with one unique value constituting most of the total percentage breakdown (similar phenomenon to the previous 2 feature variables). Again, these two variables were dropped from the dataframe. Figure 3 shows an example of this visual.

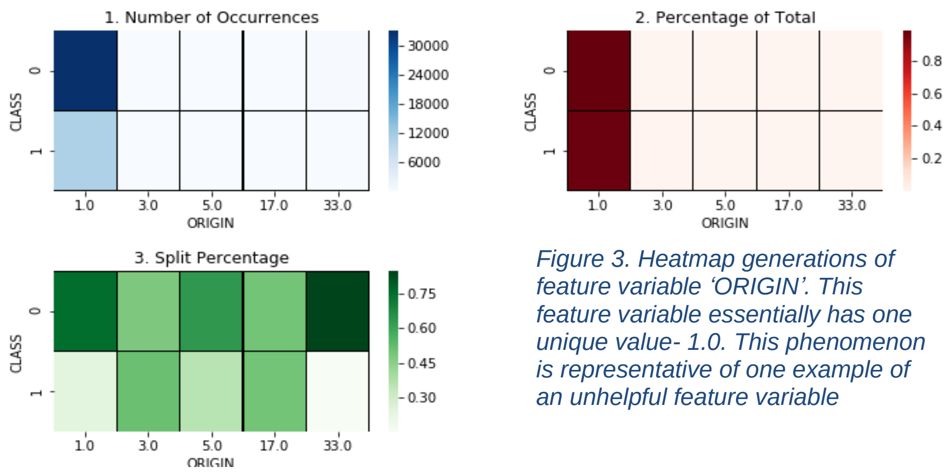


Figure 3. Heatmap generations of feature variable 'ORIGIN'. This feature variable essentially has one unique value- 1.0. This phenomenon is representative of one example of an unhelpful feature variable

On the opposite side of the filtration spectrum, there were 6 more feature variables with far too many unique values. I was able to see this in the *Percentage of Total* heatmap were even the darkest areas of the map only represented approximately 2% of the values. Figure 4 shows the heatmap output for the feature variable 'Codons'. Looking at the *Percentage of Total* heatmap, we see that the most frequently occurring value only constitutes 1.8% of the number of values. As mentioned in the data wrangling and processing phase, too many unique values could lead a model to overfit the data and perform poorly. Therefore, these feature variables were also dropped.

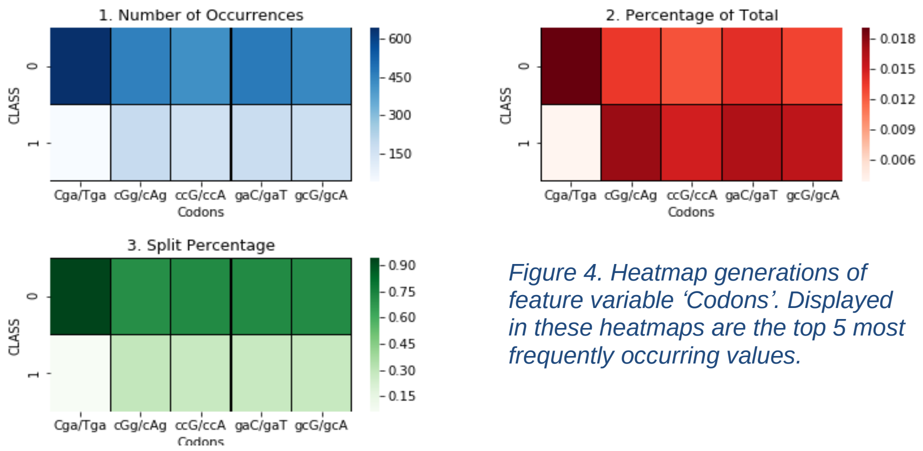


Figure 4. Heatmap generations of feature variable 'Codons'. Displayed in these heatmaps are the top 5 most frequently occurring values.

The heatmap output of the feature variable 'CHROM' is shown in Figure 5. This serves as an example of what I was looking for visually in the feature variables. The *Percentage of Total* heatmap shows the most frequently occurring value constituting roughly 16% of the values, with other highly occurring values constituting about 10% of the values. This implies that this feature variable contains significant values constituting the bulk of the data.

