CSCA 5622 Supervised Learning Final Project

Bradley Leavitt

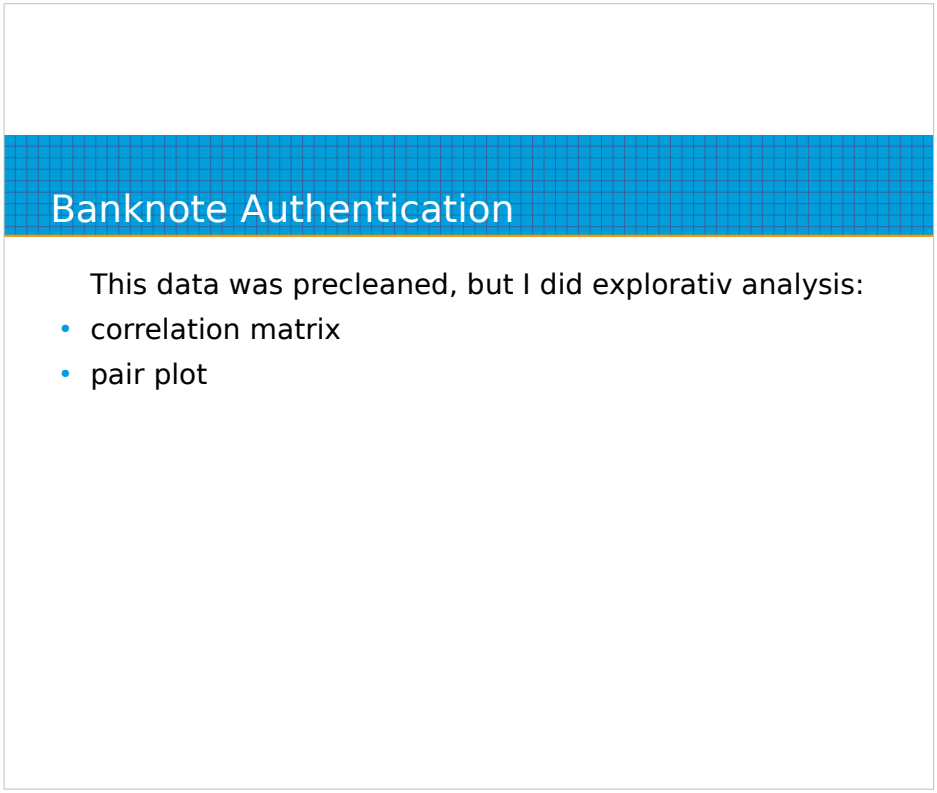
Banknote Authentication

Lohweg, V. (2012). Banknote Authentication [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C55P57.

https://github.com/brle1242/CSCA-5622-Supervised-Learning-Final-Project

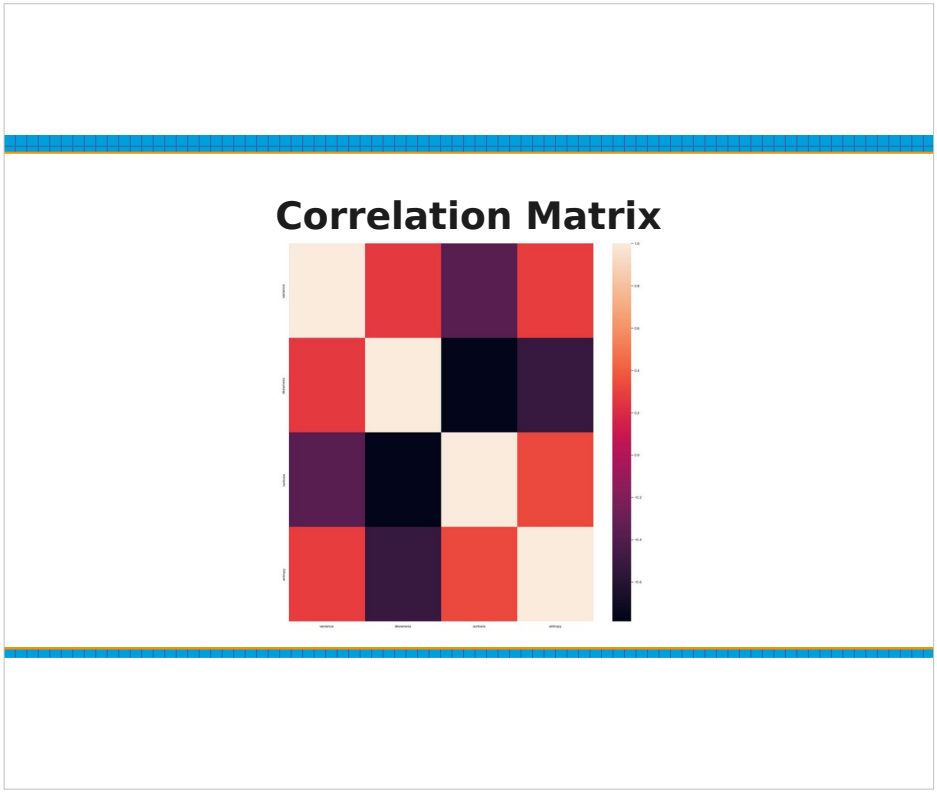
Data was extracted from images that were taken from genuine and forged banknote-like specimens.

An industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object, gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tools were used to extract features from images.

