Prevent Scam Transactions

Roja Kasula

March 01, 2022

Contents

Length	Columns
284807	31

1 Executive Summary

Customers are not charged for products that they did not purchase, if the credit card companies are able to recognize fraudulent payment transactions. Data set contains transactions made by credit cards.Due to imbalancing nature of the data, many observations could be predicted as False Negative, in this case legal transactions instead of fraudulent transactions.

For example, a model that predict always **0** (Legal) can achieve an accuracy of **99.8**. For that reason, the metric used for measuring the score is the **Area Under The Precision-Recall Curve (AUCPR)** instead of the traditional AUC curve. Expected result is an AUCPR at least greater than **0.85**.

For classifying fraud detection, there are several trained algorithms such as Naive Bayes Classifier, KNN, SVM, Random Forest, GBM, XGBoost and LightGBM.To meet the goal in this analysis, using a XGBoost Model , which is capable of an AUCPR of **0.8623** .

2 Data Analysis

2.1 The Dataset

This data set presents transactions that occurred in two days, where we have **492 frauds** out of **284,807 transactions**. It is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. Due to legal ramifications detail data is not published . The data set contains only numerical input variables which are the result of a PCA transformation. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'.

Dataset Source

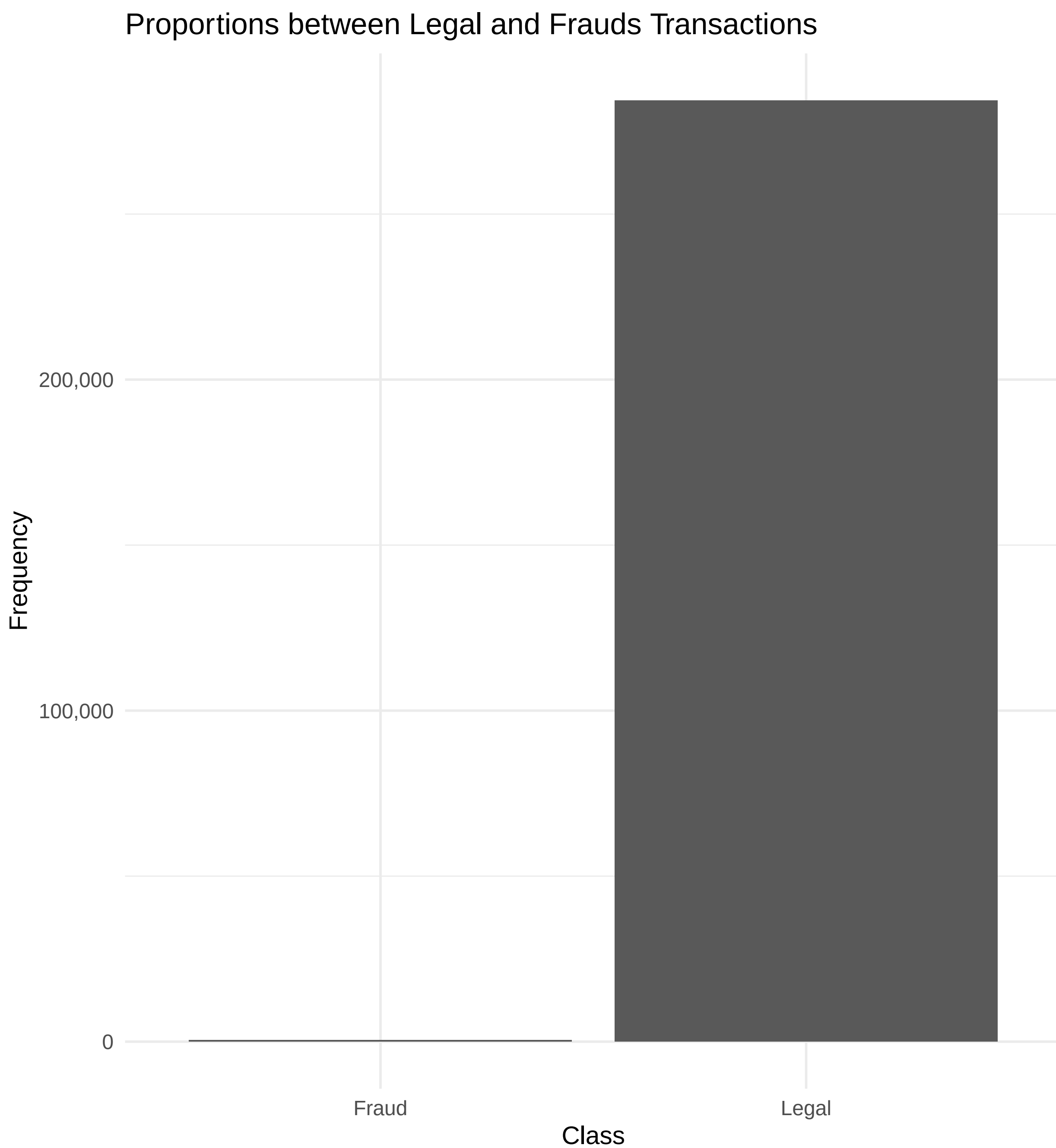
https://www.kaggle.com/mlg-ulb/creditcardfraud

Dimensions

Imbalanced Dataset

This is imbalanced data set. It means that there are few rows that represent a class. In this case, only **492** transactions are frauds, represented by **1** and **284315** are not, represented by **0**.