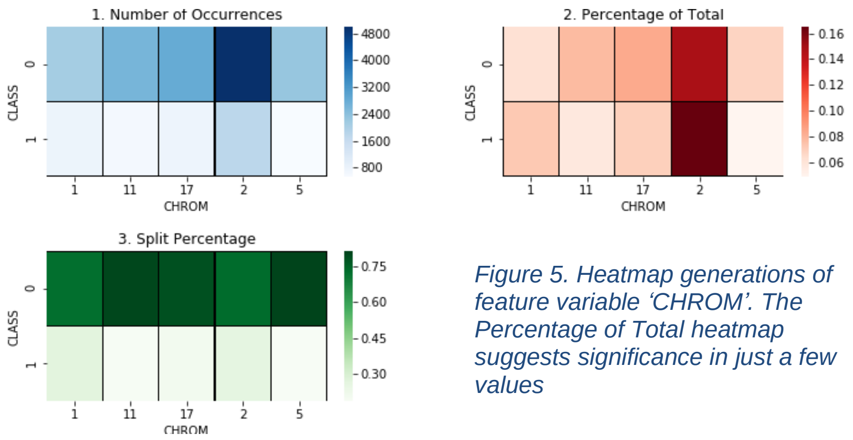


Figure 5. Heatmap generations of feature variable 'CHROM'. The Percentage of Total heatmap suggests significance in just a few values

By the end of the visualization, I dropped 11 more feature variables (current count of 11) from the dataframe and kept those that displayed the strongest correlations in their values to the target variable. Of these feature variables, I will perform a final statistical analysis on them to determine from a quantitative stance if their correlations to the target variable are in fact relevant and should be pursued in the ML algorithms.

Statistical Significance Analysis

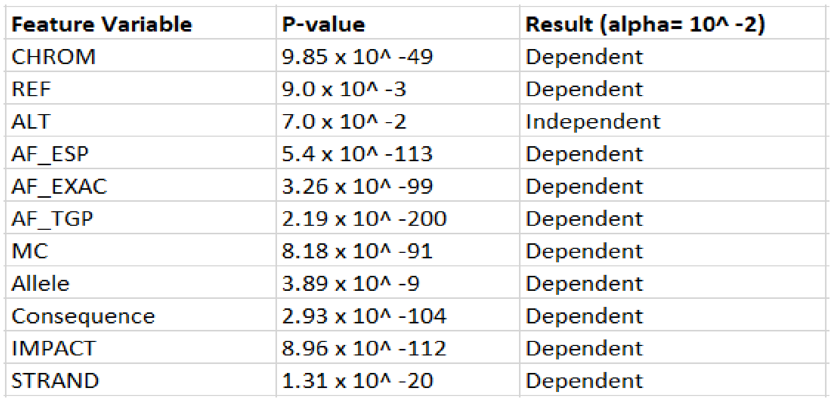
In preparation for applying machine learning algorithms to the predictor variables and target variable, I apply another step for determining relevance in my feature variables- statistical analysis. Up until this point I've used number and count thresholds, percentage cutoffs, and visualization filters. To complete my ML feature selection, I will be implementing a statistical test on the remaining feature variables.

In the case of classification problems where input variables are categorical, we can use statistical tests to determine whether the output variable is dependent or independent of the input variables. If independent, then the input variable is a candidate for a feature that may be irrelevant to the problem and removed from the dataset.

For my statistical analysis I use the Pearson's chi-squared statistical hypothesis test for quantifying the independence of pairs of categorical variables. The chi-squared test assumes (the null hypothesis) that the observed frequencies for a categorical variable match the expected frequencies for the categorical variable.

In python, the chi-squared test does this for a contingency table. During the visualization phase of this analysis, I wrote an algorithm that constructed a contingency table. This same algorithm will be used to construct the *observed frequencies* contingency tables to be used for the chi-squared test. First calculating the expected frequencies for the groups, then determining whether the division of the groups- the observed frequencies- matches the expected frequencies. The result of the test is a test statistic that has a chi-squared distribution and can be interpreted to reject or fail to reject the null hypothesis that the observed and expected frequencies are the same. The test statistic is then compared to the critical value computed at 99% probability. If the test statistic is greater than the critical value at, then we reject the null hypothesis and claim dependence.

