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. 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
```

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
```

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

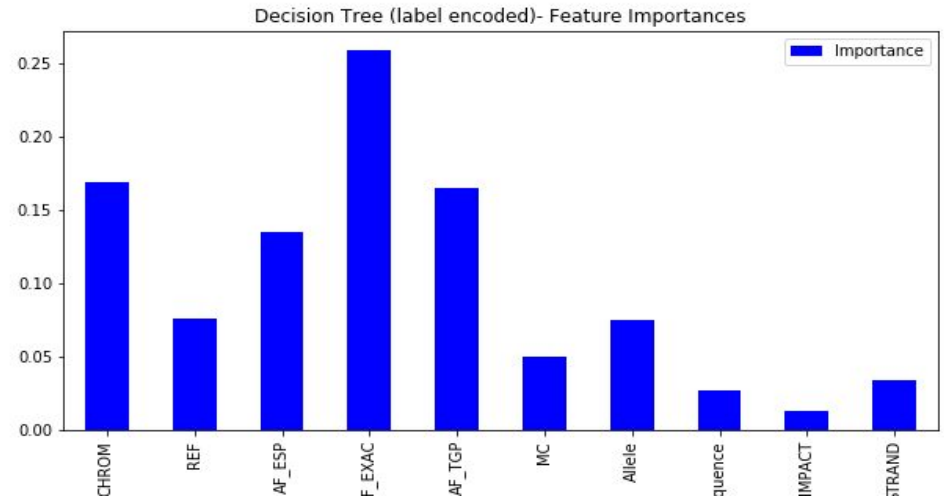
```
Random Forest (label encoded)

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
```

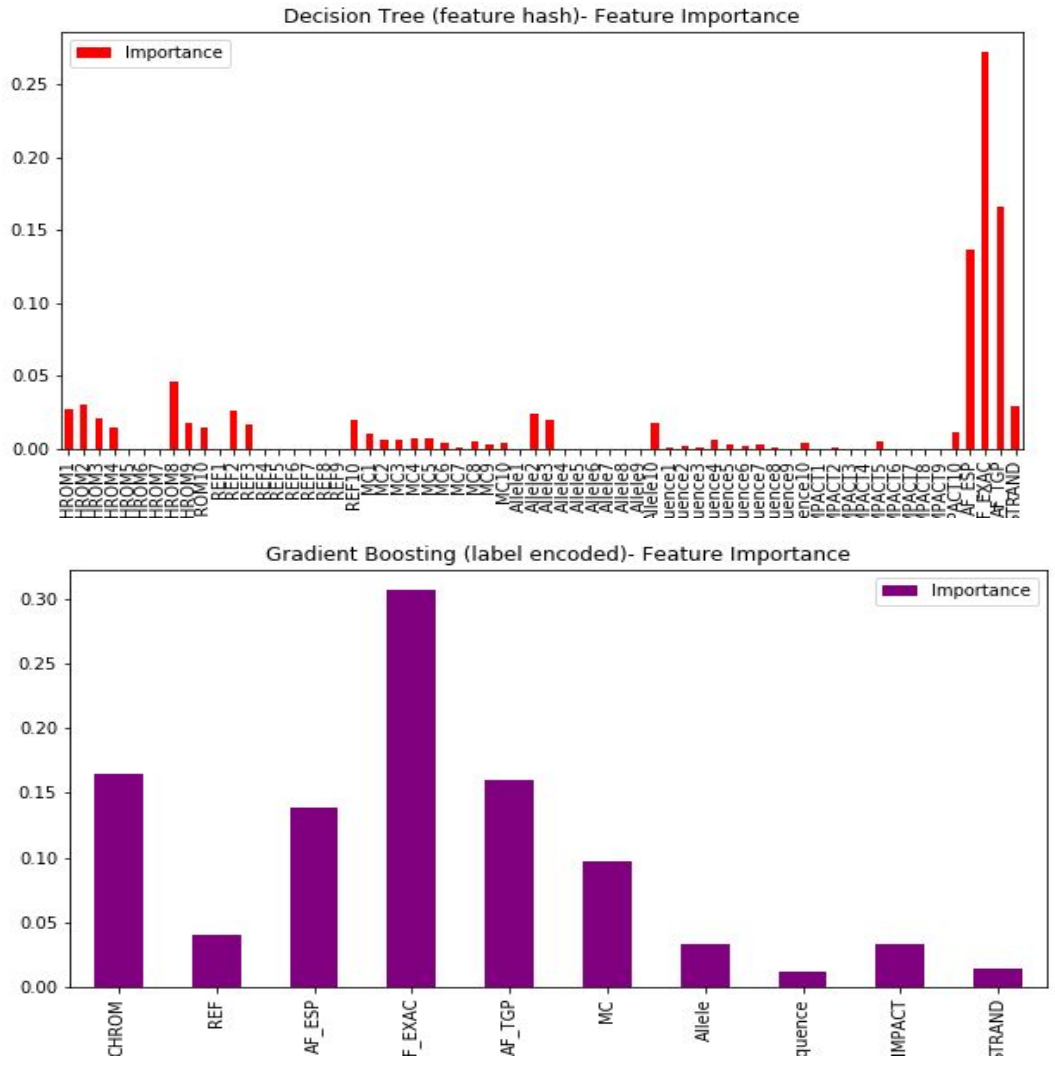
The GB model resulted in the highest accuracy yet at 76%. Unfortunately this accuracy came at the cost of the disagreement recall score.

```
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
```

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 the plots.