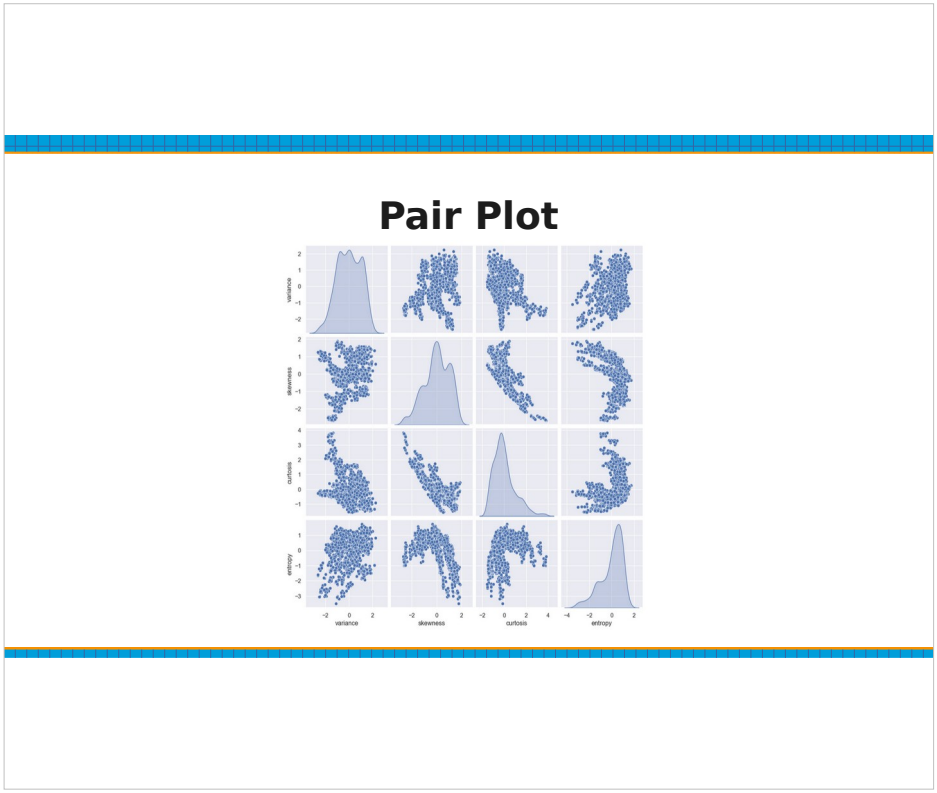


I checked the python type of the data and verified that it was a pandas dataframe. I then checked the dataframe column names and verified that there were no null values.

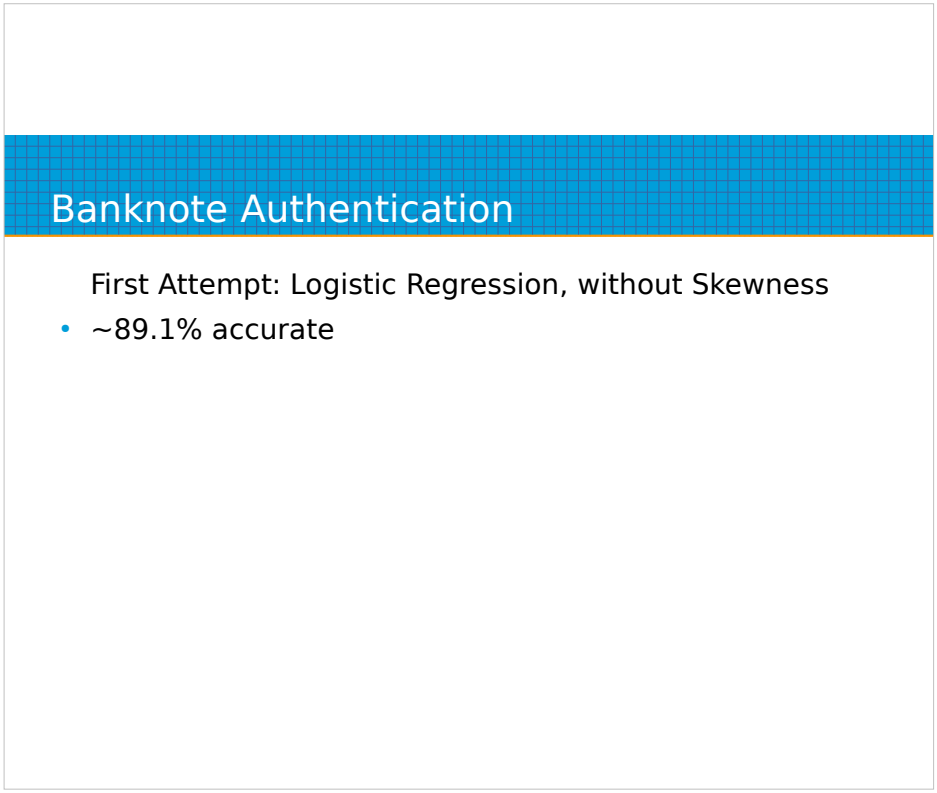
I also normalized the data as the first model I planned to try was logistic regression which can benefit from data normalization.



