MovieLens

Abinav Bhagam

2024-07-06

Introduction

This project - Credit Card Fraud Detection - is the the first of the two projects required to pass the *HarvardX - PH125.9x - Data Science: Capstone* course, the finale of the *Data Science Professional Certificate* program

The objective of this project is the development of a fraud identification system using the `creditcard` dataset. The dataset contains a fraction of all credit card transactions made over a two day period on September 2013 by European cardholders.

To further clarify the objective, the `creditcard` dataset is highly unbalanced. As such, many of th observations could possibly be False Positive, or rather, Legal Transactions instead of False Transactions in this scenario. The models created, will be able to detect when a False Positive is indeed, a False Positive

The ideal graph to illustrate our models would be the ROC (Receiver Operating Characteristic) curve, and to be specific, we will be measuring the AUC (area Under Curve). The AUC ranges from 0 to 1, the closer it is to 1, the better our model is preforming. We will be utilizing the aid of various machine learning algorithms to see how efficient we can push our system to become.

Data Set Source

https://www.kaggle.com/mlg-ulb/creditcardfraud (https://www.kaggle.com/mlg-ulb/creditcardfraud)

Analysis and Refinement of the Dataset

Analysis

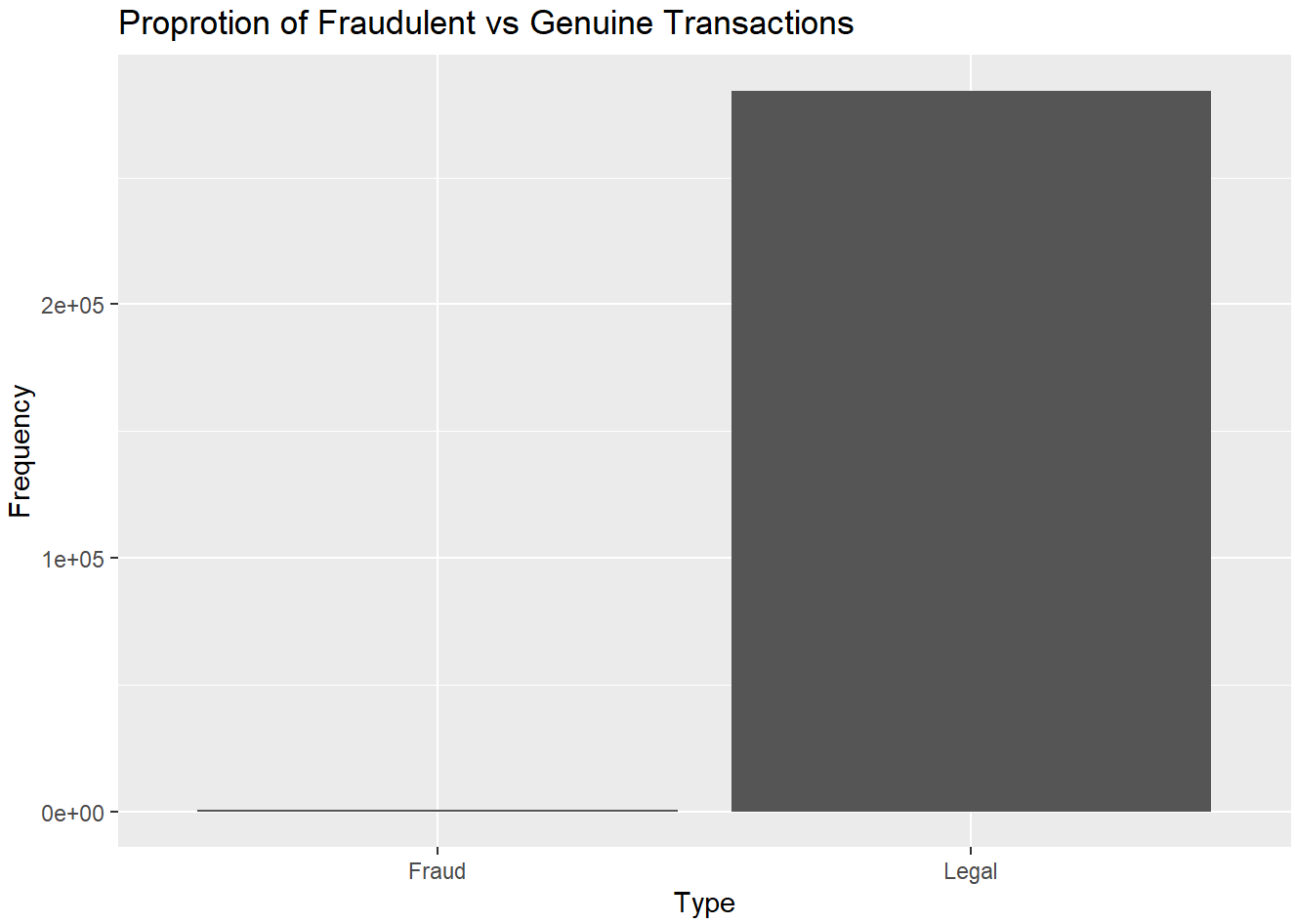
As previously mentioned, the dataset is highly unbalanced. Of the 284,807 transactions recorded only 492 are fraudulent. A scant 0.173%.

