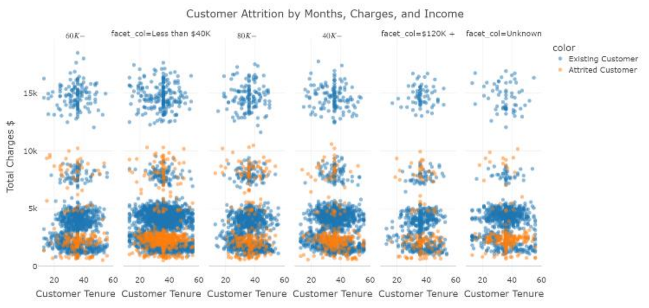
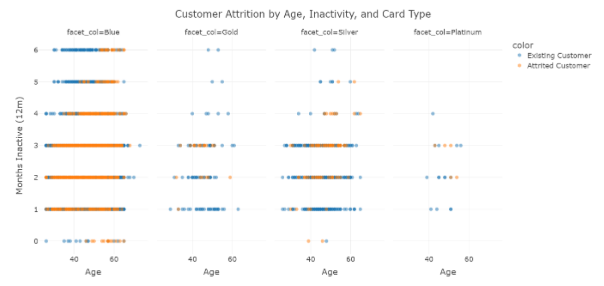
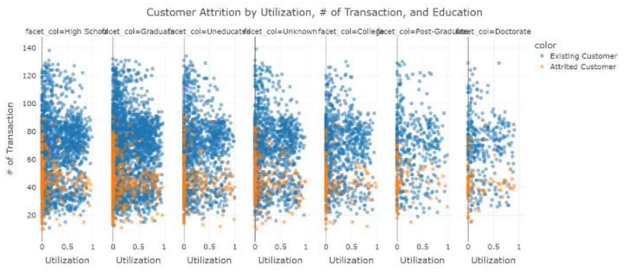
Identifying Customer Attrition by: Stefan Zamurovic

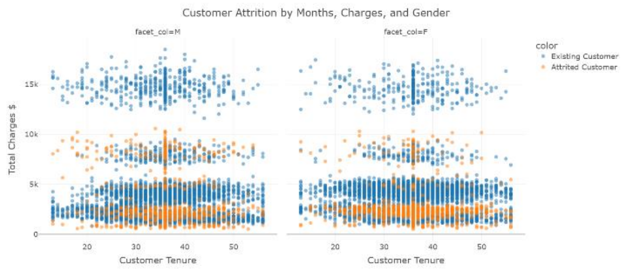
To start building a model the process was split into five different parts: Data Analysis, Data Preparation, Model Selection & Training, Hyperparameters Tuning, and Analysis. Starting with Data Analysis I read through a good portion of the data to get a good grasp of what I was working with. With 20 different features and a combination of classifications that are integers, floats, or strings, there was a lot to work with. The first data point that caused some initial concern was the "unknown" classification in some of the features. Next was deciding whether I wanted to change the categorical features to an integer-based classification or go with one-hot encoding. Last I picked a few features to build graphs of to look for any correlation or relationships in the data to help with deciding which features to use in my model.

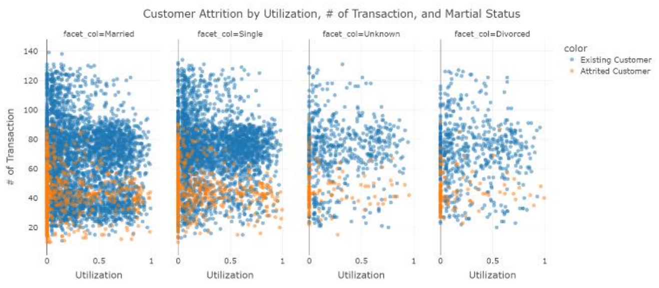
Now to start tackling the different parts of my Data Analysis in my Data Preparation step. First was figuring out what to do with the "unknown" classification that some features have. There were too many data points that included the "unknown" classification for me to ignore or remove the data that was included so I needed to find a way to use it. First, I tried to reclassify it by comparing the other features for that data point to other data that were similar, but I was unsuccessful for a good amount of the data was hard to reclassify without making large assumptions that didn't seem right to do as it looked like the data was being changed drastically. I started to think of another way to reclassify the data but in the end, I realized that I could just use the "unknown" as a classification for the model. Next was deciding if I should change categorical data types to integers or do one-hot encoding, and I decided to not change it integers for the few models I was looking to use would be able to do classification with categorical data. Last was build different graphs of the data to look for a correlation and relationships in the data.