Class	Count
0	284315
1	492



	Missing Values
Time	0
V1	0
V2	0
V3	0
V4	0
V5	0
V6	0
V7	0
V8	0
V9	0
V10	0
V11	0
V12	0
V13	0
V14	0
V15	0
V16	0
V17	0
V18	0
V19	0
V20	0
V21	0
V22	0
V23	0
V24	0
V25	0
V26	0
V27	0
V28	0
Amount	0
Class	0

Missing Values

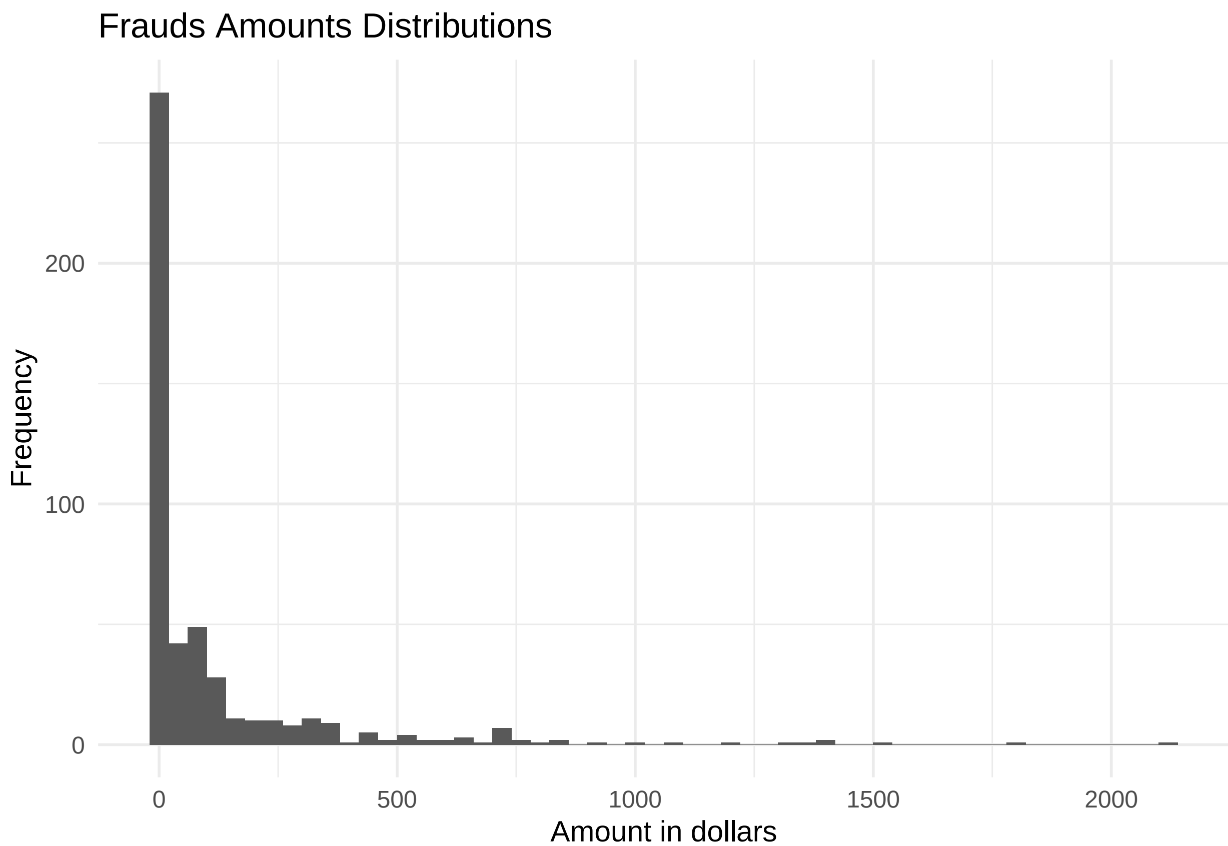
Missing values does not exist in this data frame as per the table .

Top 10 rows of `creditcard` dataset

Time	V1	V2	V3	V4	V5	V28	Amount	Class
0	-1.3598071	-0.0727812	2.5363467	1.3781552	-0.3383208	-0.0210531	149.62	0
0	1.1918571	0.2661507	0.1664801	0.4481541	0.0600176	0.0147242	2.69	0
1	-1.3583541	-1.3401631	1.7732093	0.3797796	-0.5031981	-0.0597518	378.66	0
1	-0.9662717	-0.1852260	1.7929933	-0.8632913	-0.0103089	0.0614576	123.50	0
2	-1.1582331	0.8777368	1.5487178	0.4030339	-0.4071934	0.2151531	69.99	0
2	-0.4259659	0.9605230	1.1411093	-0.1682521	0.4209869	0.0810803	3.67	0
4	1.2296576	0.1410035	0.0453708	1.2026127	0.1918810	0.0051678	4.99	0
7	-0.6442694	1.4179635	1.0743804	-0.4921990	0.9489341	-1.0853392	40.80	0
7	-0.8942861	0.2861572	-0.1131922	-0.2715261	2.6695987	0.1424043	93.20	0
9	-0.3382618	1.1195934	1.0443666	-0.2221873	0.4993608	0.0830756	3.68	0