The dataset contains only numerical input variables as a result of a PCA transformation. Due to confidentiality issues we, unfortunately, cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only variables to not been

transformed with PCA are ‘Time’, ‘Amount’ and ‘Class’.

Dataset Imbalance

One of the three variables to not have been altered by PCA is ‘Class’. This variable determines whether or not the transaction was fraudulent. The 492 fraudulent transactions all have a ‘Class’ of 1, the remaining 284,315 have a 0.



As one can clearly see, the proportion of fraudulent transactions is essentially negligible compared to the genuine transacations. Which is pretty good news! Nice to know our money is safe.

So what does the dataset actually look like?

First 5 rows of the dataset

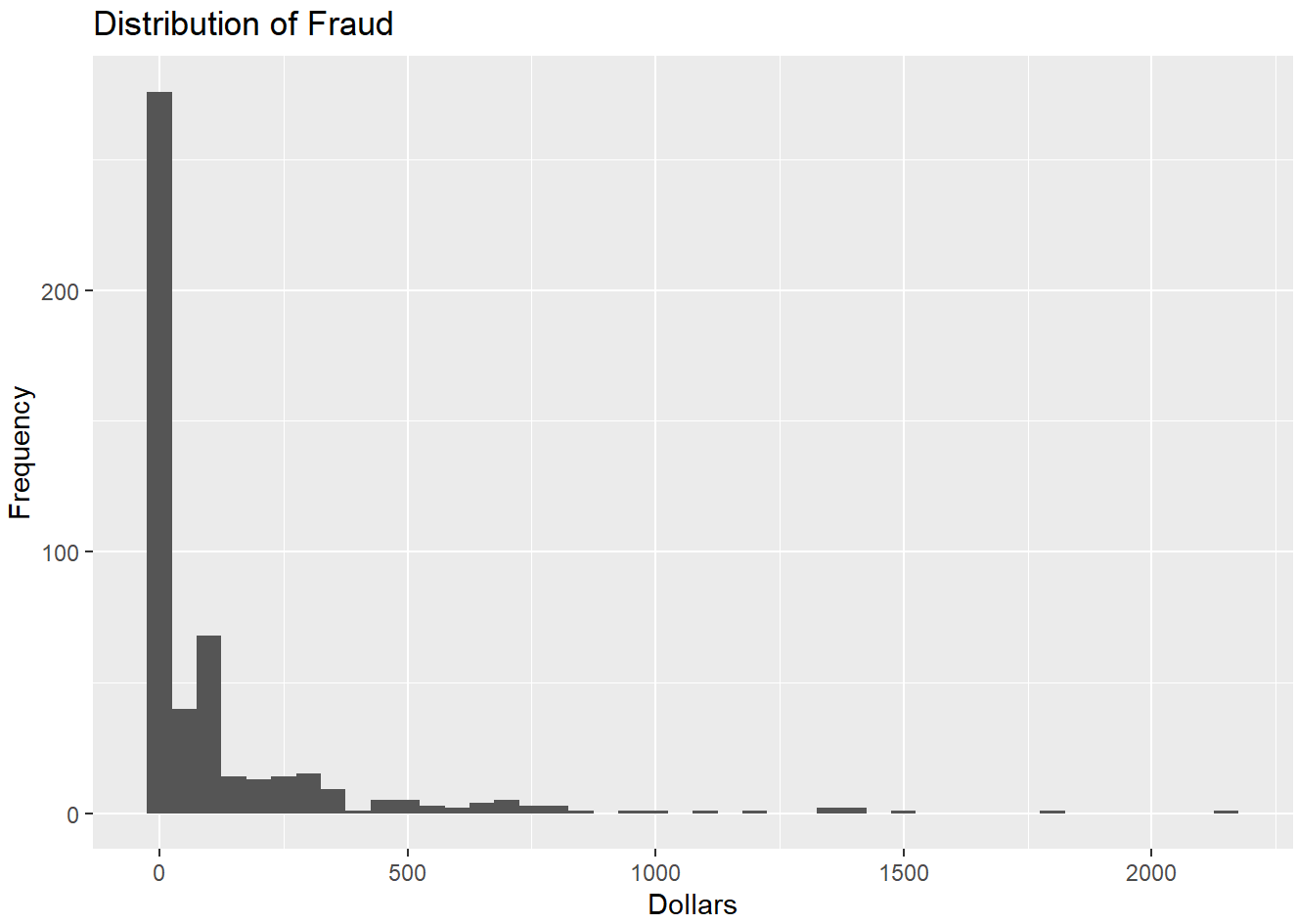
	Time	V1	V2	V3	V28	Amount	Class	Class2
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<chr>
1	0	-1.3598071	-0.07278117	2.5363467	-0.02105305	149.62	0	Legal
2	0	1.1918571	0.26615071	0.1664801	0.01472417	2.69	0	Legal
3	1	-1.3583541	-1.34016307	1.7732093	-0.05975184	378.66	0	Legal
4	1	-0.9662717	-0.18522601	1.7929933	0.06145763	123.50	0	Legal
5	2	-1.1582331	0.87773675	1.5487178	0.21515315	69.99	0	Legal

5 rows

Most of these variables don't offer immediately relevant information. The only variables that do are the ones unaffected by the PCA transformation. So let's compare fraudulent transactions against those unaafected varibales.

First up, Fraud vs Amount

Distribution of Fraud



The majority of the fraudulent transactions appear to be of minuscule amounts ~ $1. Makes sense, large scams tends to attract law enforcement.