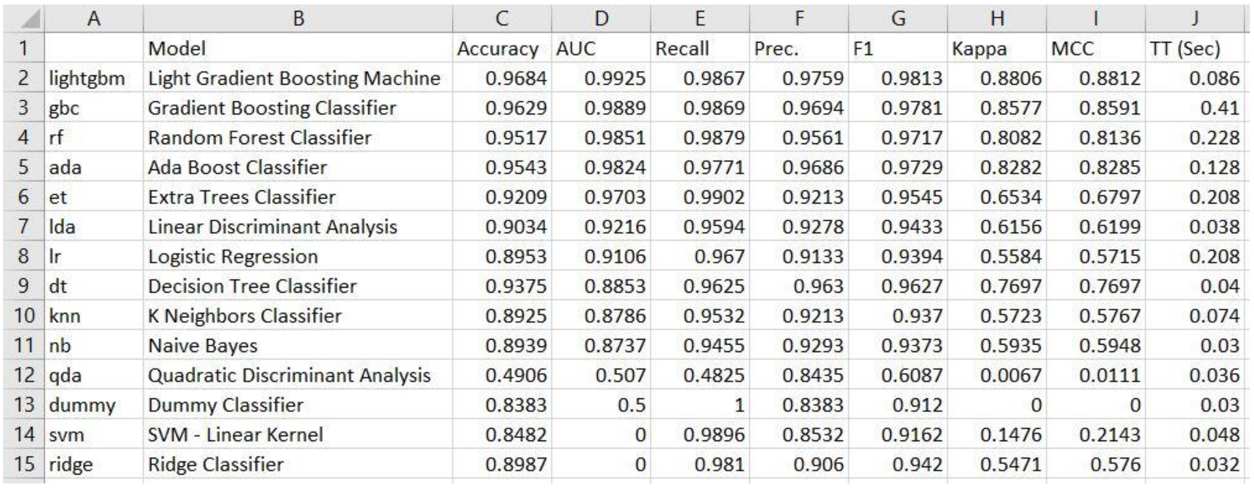






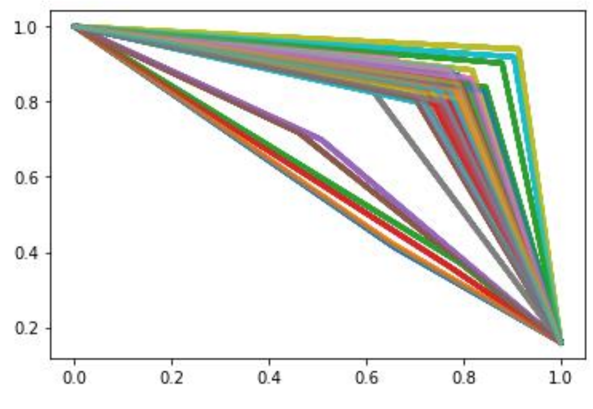
Looking through the graphs there were a couple strong relations between features and whether or not that customer was Attrited. First was the classification of the Card Type and Attrited Customers and the second was Marital Status and Attrited Customers. This was important for when I start building my model to know what features and classifications to use to help train the model.

Next was to pick my model and start training it. I first decided to take a broad look at a bunch of different models and to see which model would work best. I already had my eyes set on LightGBM, AdaBoost, and Random Forest but I still included multiple other models to so I would jump the gun and leave out a model that might be better then one of those three. In the end LightGBM was the best model compared to 13 other models.