Frauds Amount Distributions

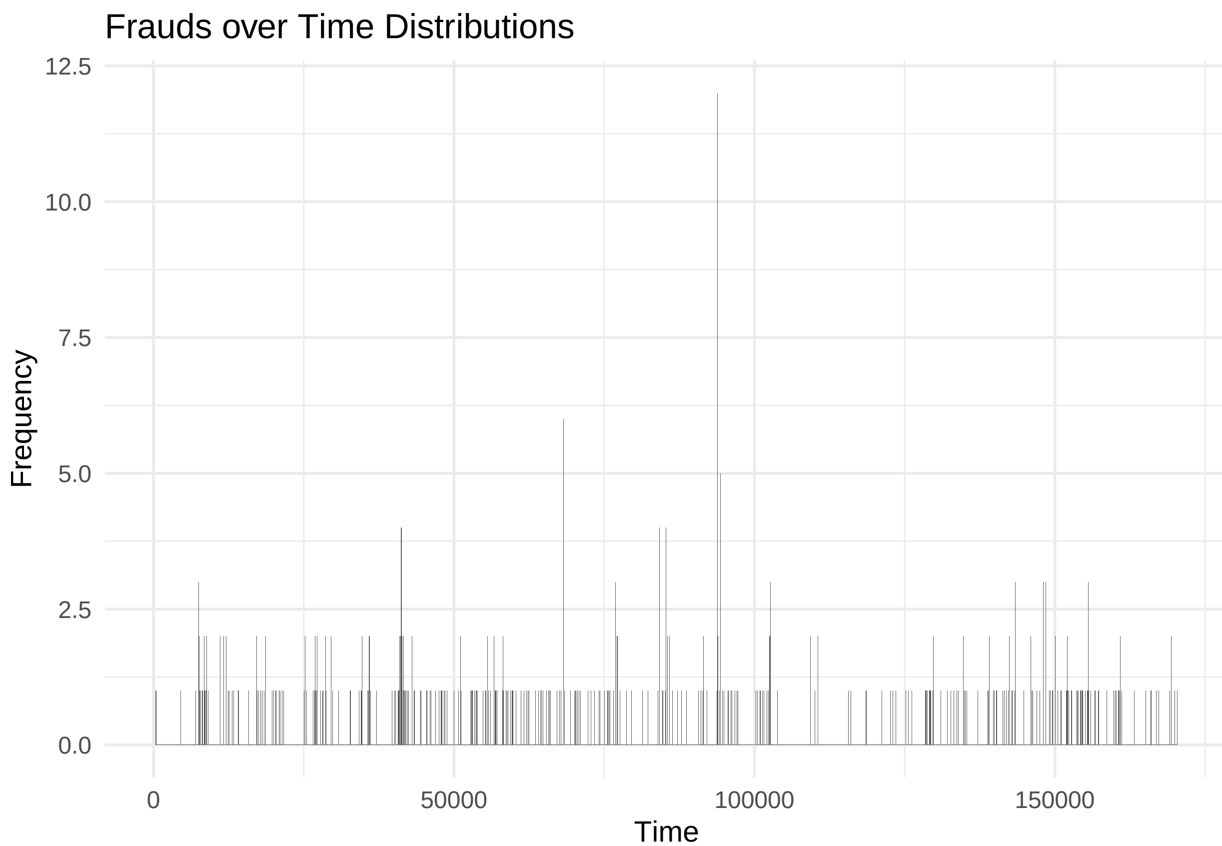
Small amount of money, less or equal of one dollar are scammed more frequently.



Amount	count
1.00	113
0.00	27
99.99	27
0.76	17
0.77	10
0.01	5
2.00	4
3.79	4
0.68	3
1.10	3