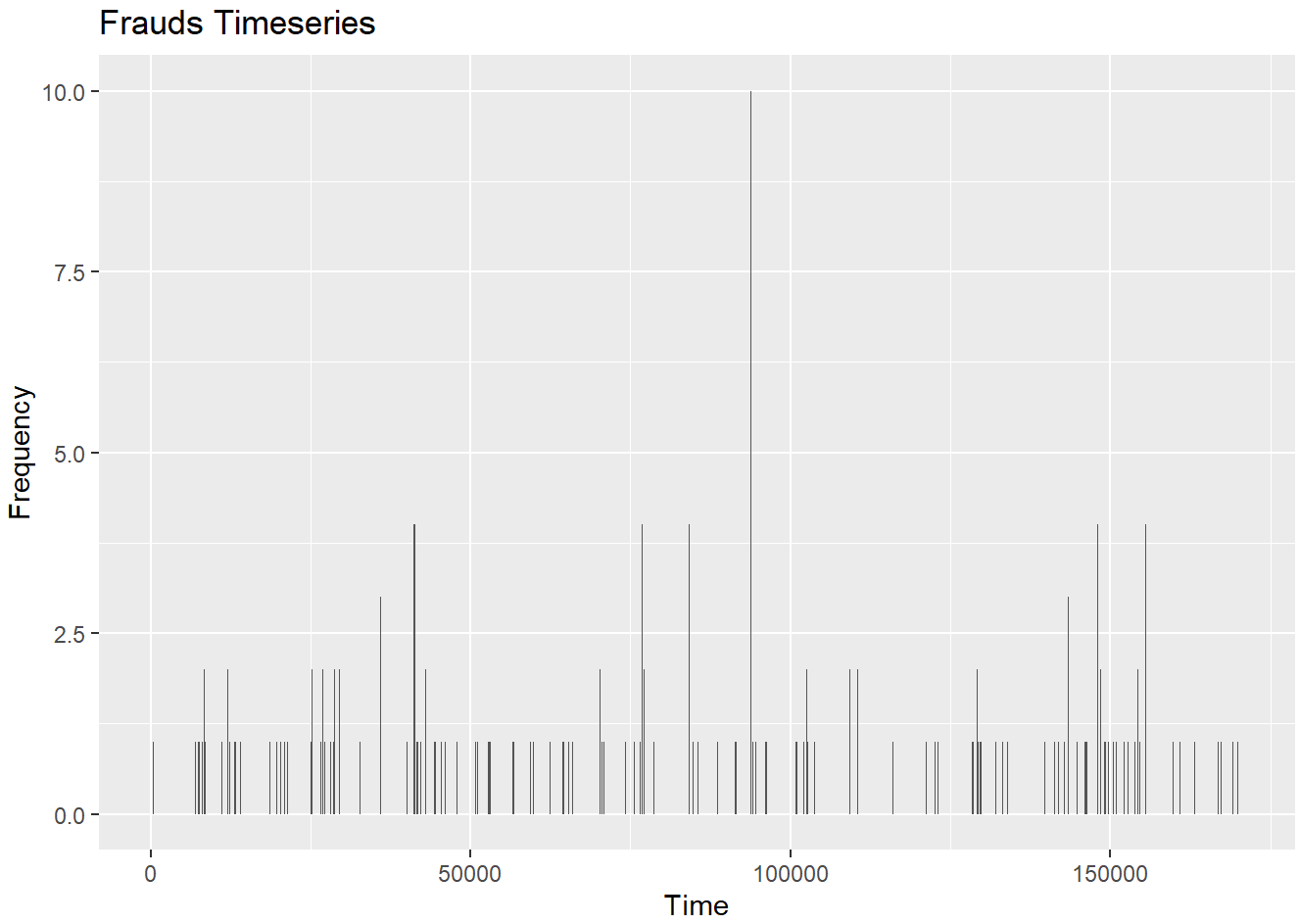
Amount <dbl>	count <int>
1.00	113
0.00	27
99.99	27
0.76	17
0.77	10

5 rows

None of the top 5 amounts being scammed are above a dollar.

We've compared frauds vs amount, but what about fraud vs time? Does time play a role in when fraudulent activity occurs?

Fraud Timeseries



There doesn't seem to be much a correlation here.

