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. Thus the focus of this analysis will be on the binary classification, while examining relative variables that may be linked to each classification case.

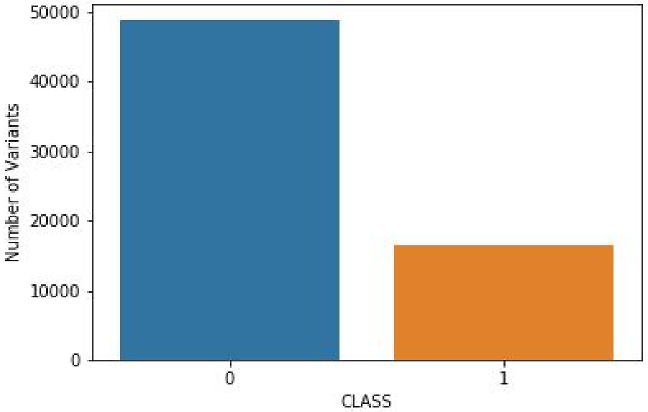
The binary variable of interest, 'CLASS' will be analyzed across several feature variables (columns) to determine patterns and relationships between classifications and other measured quantities. Both binary cases- classification agreements and disagreements- will be analyzed.

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.

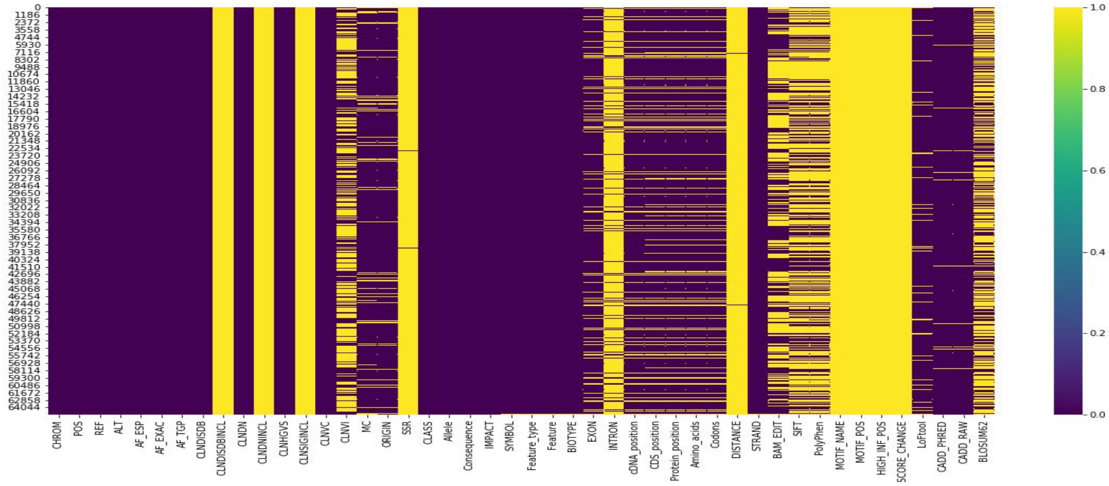
Data Wrangling and Processing

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. The plot shows the data breakdown between lab classification agreements (0) and conflicts (1).



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- 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.



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.