The two functions that are used for the chi-squared test and to facilitate an interpretation of the results are chi2 and chi2_contingency from the scipy.stats module. My job then becomes to construct an algorithm that chains these two functions together and adds supporting lines of code to succinctly interpret the results. A summary of my results is seen in Table 1 below.



Feature Variable	**P-value**	**Result (alpha= 10^ -2)**
CHROM	9.85 x 10^ -49	Dependent
REF	9.0 x 10^ -3	Dependent
ALT	7.0 x 10^ -2	Independent
AF_ESP	5.4 x 10^ -113	Dependent
AF_EXAC	3.26 x 10^ -99	Dependent
AF_TGP	2.19 x 10^ -200	Dependent
MC	8.18 x 10^ -91	Dependent
Allele	3.89 x 10^ -9	Dependent
Consequence	2.93 x 10^ -104	Dependent
IMPACT	8.96 x 10^ -112	Dependent
STRAND	1.31 x 10^ -20	Dependent

Table 1. Summary of chi-squared test of significance between feature variables and target variable.

Of the 11 statistically tested feature variables, only one failed to reject the chi-squared null hypothesis of independence- ‘ALT’. As a result, it is dropped from the dataframe.

The results of the chi-squared statistical analysis determined a statistically highly likely non-independence correlation between 11 feature variables. These variables will be preprocessed using feature engineering before applying ML algorithms to them.

Feature Engineering and Machine Learning

The results of all of my data wrangling, processing, visualizations, and statistical analysis left me with 11 feature variables to work with in applying machine learning algorithms. However, as is often in the case, most of these feature variables are categorical and therefore many algorithms cannot be directly applied. Therefore, my first step in the machine learning phase of my analysis is feature engineering.

By feature engineering, I mean turning my data into something interpretable by popular algorithms- i.e. numbers. I decide to preprocess my features in 3 different ways: one hot encoding, label encoder, and feature hashing.

One hot encoding, essentially, takes each unique value in a feature variable and gives it its own column in a dataset. Then if an observation took on this value, that values’ column gets a 1 for the observation. Else, it gets a 0. Now the problem with this becomes apparent very quickly: what if a feature variable has hundreds or even thousands of unique values? It turns out that three of my variables had a magnitude of 1000 unique values. In my visualization phase, I did notice that these three feature variables had about 50% of one specific value and the other 50% was split between thousands of other values. This is a perfect case to break these feature variables down into a one-vs-rest encoding scheme. In this regard, the feature variable is split into two binary columns- whether the observation took on the majority value or not. The other feature variables were all one hot encoded and this made up one preprocessed dataframe.

The next feature engineering method I employed is the label encoder. This preprocessing method is by far the most straightforward. For every unique value in the feature variable, the value gets mapped to a unique numeric key. A quick apply method on the main dataframe applies this function to all feature variables and the label encoded dataframe is constructed.

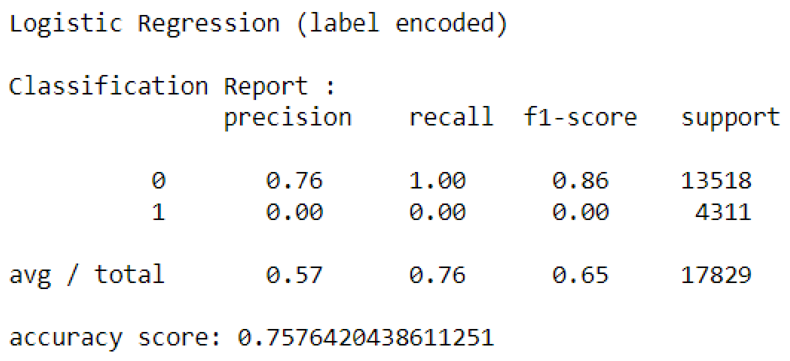
The final method I used for preprocessing the data is feature hasher. Feature hashing is a fast and space efficient way of turning arbitrary (and in this case categorical) into indices in a matrix. The hashing trick is similar to word vectorizing (used in NLP) in that it stores information in a sparse matrix. The downside to this method is that it cannot handle float variables, which four of my feature variables happen to be. Instead these four feature variables get label encoded and combined with the rest of the feature hashed dataframe, thus completing the preprocessing of my final dataframe.