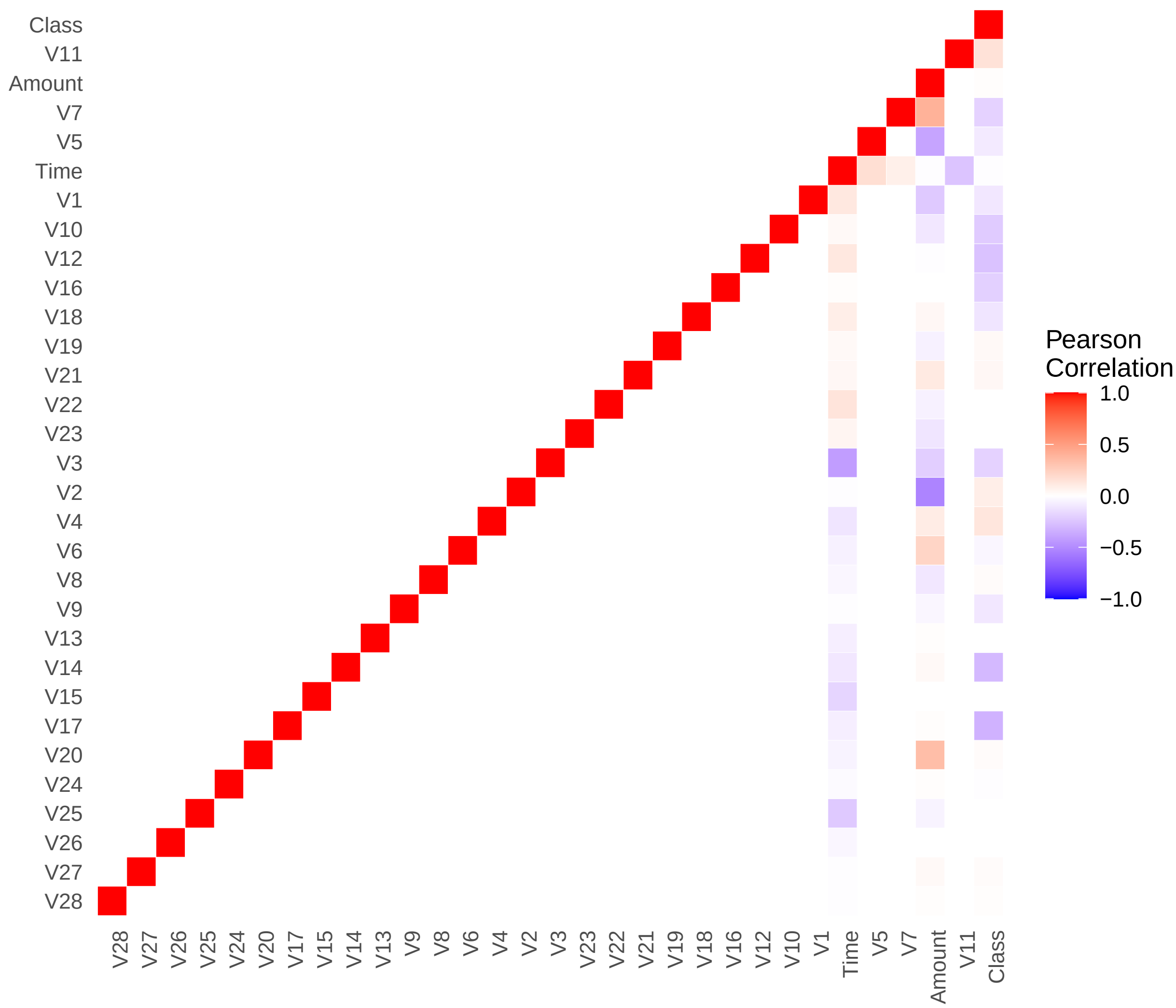
Frauds over Time Distribution

There is no correlation between `time` and frauds. A fraud can happen anytime. It seems not particularly useful for the modelling phase. The correlation matrix below, confirms this assumption.



Time	count
68207	6
84204	4
85285	4
93853	4
93860	4
93879	4
94362	4
148053	2
406	1
472	1

Correlations between each variables



Class
V11
Amount
V7
V5
Time
V1
V10
V12
V16
V18
V19
V21
V22
V23
V3
V2
V4
V6
V8
V9
V13
V14
V15
V17
V20
V24
V25
V26
V27
V28
Pearson Correlation
1.0
0.5
0.0
−0.5
−1.0

3 Data Pre-Processing and Feature Engineering

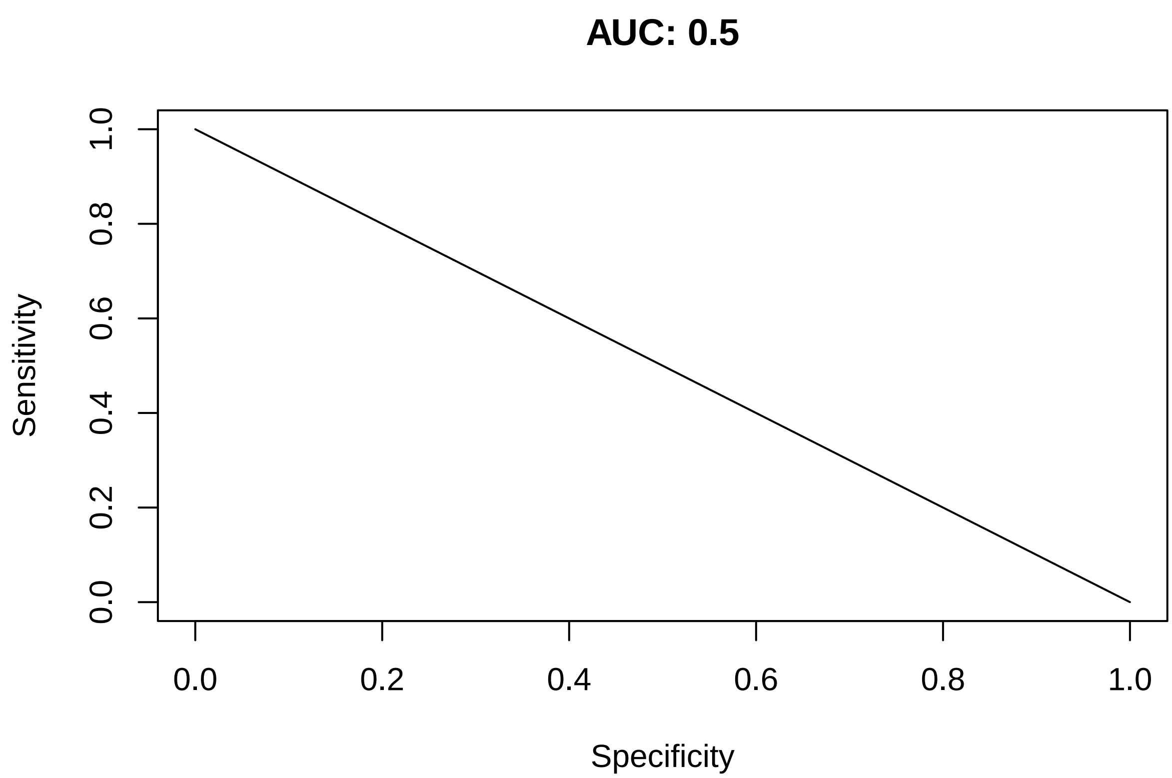
Before continuing to build models, doing data pre-processing:

1. Remove the "Time" column from the data set. It isn't useful.
2. Split the data set into train, test, cv data set.

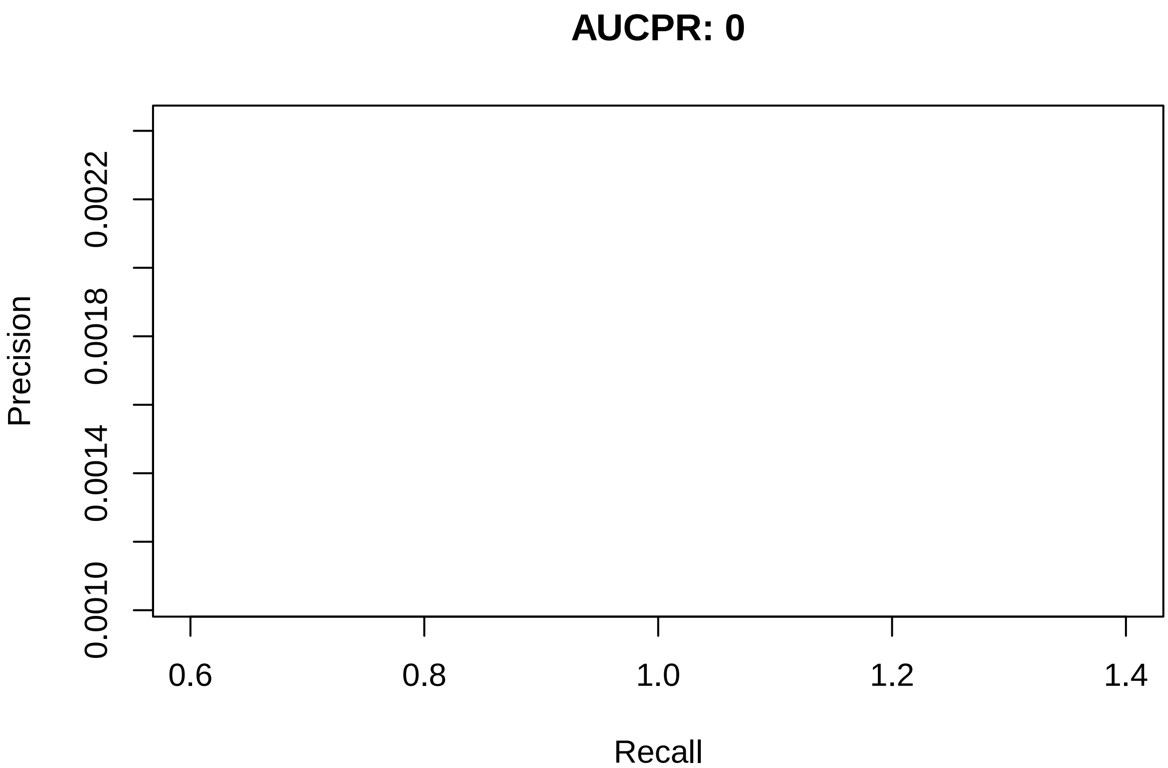
4 Explore Machine Learning Models

4.1 Naive Baseline Algorithm

Predicting always "Legal" transaction can achieve an impressive accuracy of **99.8** and an AUC of **0.92**. Because the recall and precision are **0**, it is impossible to compute the AUCPR, so that is **0**.