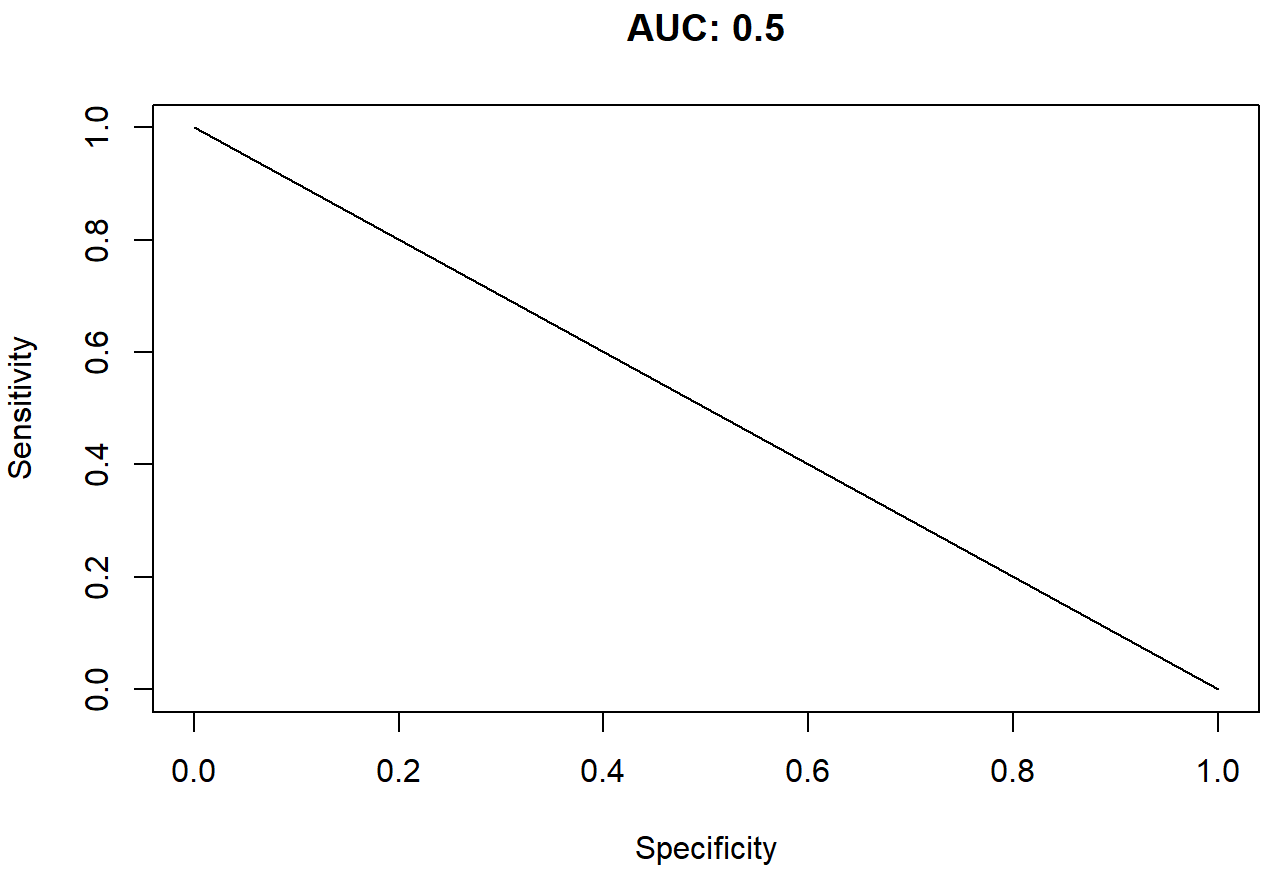
Refinement

Before we move onto model construction, it will be neccesarry to construct training, test and validation datasets.