I decide to apply several machine learning algorithms to these preprocessed dataframes in order to maximize model predictions. The algorithms I chose to apply are logistic regression (LR), decision tree (DT), random forest (RF), gradient boosting (GB), and support vector machine (SV). These algorithms cover a wide range of machine learning tactics and all have support for classification predictions. In these models, I will be analyzing a few metrics to gauge model performance: accuracy score, precision score, recall score, and f1 score.

- Accuracy- ratio of correctly predicted observation to the total observations
- Precision- number of true positives divided by the number of true positives plus the number of false positives
- Recall (Sensitivity) - number of true positives divided by the number of true positives plus the number of false negatives
- F1 score- harmonic mean of precision and recall

The models are tested on their predictions of the binary classification of lab agreement (0) or disagreement (1).

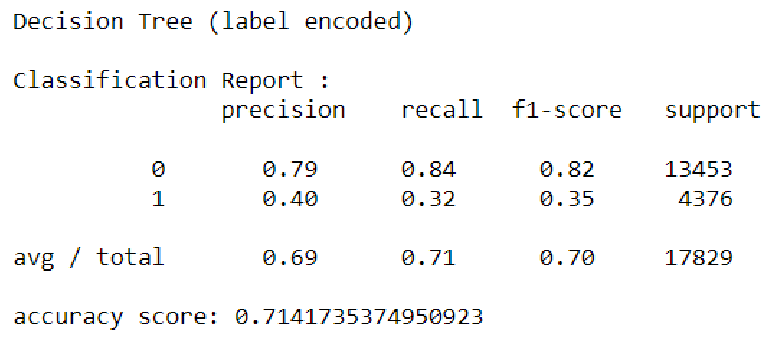
I first apply the LR model to my preprocessed dataframes. The accuracy score looks good for all three at around 75%, but the recall and f1 scores are 0 for the lab disagreement variable. While the model performed well in predicting lab agreements, it performed very poorly when it came to accurately identifying disagreements.



```
Logistic Regression (label encoded)

Classification Report :
              precision    recall  f1-score   support

          0       0.76      1.00      0.86     13518
          1       0.00      0.00      0.00      4311

avg / total       0.57      0.76      0.65     17829

accuracy score: 0.7576420438611251
```

Table 2. Classification Report and accuracy for the Logistic Regression Model applied to the label encoded data

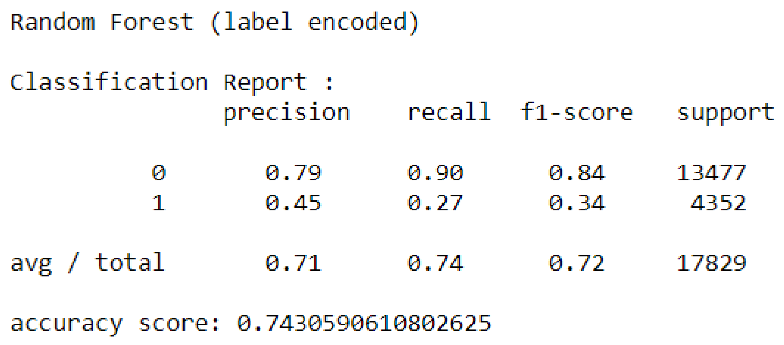
The DT algorithm saw a decrease in accuracy by a couple percent, but a dramatic increase in the recall and f1 score for the disagreement prediction. This shift came at the cost of the agreement prediction losing some percentages across the recall and f1 scores. However, the overall prediction performance is vastly superior to that of the LR model.



```
Decision Tree (label encoded)

Classification Report :
              precision    recall  f1-score   support

          0       0.79      0.84      0.82     13453
          1       0.40      0.32      0.35      4376

avg / total       0.69      0.71      0.70     17829

accuracy score: 0.7141735374950923
```

Table 3. Classification Report and accuracy for the Decision Tree classifier applied to the label encoded data.