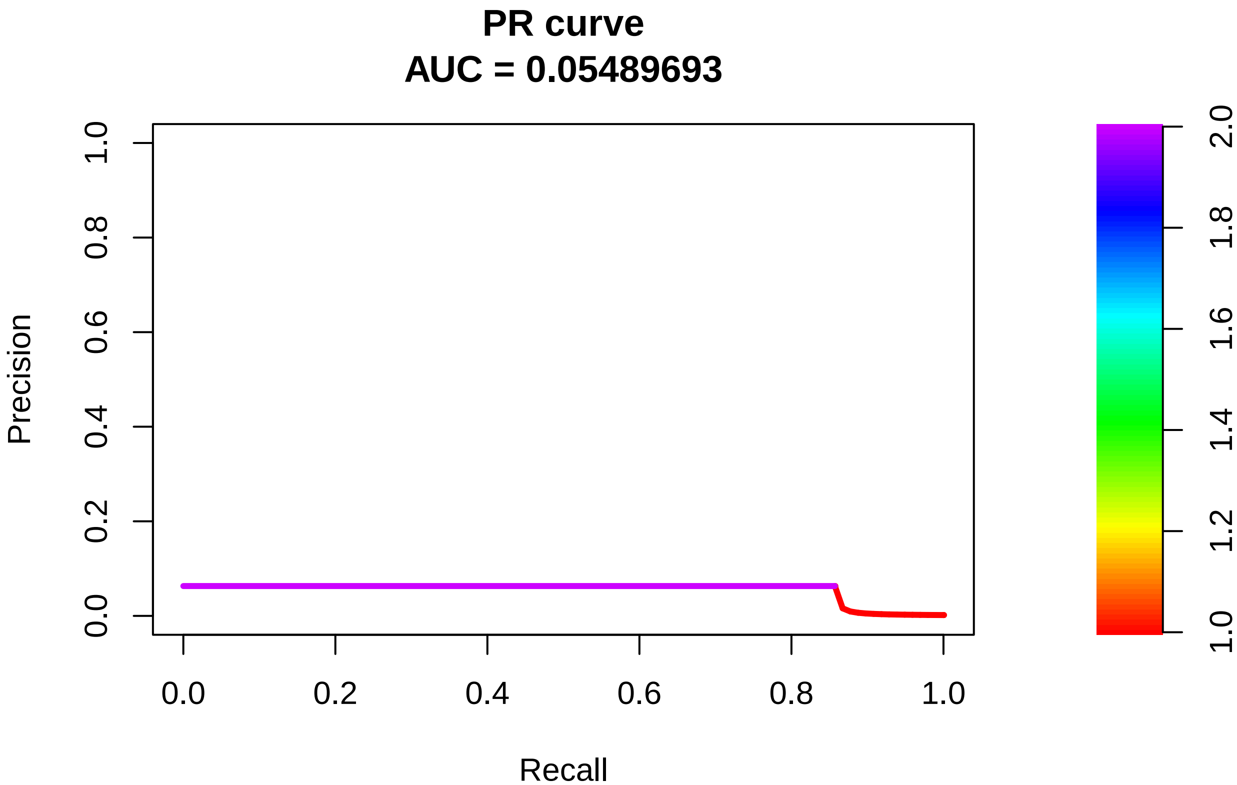


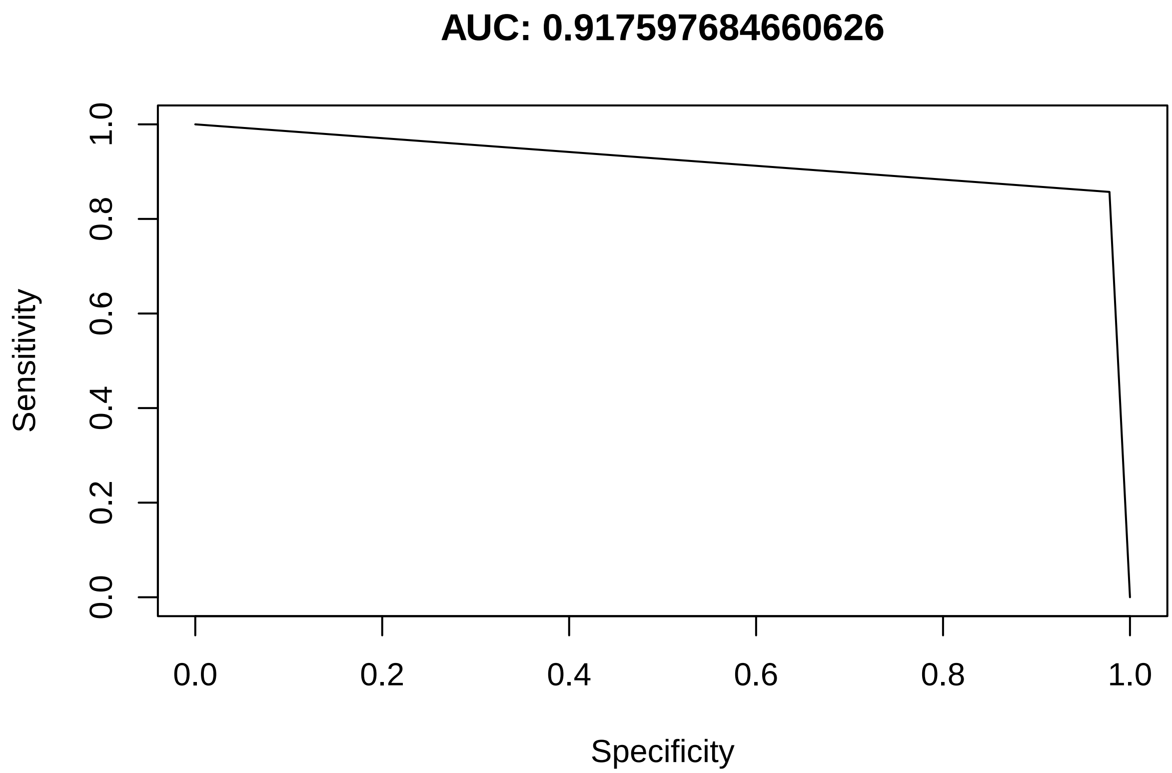
Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5	0