Analysis - Model Construction

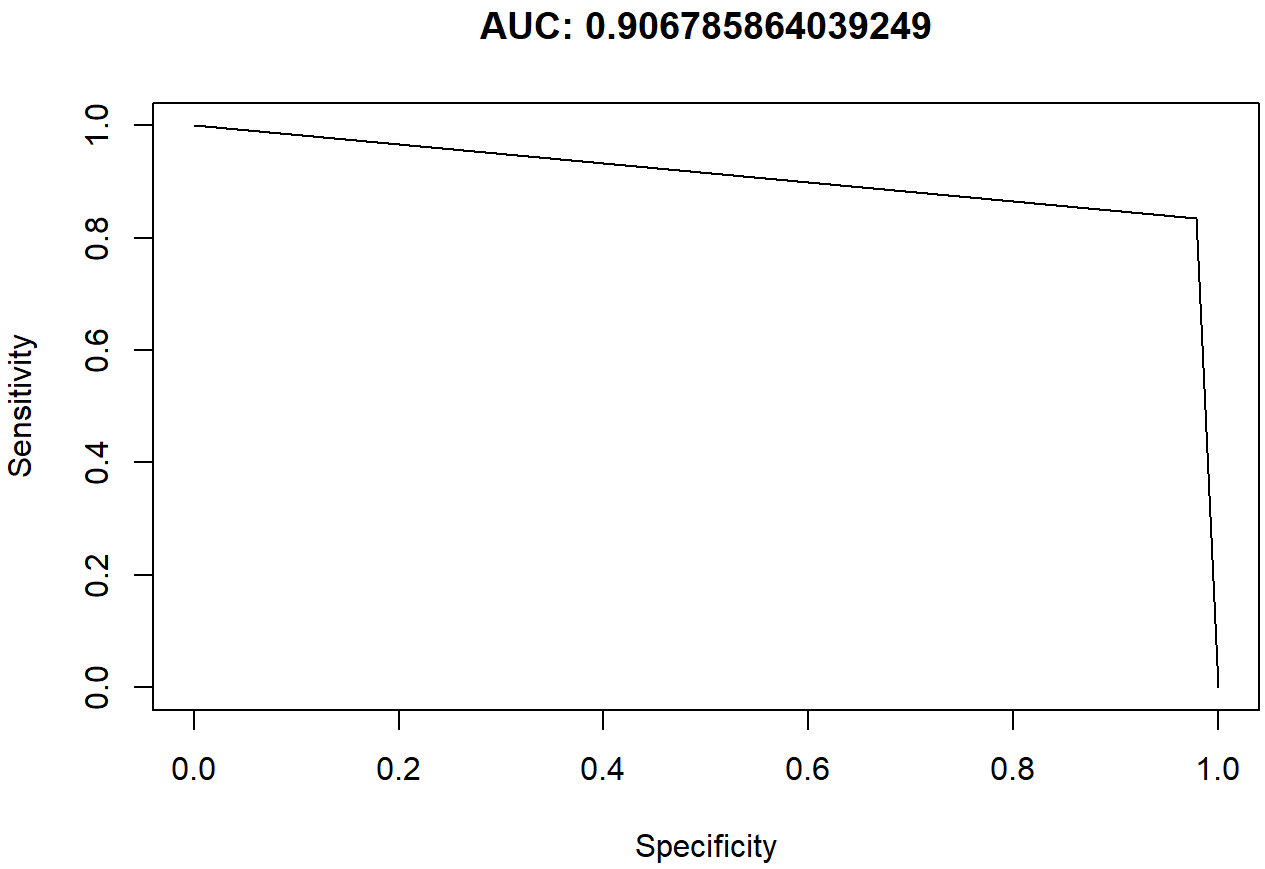
Baseline

The most basic form of our model. It predicts what we already know. That all ‘Genuine’ transactions are indeed, ‘Genuine’. As one may expect, we receive an AUC of **0.5**.