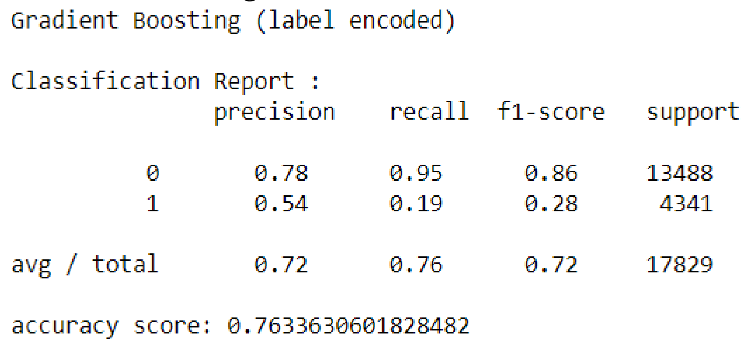
Since the Random Forest by nature is derived from the Decision Tree, it is no surprise that I notice similar performances in the metrics.



```
Random Forest (label encoded)

Classification Report :
              precision    recall  f1-score   support

          0       0.79      0.90      0.84     13477
          1       0.45      0.27      0.34      4352

avg / total       0.71      0.74      0.72     17829

accuracy score: 0.7430590610802625
```

Table 4. Classification Report and accuracy for the Random Forest model applied to the label encoded data.

The GB model resulted in the highest accuracy yet at 76%. Unfortunately, this accuracy came at the cost of the disagreement recall score.



```
Gradient Boosting (label encoded)

Classification Report :
              precision    recall  f1-score   support

          0       0.78      0.95      0.86     13488
          1       0.54      0.19      0.28      4341

avg / total       0.72      0.76      0.72     17829

accuracy score: 0.7633630601828482
```

Table 5. Classification Report and accuracy for the Gradient Boosting model applied to the label encoded data.

The SV algorithm is excluded from this report. The computation time for all preprocessed data was exceedingly high and the results were far poorer than most of the models.

The results of applying ML algorithms determined that the preprocessed data via label encoding and feature hashing provided the best and similar results. Furthermore, in terms of overall metric analysis the Decision Tree classifier performed the best. I analyze this model to understand which features showed the greatest importance in predictions. From the analysis of feature importance, I saw that the feature variables ‘AF_EXAC’, ‘AF_TGP’, ‘AF_ESP’, and ‘CHROM’ all played a big role in the prediction of lab agreement or disagreement for the decision tree model. I sought to confirm these variable’s impact through examination of the feature hashed data and the gradient boosting model. With further examination, these feature variables importance is confirmed. The visualization of these feature variables importance and consistency can be seen below in Figures 6-8.

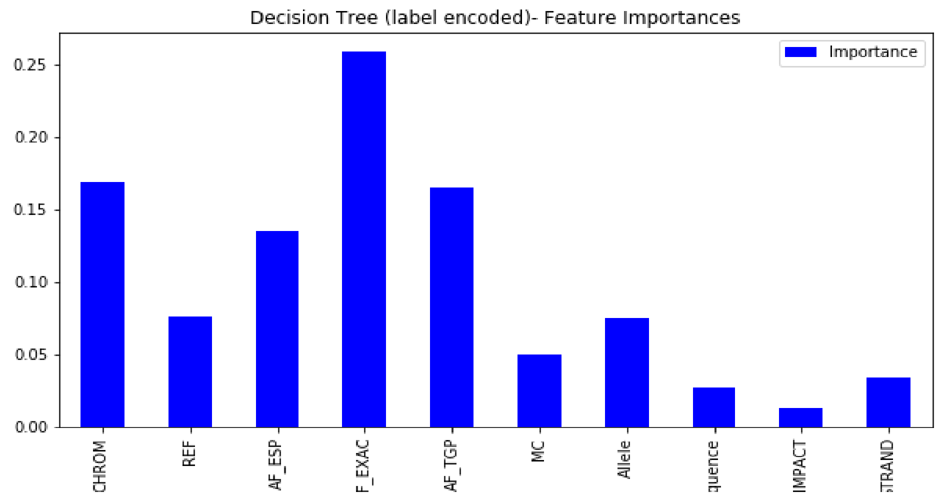


Figure 6. Bar chart visualization of feature importance of the label encoded data as interpreted by the Decision Tree classifier