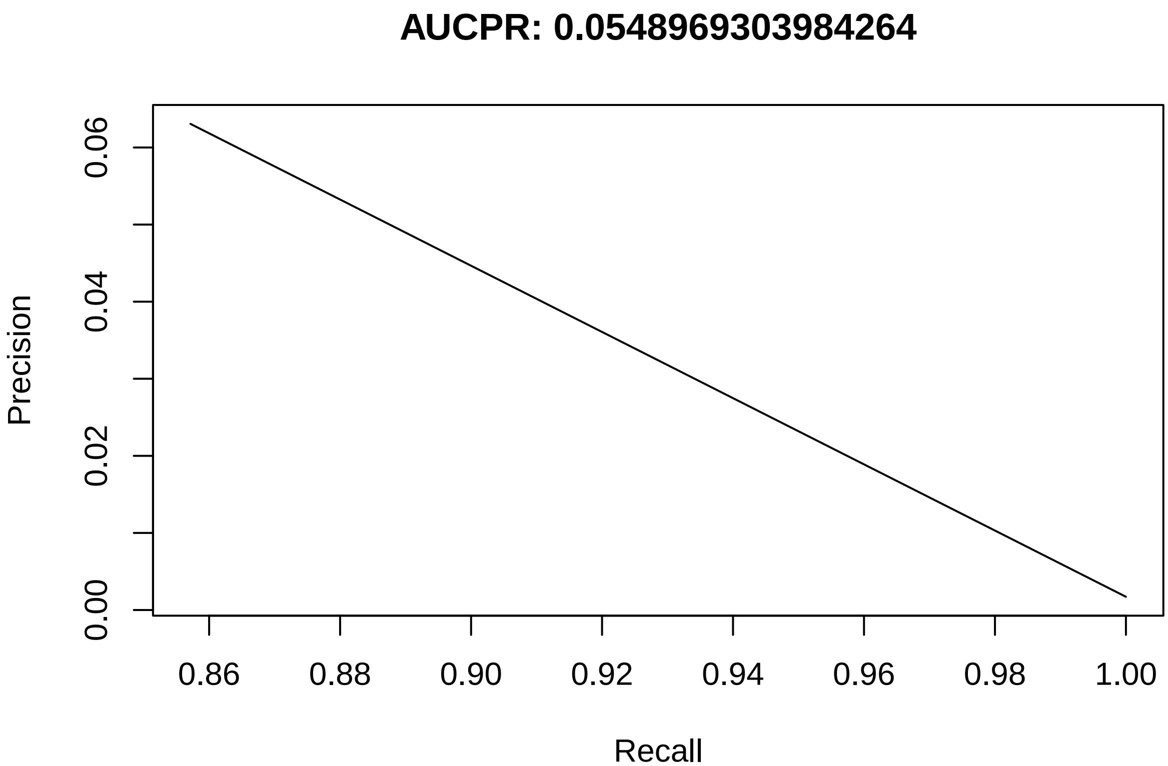
4.2 Naive Bayes

A step forward is building a Naive Bayes Classifier. The performance improve a little bit: AUC is **0.92** and finally the there is an AUCPR of **0.05**. output is not expected way , according to the expected metric and it is easy to improve.