	A	B	C	D	E	F	G	H	I	J
1		Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
2	lightgbm	Light Gradient Boosting Machine	0.9684	0.9925	0.9867	0.9759	0.9813	0.8806	0.8812	0.086
3	gbc	Gradient Boosting Classifier	0.9629	0.9889	0.9869	0.9694	0.9781	0.8577	0.8591	0.41
4	rf	Random Forest Classifier	0.9517	0.9851	0.9879	0.9561	0.9717	0.8082	0.8136	0.228
5	ada	Ada Boost Classifier	0.9543	0.9824	0.9771	0.9686	0.9729	0.8282	0.8285	0.128
6	et	Extra Trees Classifier	0.9209	0.9703	0.9902	0.9213	0.9545	0.6534	0.6797	0.208
7	lda	Linear Discriminant Analysis	0.9034	0.9216	0.9594	0.9278	0.9433	0.6156	0.6199	0.038
8	lr	Logistic Regression	0.8953	0.9106	0.967	0.9133	0.9394	0.5584	0.5715	0.208
9	dt	Decision Tree Classifier	0.9375	0.8853	0.9625	0.963	0.9627	0.7697	0.7697	0.04
10	knn	K Neighbors Classifier	0.8925	0.8786	0.9532	0.9213	0.937	0.5723	0.5767	0.074
11	nb	Naive Bayes	0.8939	0.8737	0.9455	0.9293	0.9373	0.5935	0.5948	0.03
12	qda	Quadratic Discriminant Analysis	0.4906	0.507	0.4825	0.8435	0.6087	0.0067	0.0111	0.036
13	dummy	Dummy Classifier	0.8383	0.5	1	0.8383	0.912	0	0	0.03
14	svm	SVM - Linear Kernel	0.8482	0	0.9896	0.8532	0.9162	0.1476	0.2143	0.048
15	ridge	Ridge Classifier	0.8987	0	0.981	0.906	0.942	0.5471	0.576	0.032

Now that I knew which model, I would be using next was to start training my model with the data. Since I already did my data preparation step there wasn't much left for me to do to make sure there was not any issue with fitting the data to the LightGBM model. Next, I split the data up to start doing 5-fold cross validation to get a stronger understanding of my results so that we can move to tuning the data. When splitting the data, a few issues raised for the data was not being evenly split. Certain prebuilt k-fold or data splitting methods would split the data even but not the Attritted and Existing customers. To overcome this I split the attritted data and existing customers data from each other, and then separated them into the 5 folds and validation set. Now the data was split evenly between attritted and existing customers I recombined the data and shuffled the data points around, so it was no longer sorted. To tune the data, I decided to do Hyperparameter Tuning and picked a few parameters such as: learning rate, boosting type, sub features, column sample bytree, number of leaves, and max depth. To tune these parameters, I modeled 1000 different combinations of the parameters and trained the model 1000 times for each combination using accuracy, precision, recall and AUC as evaluation metrics for each model. Interestingly enough after implementing the tunned parameters the mean Accuracy and the Precision Recall AUC were lower than the base model, but the Standard Deviation of the mean Accuracy was smaller with the tunned parameters than the base model which means that it was working in some way. After not understanding why the accuracy was lower but the standard deviation was also lower, I looked into what could be going wrong. It turns out that even though the model was training well it was overfitting and was not doing well with the validation data set. Just to make sure this wasn't just a single occurrence I ran a few more times and they all seemed to have similar results. Though I wanted the hyperparameter tunning to work so I decided to run a couple more tests to try and improve my results, but nothing seemed to do the trick.



The figure above is graph of each of the precision and recall curves for every fold of the data of both the base model and tunned model. This helped shows how the model was improving over folds. For the base model was getting a mean Accuracy of around 0.9307456790123458 and for the tunned model I was getting a mean Accuracy of around 0.9226469135802469. Now the standard deviation of the accuracy of the base model was 0.04924628660517176 and standard deviation of the accuracy of the tunned model was 0.04675077768491327. Last the area under the precision recall curve for the base model was 0.9337733660307501 and the area under the precision recall curve for the tunned model was 0.9193370132953466. In conclusion the mean Accuracy of my model was 0.9307456790123458 with a Standard Deviation of 0.04924628660517176 and the AUC of my Precision Recall Curve was 0.9337733660307501.