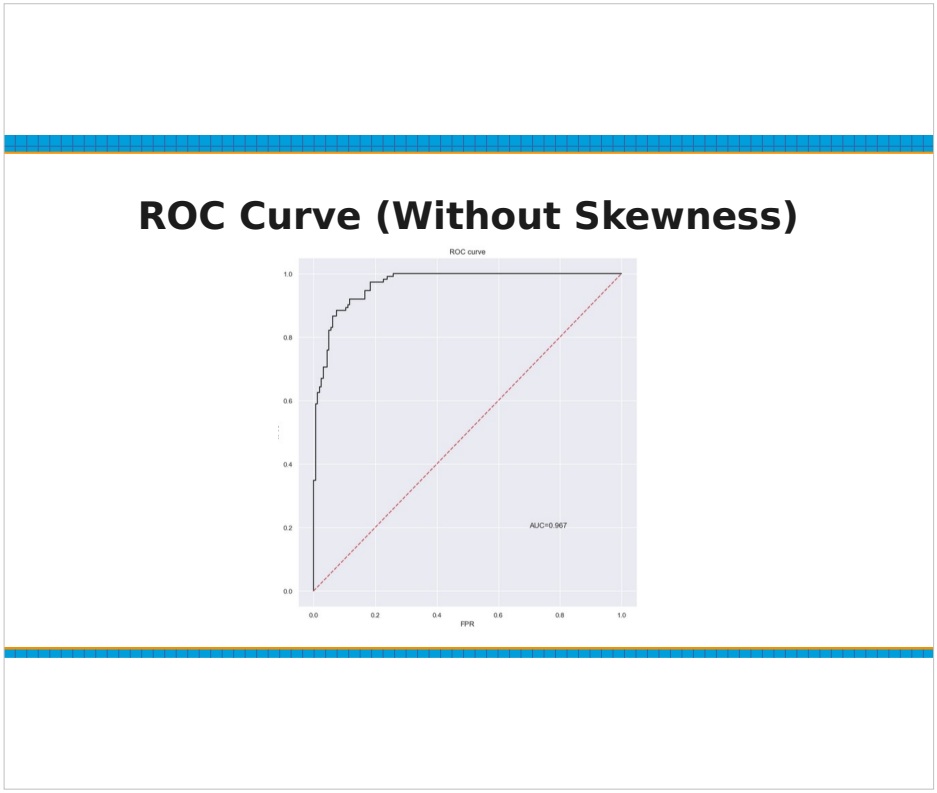
The correlation matrix shows very low correlation between skewness and curtosis and generally below .4 correlation for other values. This is expected from a cleaned data set.



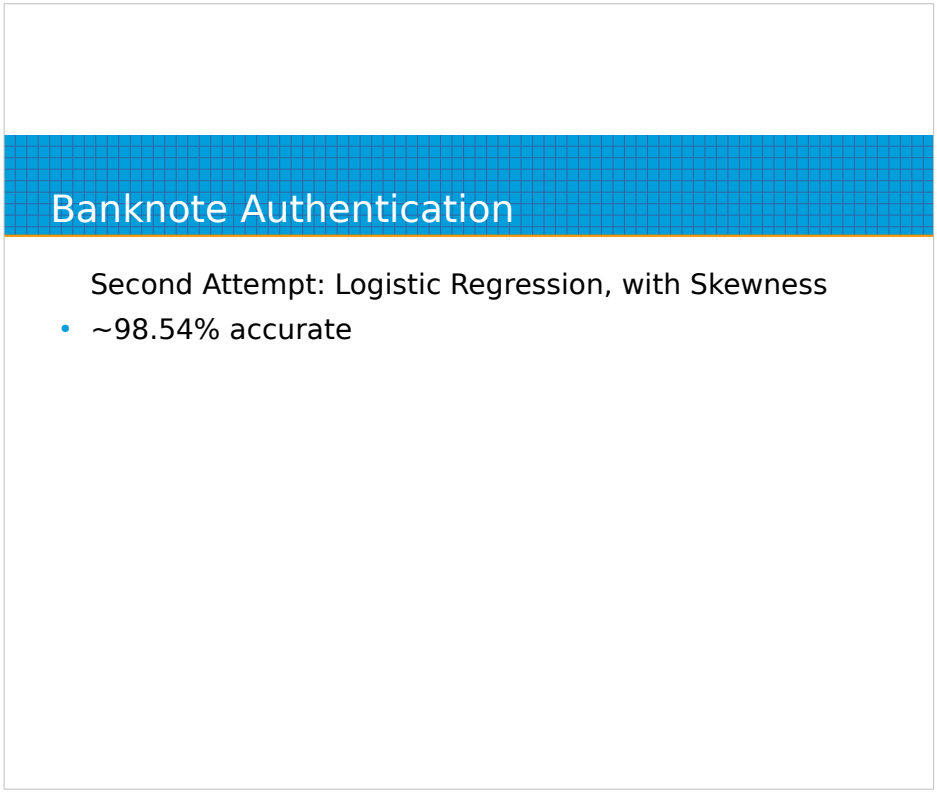
However, the pair plot shows a very narrow funnel shape between curtosis and skewness, suggesting that they are related in some way. Therefore, I thought it might be interesting to start investigating models with and without skewness to start with.