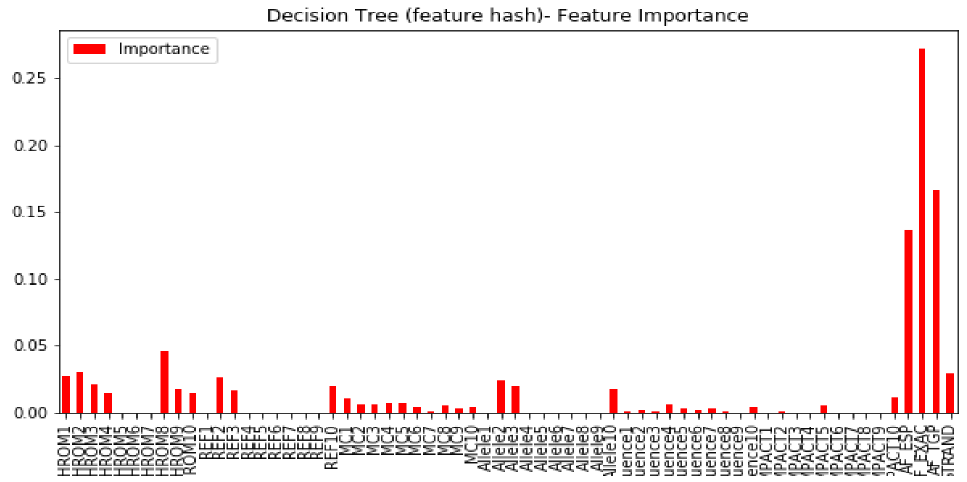


Figure7. Bar chart visualization of feature importance of the feature hashed data as interpreted by the Decision Tree classifier

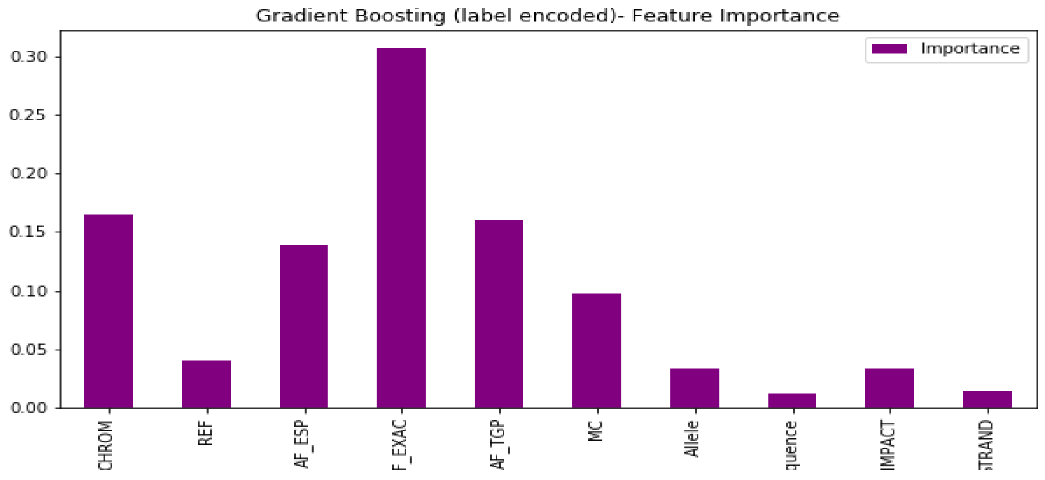


Figure 8. Bar chart visualization of feature importance of the label encoded data as interpreted by the Gradient Boosting algorithm

I decided to optimize the Decision Tree model by tuning a couple of parameters to see if I could increase the model performance. Of the classifier, I set out to tune ‘max_depth’, ‘min_samples_split’, and ‘min_samples_leaf’. To do this, I implement a grid search with a cross-validation on the classifier and data, using f1 score to judge the performance of the model. Tuning just 3 parameters takes quite a bit of computation time so I do not seek to add another parameter. I instantiate a new Decision Tree with the computed best parameters and apply the model once more.

```
Optimized Decision Tree (label encoded)

Classification Report :
              precision    recall  f1-score   support

          0       0.80      0.88      0.84     13482
          1       0.45      0.31      0.37      4347

avg / total       0.71      0.74      0.72     17829

accuracy score: 0.7405911716865781
```

Table 6. Classification Report and accuracy for the tuned Decision Tree model applied to the label encoded data.