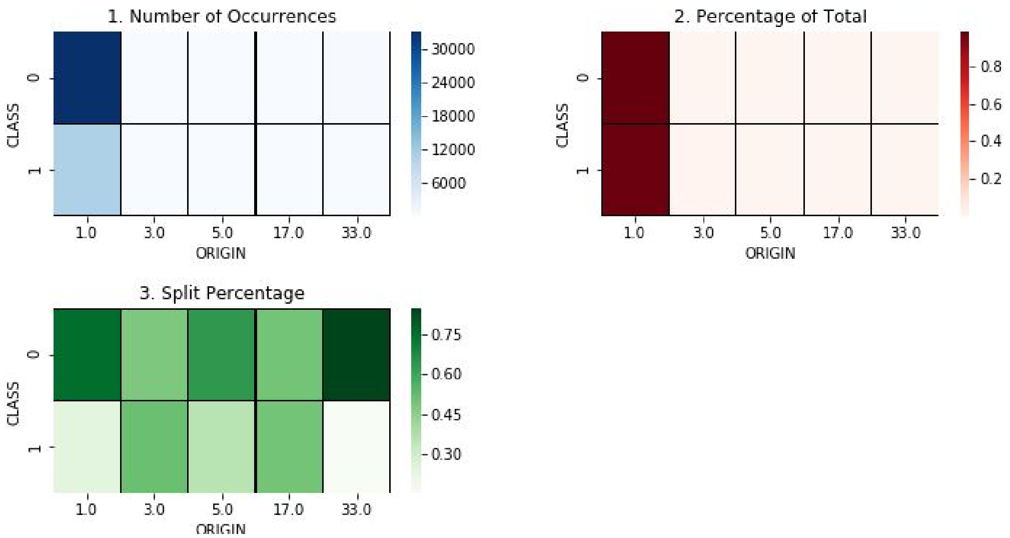
I constructed a function containing an algorithm that took my cleaned and processed dataframe and cross tabulated a predictor variable and the 'CLASS' target variable. Once the data was in this cross tabulated form, the seaborn heatmap could be produced.

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.

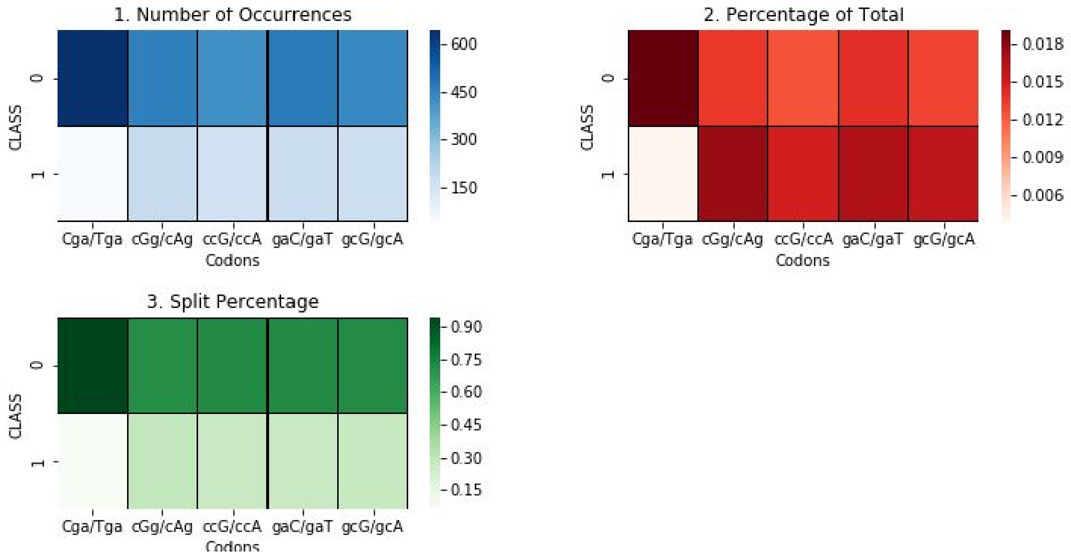
My next visualization tactic had this in mind. Instead of looking at pure counts of values in each feature variable as they relate to the target variable, I would instead look at percentages of values. I wanted to look at the values in terms of two percentages: the percentage of the value count in regards 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.