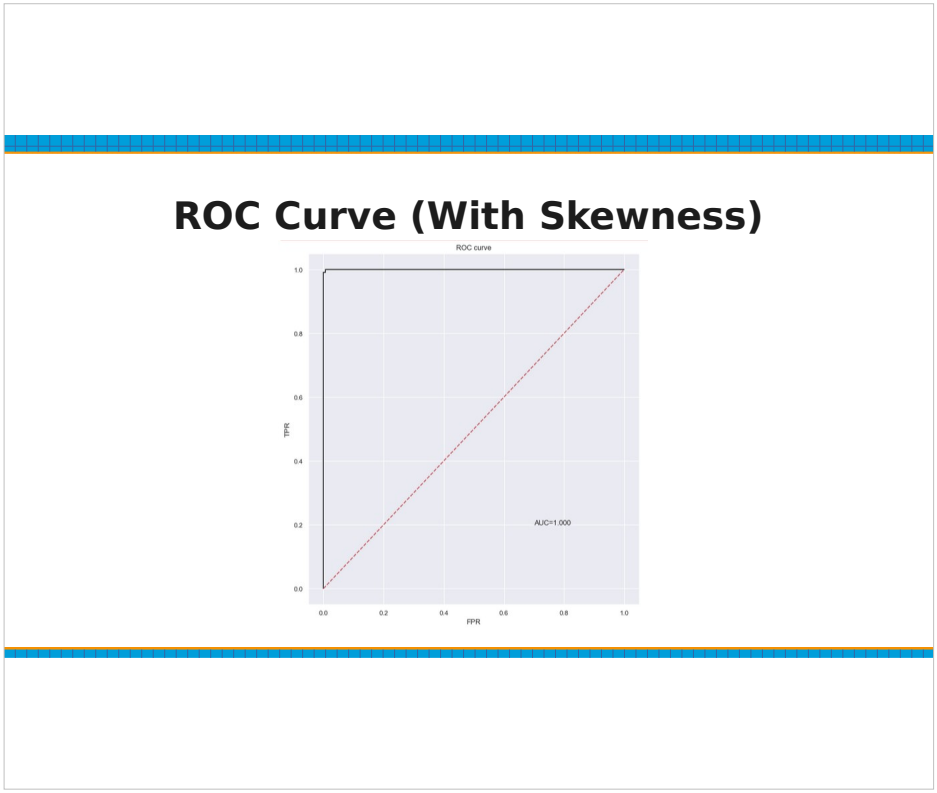
I began with a simple logistic regression model and dropped skewness from the data. This simple model already performed fairly well at 89% accuracy.



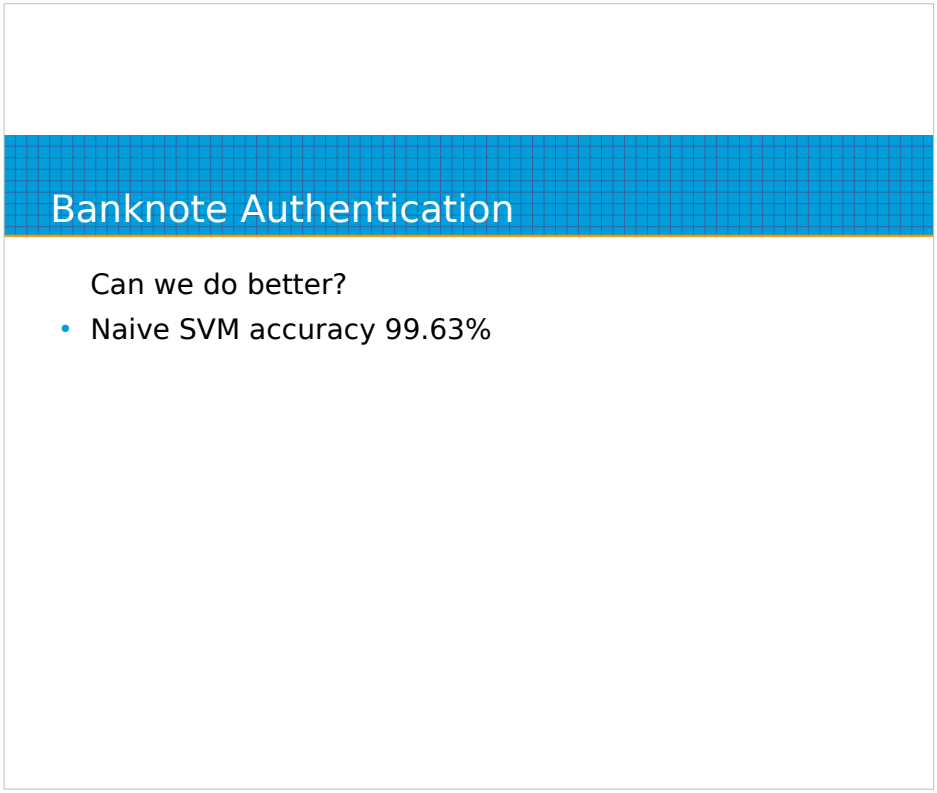
Here is the ROC curve for the logistic model without skewness. We can see that the area under the curve is quite large indicating that the model may be a strong fit.