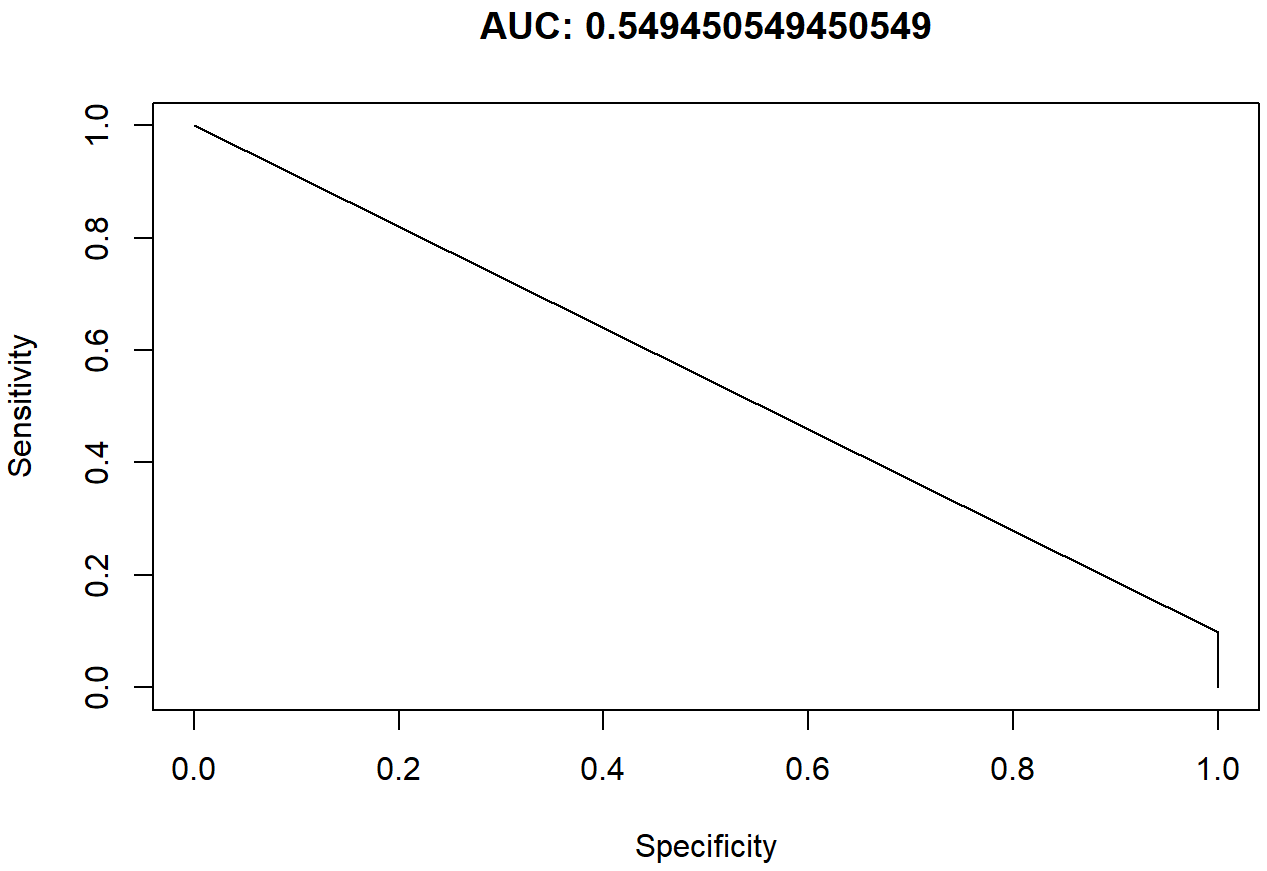
Bayes

The first evolution of our model will be the incorporation the Bayesian Classifier. Our AUC is boosted by quite a bit, all the way to **0.907**.



KNN - K-Nearest Neighbors

Can a KNN model where k = 5 go beyond the Bayesian Classifier? Evidently not, for the AUC tumbles all the way back down to **0.549**, barely above the baseline model.