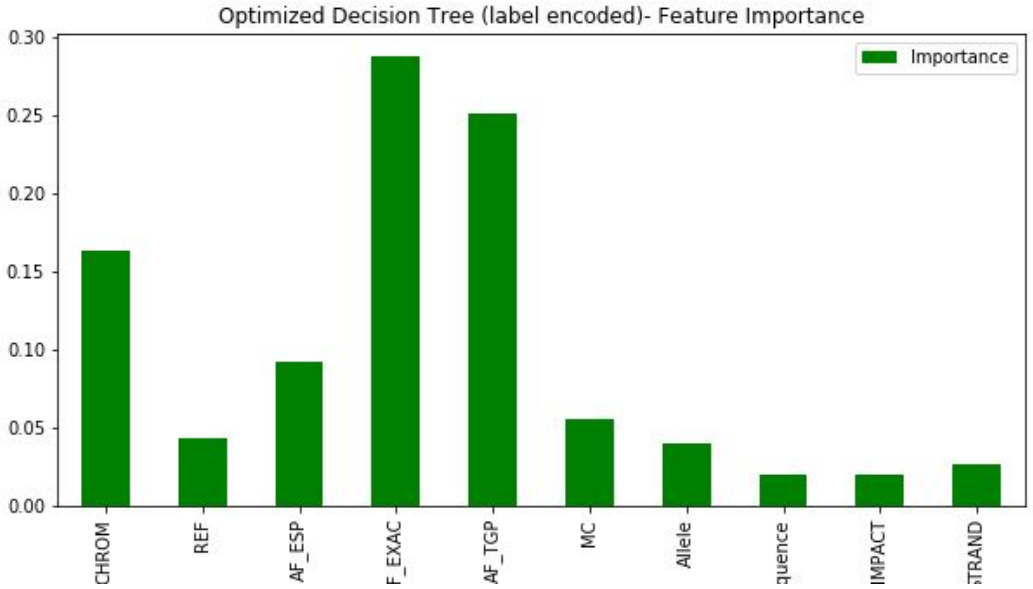




I decided to optimize the Decision Tree model by tuning a couple of parameters to see if I could increase the model performance. Of the classifier, I set out to tune 'max_depth', 'min_samples_split', and 'min_samples_leaf'. To do this, I implement a gridsearch with a cross-validation on the classifier 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
```

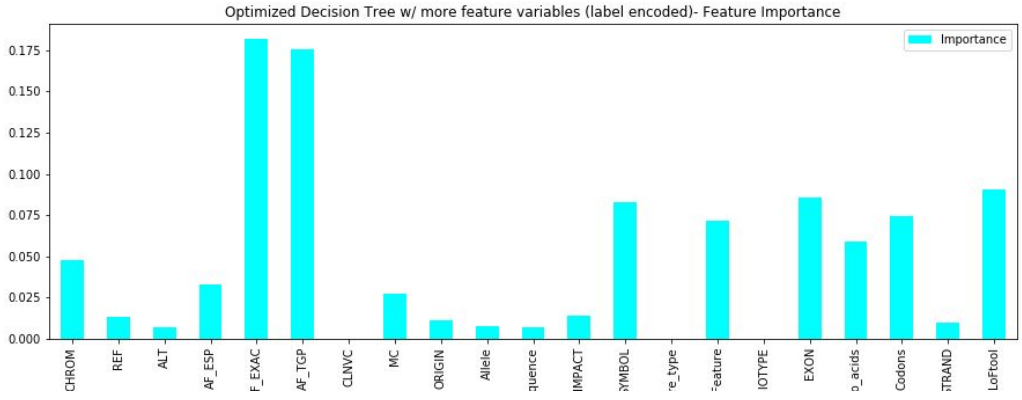
This new model saw an improvement of at least a couple percent in every score (aside from recall) for both lab agreement and disagreement. Furthermore, this new model had an emphasizing effect on the feature importance (shown below). It emphasized the importance of 'AF_EXAC' and 'AF_TGP', but not 'AF_ESP'.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



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 gridsearch. The classification report is shown below.

```
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
```

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.



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 the 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 are *ExAC* and *1000 Genome Project* (TGP). Extracting a greater 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.