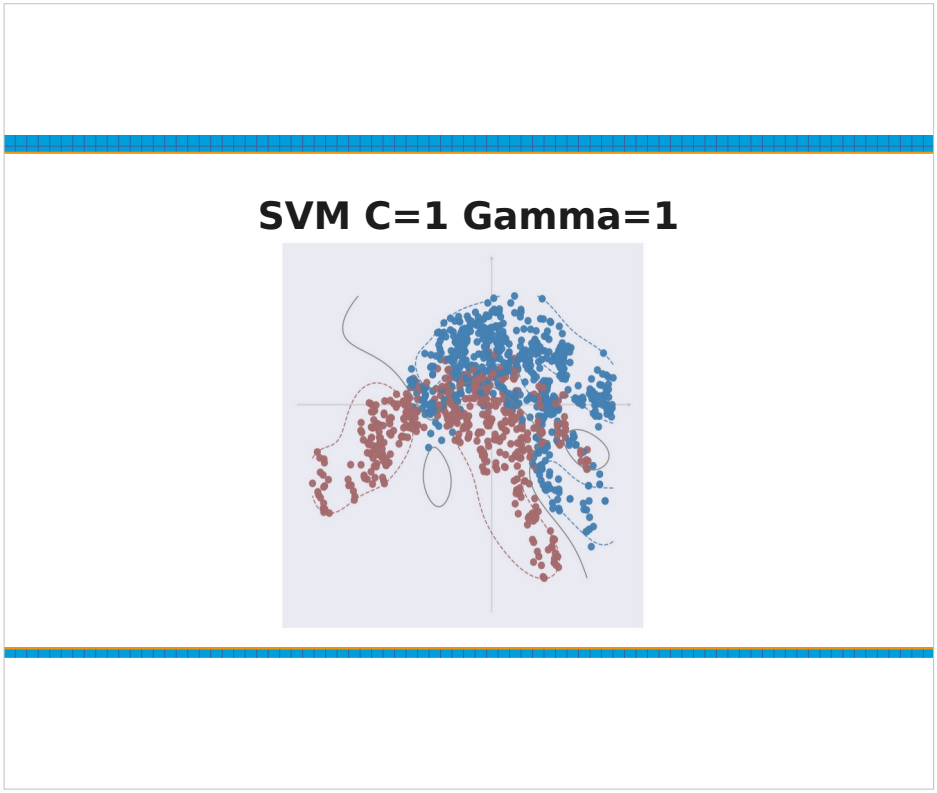
For my second attempt I wanted to compare another logistic model, this time trained on the complete data. I found that it scored a 98.5% accuracy on the test set. This is almost a 10% gain over the model without skewness, so clearly we cannot ignore skewness in the data set.



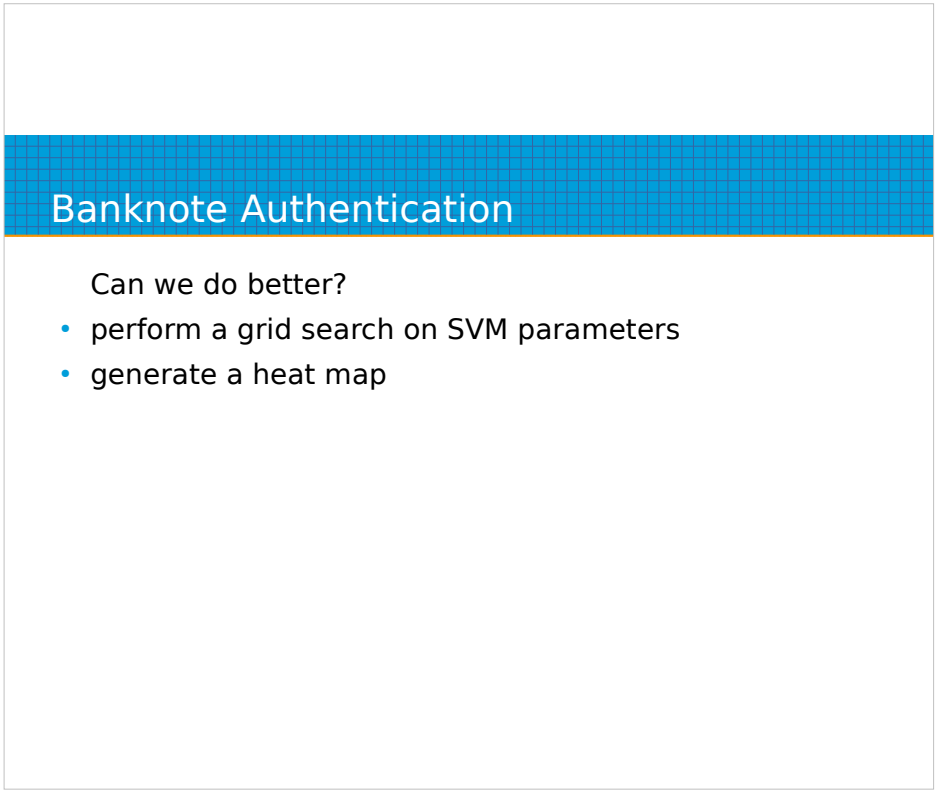
Here is the ROC curve for the second model indicating a near perfect fit with the area under the curve being almost the complete area.



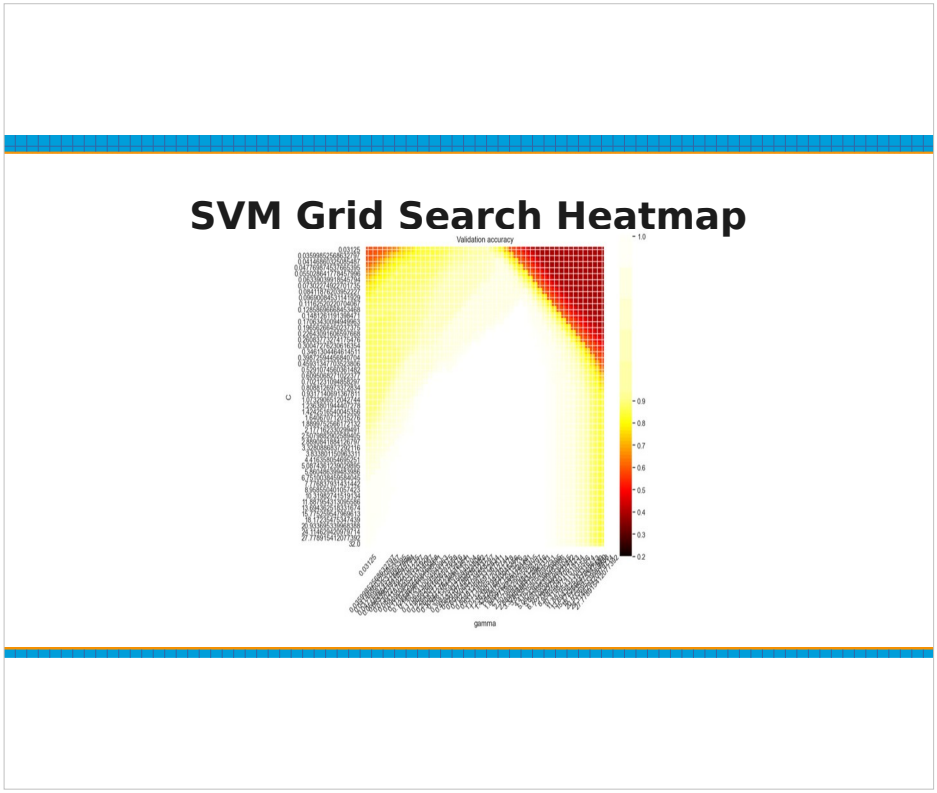
The simple logistic model already fits the data quite well at 98.5% accuracy and one might be tempted to stop there, however, in complete investigation I thought it would be valuable to try more complicated models and see if we can obtain an even better accuracy score. I began by creating a naive support vector classifier using default parameters and found that it already beat the logistic model at an amazing 99.63% accuracy on the test set.