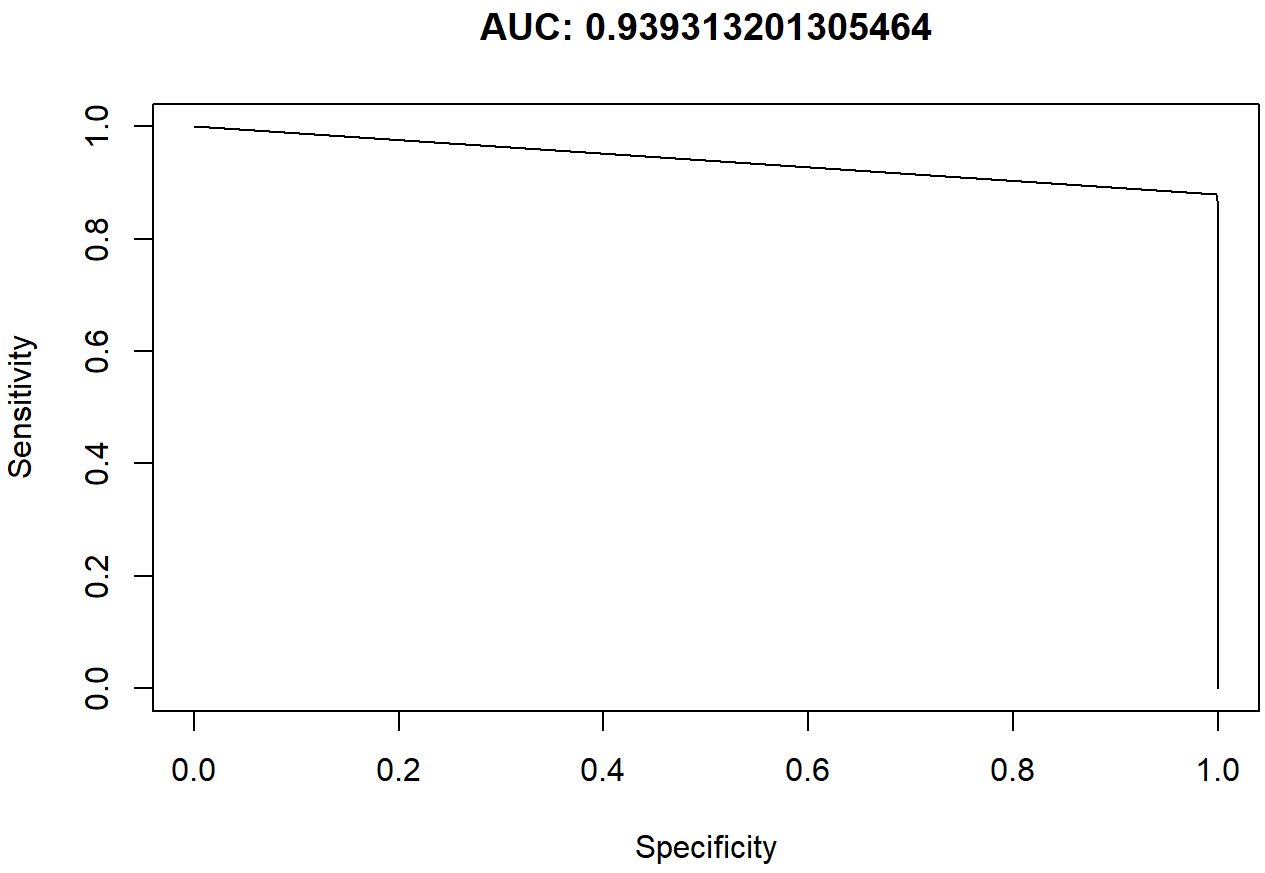


XGBoost

Perhaps it's better to move onwards to much more powerful models. I've attempted to use SVM, Random Forest, GBM and LightGBM to see if the AUC could be improved upon, but the amount of time required to train those models was beyond reasonable.

As such, I've elected to use on the strongest ML algorithms on Kaggle.com, XGBoost. And as expected of such a powerful model, it has blown the AUC through the roof. A new high of **0.939** has been achieved.

```
## [1]  test-aucpr:0.727427 validation-aucpr:0.703413
## Multiple eval metrics are present. Will use validation_aucpr for early stopping.
## Will train until validation_aucpr hasn't improved in 4 rounds.
##
## Stopping. Best iteration:
## [5]  test-aucpr:0.757962 validation-aucpr:0.752475
```



Results

These are the results for all the models built, trained, tested and validated.

Model <chr>	AUC <dbl>
Baseline	0.5000000
Bayes Model	0.9067859
KNN Model	0.5494505
XGBoost	0.9393132

4 rows

Conclusion

The ensemble method has once again confirmed itself as the undisputed champion of algorithmic modelling.

An XGBoost model can achieve an excellent AUR result of **0.939** in a fraction of the time it took the KNN model to achiev a AUR of **0.549**.