This new model saw an improvement of at least a couple percent in every score (aside from recall) for both lab agreement and disagreement. Furthermore, this new model had an emphasizing effect on the feature importance shown below in Figure 9. It emphasized the importance of ‘AF_EXAC’ and ‘AF_TGP’, but not ‘AF_ESP’. More on this phenomenon later in the report.

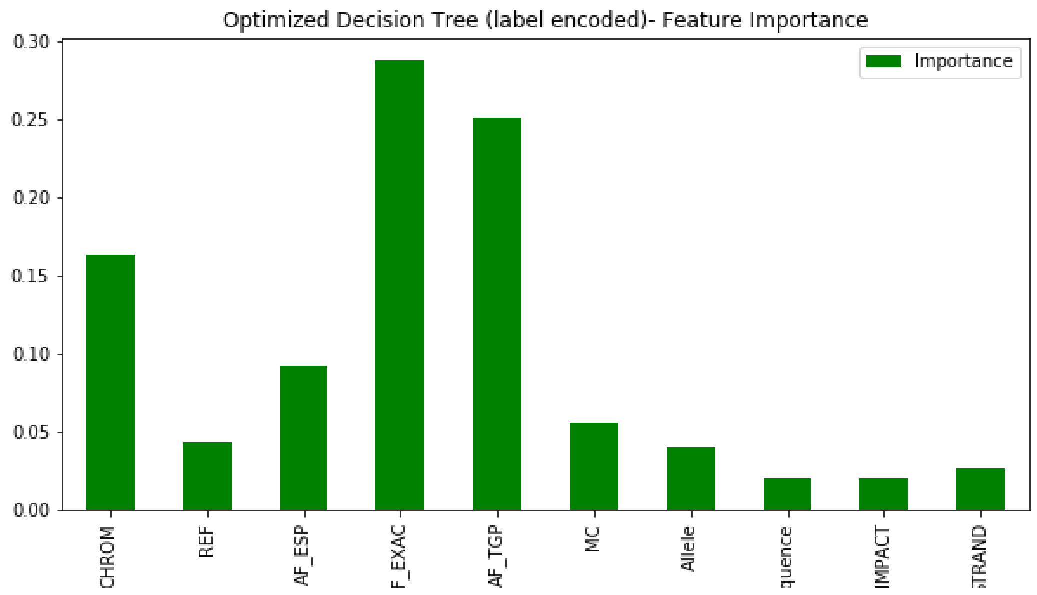
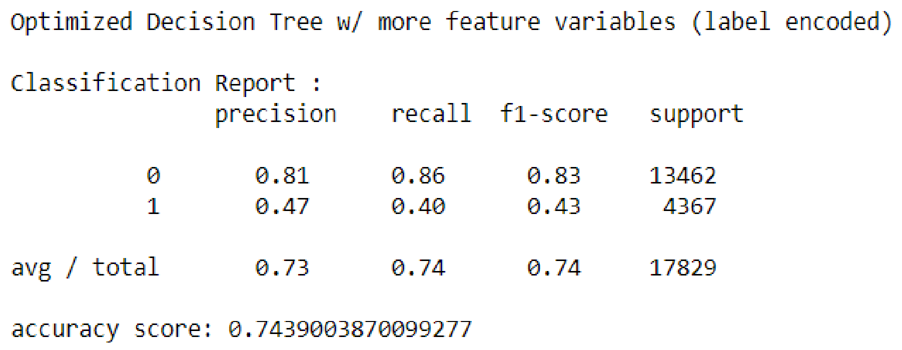


Figure 97. Bar chart visualization of feature importance of the label encoded data as interpreted by the optimized Decision Tree classifier.