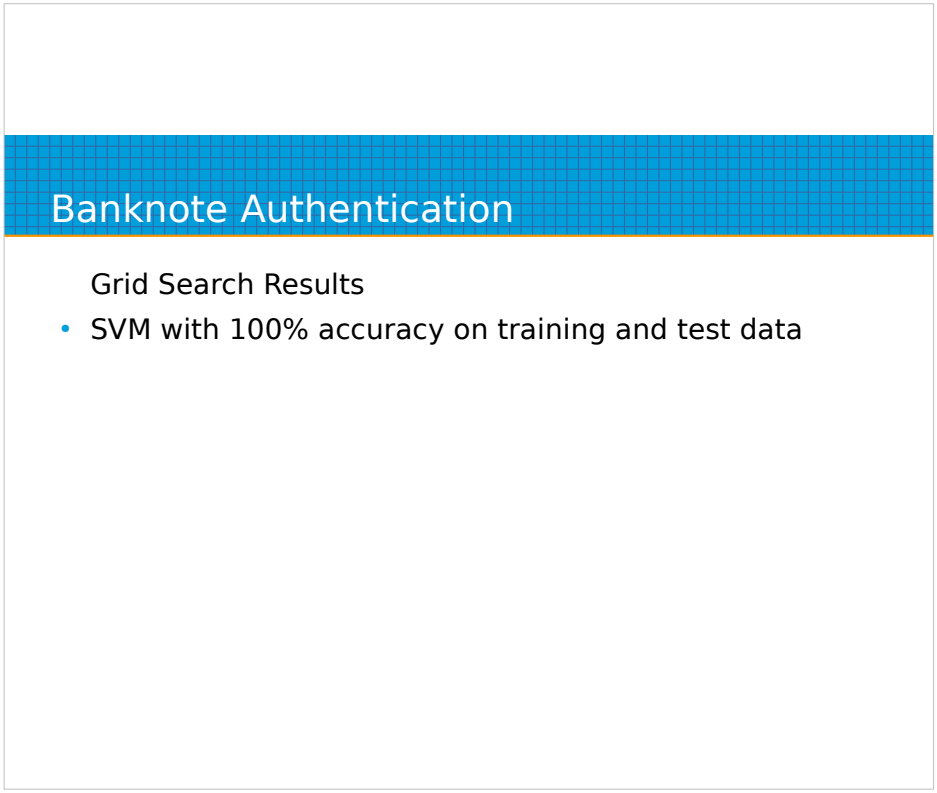
Here is a scatter graph showing the 2 dimensional fit of the naive SVM. We can see the data is closely mixed in these dimensions (I used Principal Component Analysis to reduce the data to the 2 most important dimensions for graphing). However, at 99% accuracy the data must be seperable in higher dimensions of the actual hyperplane.



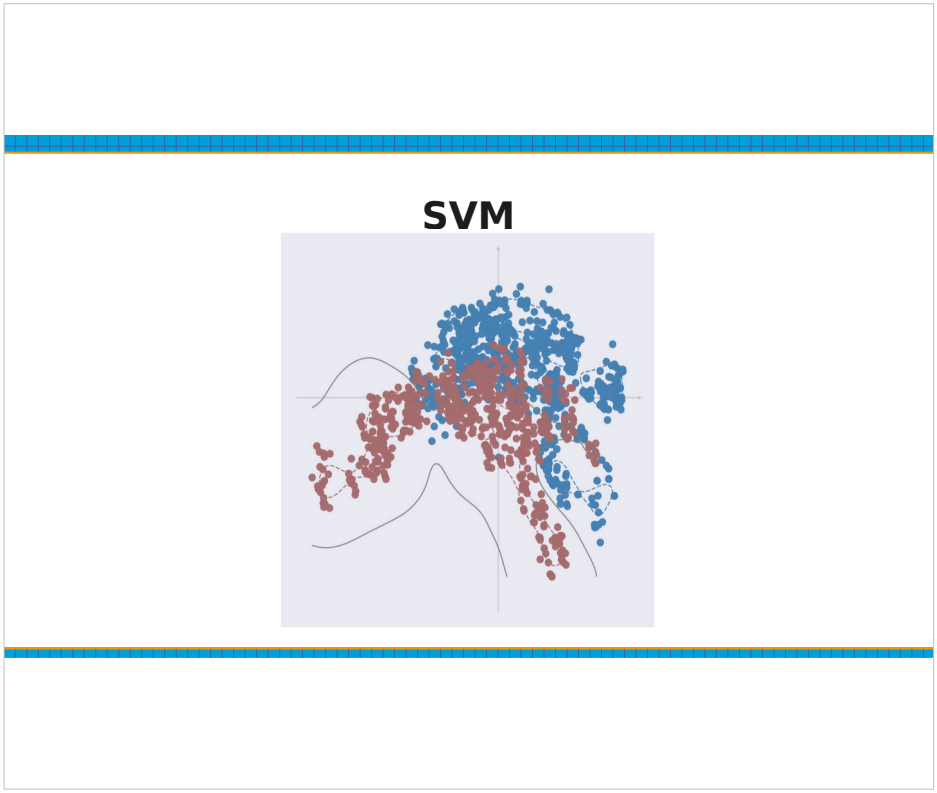
The naive SVM fitting shows promise that an SVM can out perform a simple logistic model on this data. The question remains, how high of an accuracy can we get from a trained SVM. For that I used grid search over the parameters to find an SVM with top fit. I also generated a heat map of the grids score.