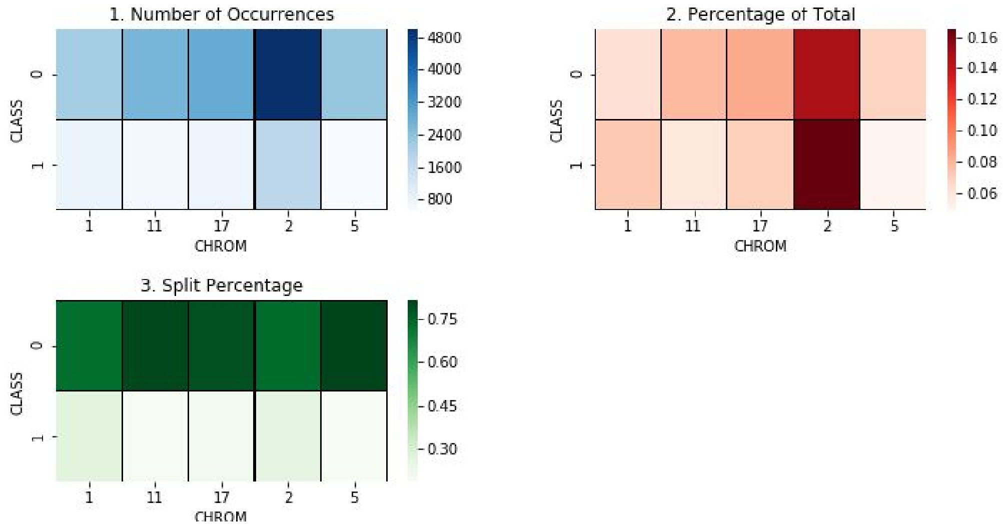
After inspecting all 21 feature variables, I was able to cut even more variables that I decided were going to be unhelpful in the ML process. My first observation was in the heatmaps corresponding to the feature variables 'BIOTYPE' and 'Feature_Type'. The visualization showed me that these feature variables 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. Below is the output of the heatmap generation of 'ORIGIN'. It is clear from the *Number of Occurrences* and *Percentage of Total* heatmaps that there is clear majority value in this 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. Below is 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.



Below is the heatmap output of the feature variable ‘CHROM’. 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 occuring value constituting roughly 16% of the values, with other highly occurring values constituting about 10% of the values.



By the end of the visualization, I was able to drop 10 more feature variables (current count of 11) from the dataframe and keep 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

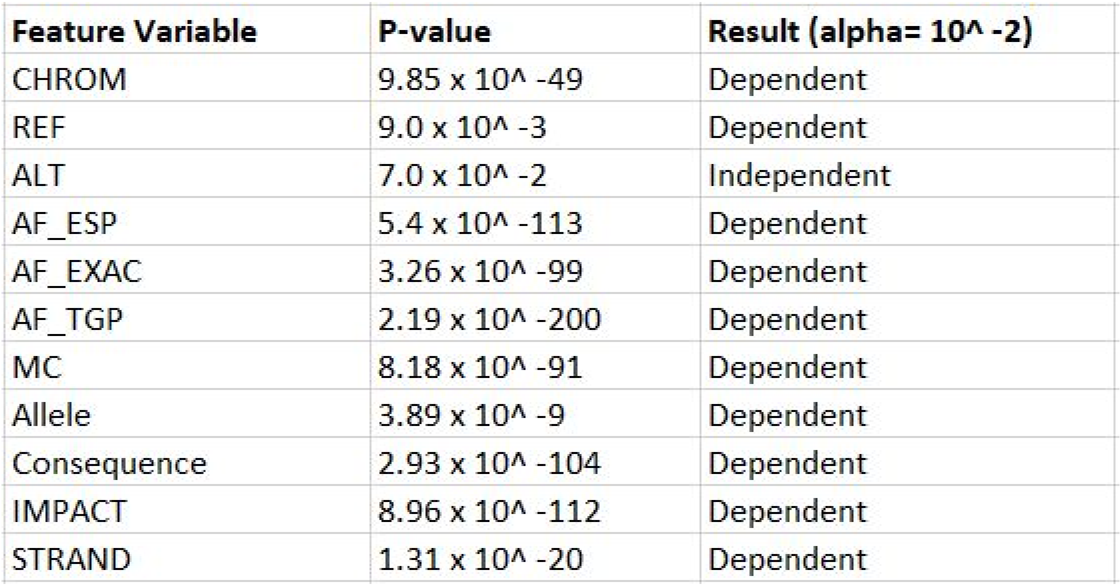
In preparation for applying machine learning algorithms to the predictor variables and target variable, I apply another step for determining relevance in my feature variables- statistical analysis. Up until this point I've used number and count thresholds, percentage cutoffs, and visualization filters. 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 tabular summary of my chi-squared test can be seen 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

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.