I sought to take this machine learning analysis even one step further. As previously described, the bulk of this project was an exercise in filtering and processing the data to give the applied algorithms the most helpful feature variables (to save on computation time and avoid overfitting). However, I wondered if implementing more variables would in fact increase performance drastically enough that would warrant the computation time and not merely overfit the data. I used a previous dataframe, which contained several more feature variables (before they were filtered out) to test a new decision tree. This dataframe was label encoded and again, I used the optimized decision tree found via the grid search.



```
Optimized Decision Tree w/ more feature variables (label encoded)

Classification Report :
               precision    recall  f1-score   support

          0       0.81      0.86      0.83     13462
          1       0.47      0.40      0.43      4367

avg / total       0.73      0.74      0.74     17829

accuracy score: 0.7439003870099277
```

Table 7. Classification Report and accuracy for the tuned Decision Tree model applied to the label encoded data with more feature variables.

The computation time noticeably increased and the metrics only showed a correspondingly slight increase. Furthermore, in examining the importance of features the model appears to be overfitting on the abundance of new features. While the two variables 'AF_EXAC' and 'AF_TGP' still stand out the most in terms of importance, there are several others that have "significant" importance in the prediction, all of which have a high count of unique values. This ultimately suggests overfitting on these additional feature variables.

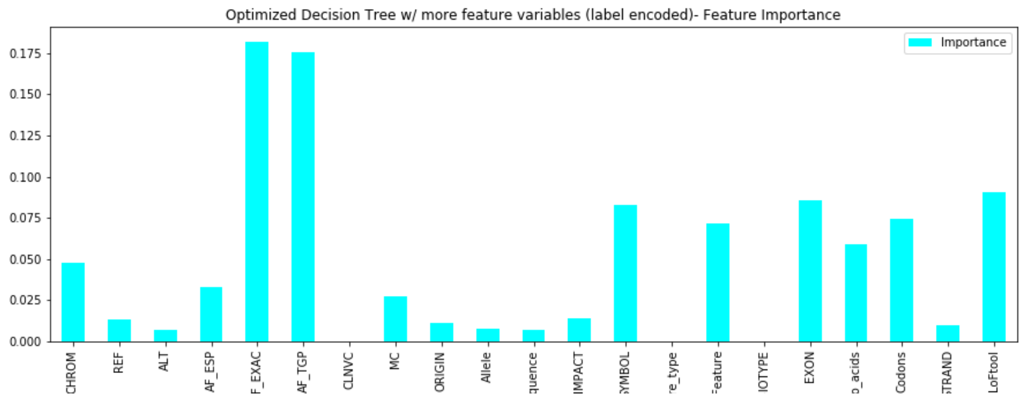


Figure 80. Bar chart visualization of feature importance of the label encoded data (with more feature variables) as interpreted by the optimized Decision Tree classifier

Both the label encoded data and feature hashed data perform similarly across all of my applied algorithms. However, the label encoded data seemed to have slightly less computation time when applying complex models such as the decision tree, random forest, and gradient boosting. Weighing all of the factors that go into model performance, the optimized decision tree when applied to the label encoded data performs the best when it comes to predicting classification agreements *and* disagreements (Table 6). Meaning it produced the best accuracy score combined with the highest precision, recall, and f1 scores with the *lowest* computation time in both agreements and disagreements.

The result of the ML algorithm application revealed an interesting insight. The two feature variables 'AF_EXAC' and 'AF_TGP' consistently had the biggest impact on more than one model's prediction. As defined in the dataset, these two feature variables are the allele frequencies from two of the three participating clinical labs. The interesting insight is that the third lab's allele frequency observations did not have as much of an impact on our models' predictive powers. This suggests that the two labs that contributed the most to the target binary variable classification agreement are ExAC and 1000 Genome Project (TGP). Extracting a greater and more explicit interpretation from this: the laboratory agreement hinged upon these two labs. If ExAC and TGP agreed on a genetic variants' classification, the third lab GO-ESP would also tend to agree and the binary variable would take on a 0 value, indicating *all* labs agreed on their classifications. Looking forward, the analysis would dive deeper into these two allele frequencies in an attempt to understand the cause of their suggested importance. Ultimately the analysis would continue to test the hypothesis that the two labs ExAC and TGP defined allele frequencies that resulted in agreements/disagreements across all *three* labs.