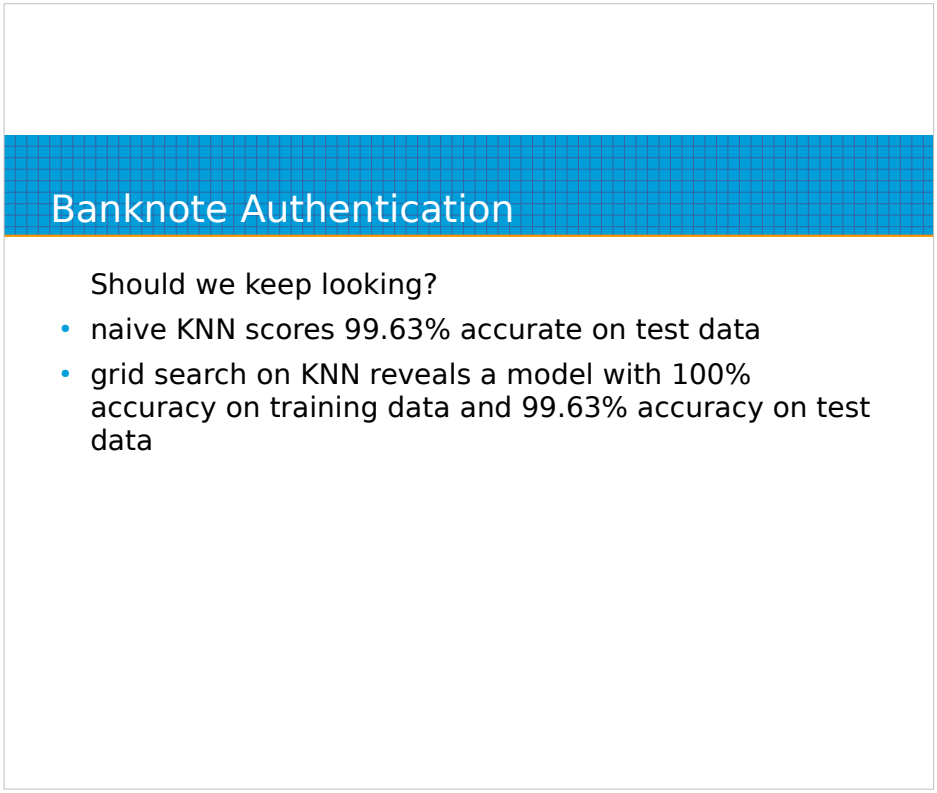
This is the heat map of the grids score. From the large portion of white, we can see that we actually have many parameters that lead to 100% accuracy. In fact it is probably too many parameter combinations to be that useful. Luckily, we can just check the most accurate model returned by the grid search.



The best SVM estimator returned by the grid search actually produces a 100% match on the training set as well as the test set. This score probably cant be beat, but we can see on the next slide that this might actually be overfit.



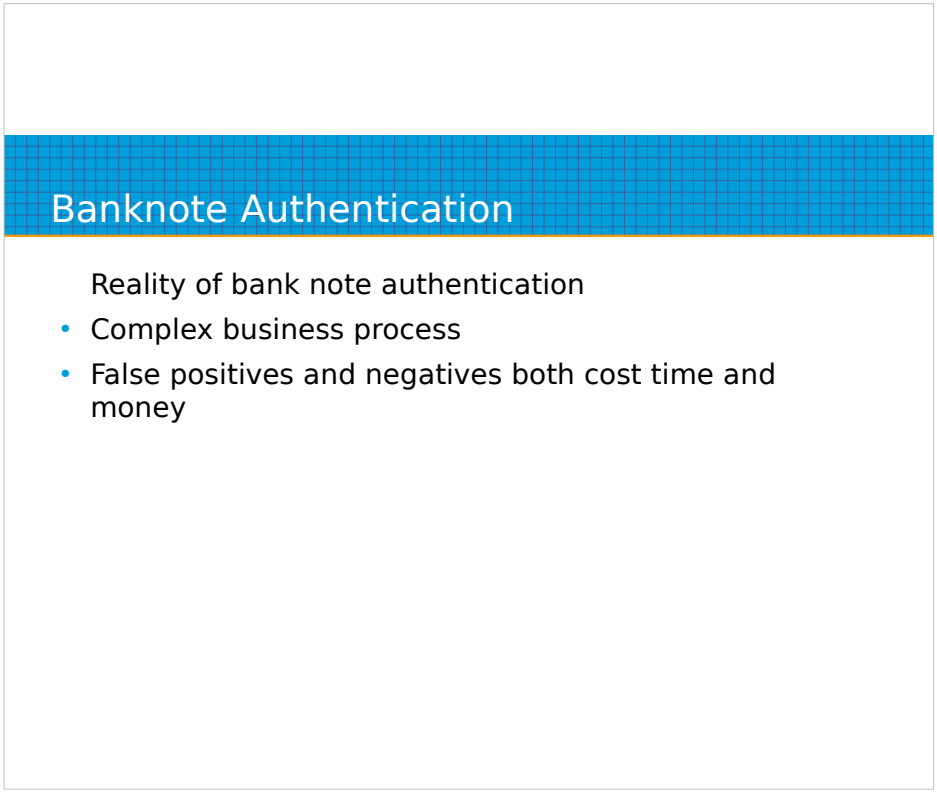
Here we see the highest accuracy SVM plotted against the same 2 dimensions as the naive SVM using an RBF kernel. This is possibly the best model for the data and clearly the data is 100% seperable with a hyperplane. However, the complexity of the model gives a low bias and high variance.



At this point a valid question is should keep looking for a better model? What could possibly beat 100% accuracy on test and training data?

I decided I should try K nearest neighbors model as it can also benefit from the data normalization done earlier and is a much simpler model than the SVM.

The naive KNN with default arguments scored 99.63% accurate on the test data, which is already promising. I decided I should perform a grid search as well. The final model reached 100% accuracy on the training data and 99.63% accuracy on the test data, approximately the same as the naive KNN.



A bank's fraud detection algorithm significantly impacts its business in several ways:

Enhanced Security and Trust: By preventing fraudulent transactions, the algorithm builds customer confidence, reinforcing the bank's reputation for safety and reliability.

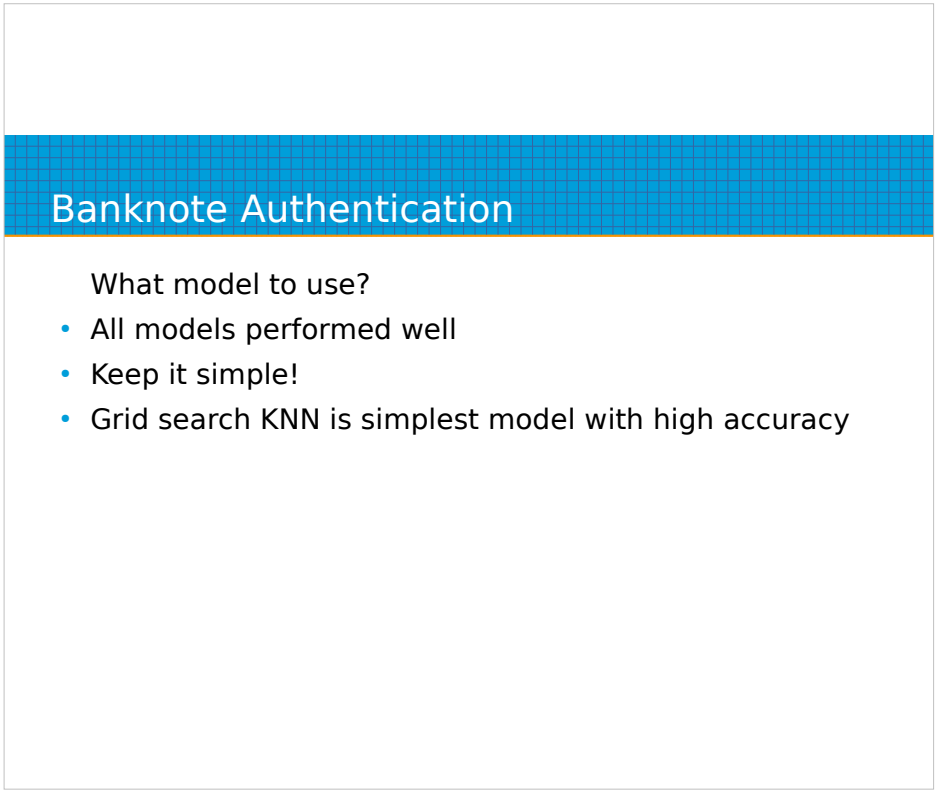
Reduced Financial Losses: Effective detection minimizes losses due to fraud, saving the bank significant amounts in reimbursement and investigation costs.

Operational Efficiency: Automated fraud detection reduces the workload on human analysts, allowing resources to focus on more complex cases or other priorities.

Regulatory Compliance: Advanced algorithms help the bank meet stringent regulatory requirements, avoiding penalties and maintaining good standing with oversight bodies.

Customer Experience: While protecting customers, overly sensitive algorithms might flag legitimate transactions as fraudulent, leading to customer frustration. Striking the right balance is crucial to maintaining a positive customer experience.

Competitive Advantage: Robust fraud detection systems can differentiate a bank from competitors, attracting more customers who prioritize security.



All models performed well on this data set. So which model makes the most sense to use?

Here I recommend using the K nearest neighbors model found via grid search. It is a simple model that's easy to understand and has high accuracy on the training set as well as the test set.

It's tempting to use the SVM which scored 100% accuracy on both the training and test set, but I believe it would be less flexible to new data than the KNN. This data set is fairly small and a real bank would need to generate a much larger data set to pick a final model.

It's also possible that the feature set is simple enough and distinguishing enough to lend itself to being easily predicted by a simple logistic model.

More testing would be needed on a larger data set to be sure. However, I think a safe place to start is with KNN.