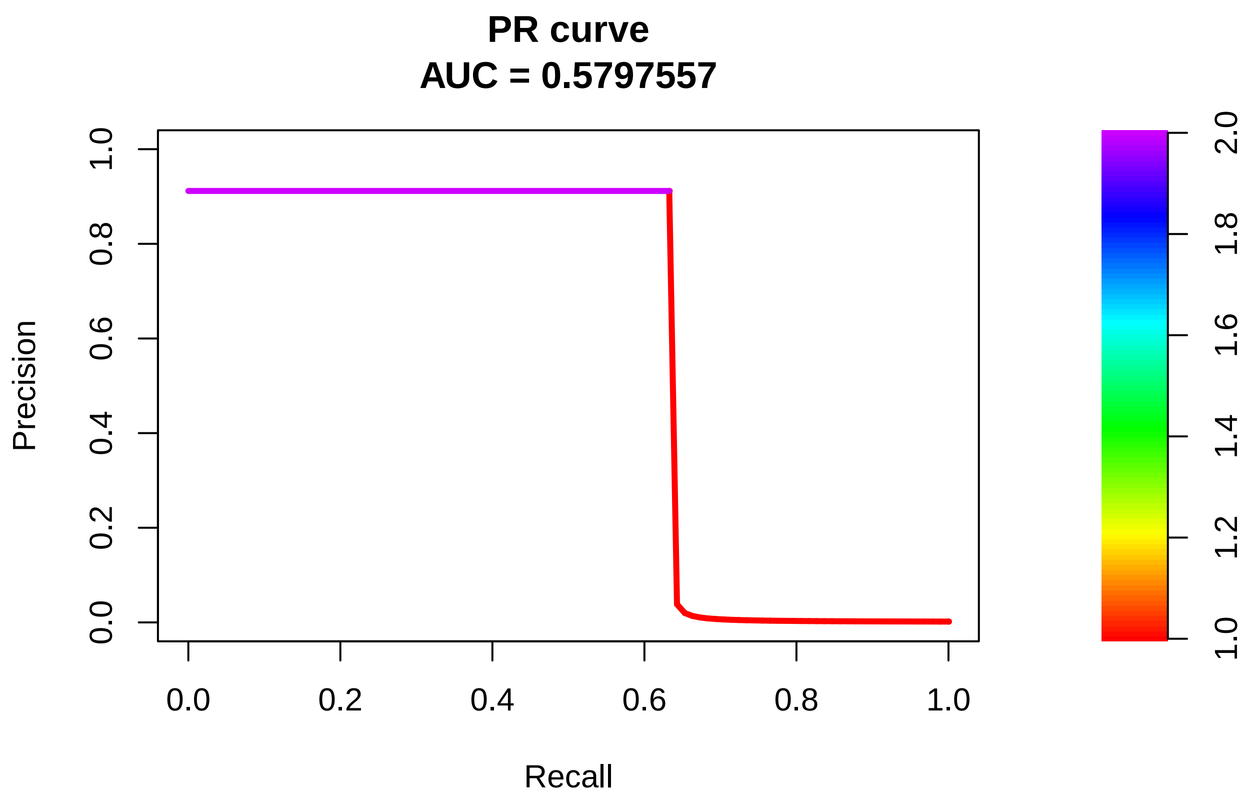

AUC: 0.917597684660626
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Specificity

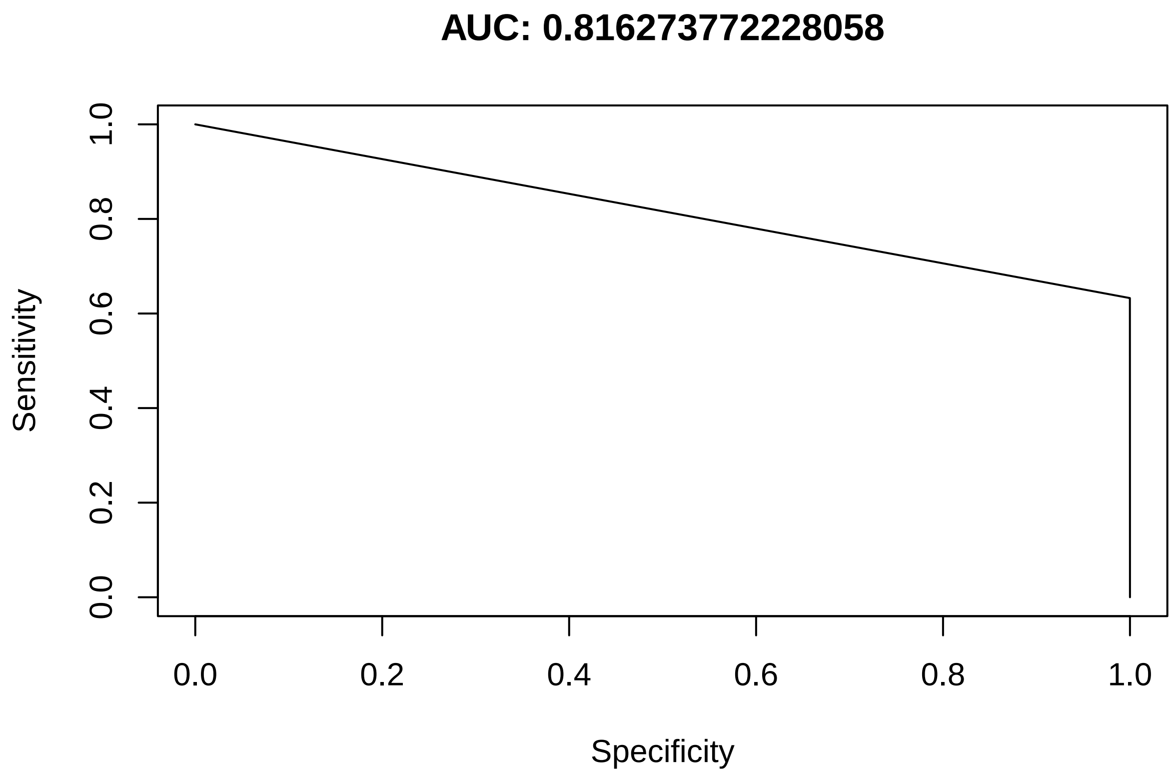
Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5000000	0.0000000
Naive Bayes	0.9175977	0.0548969



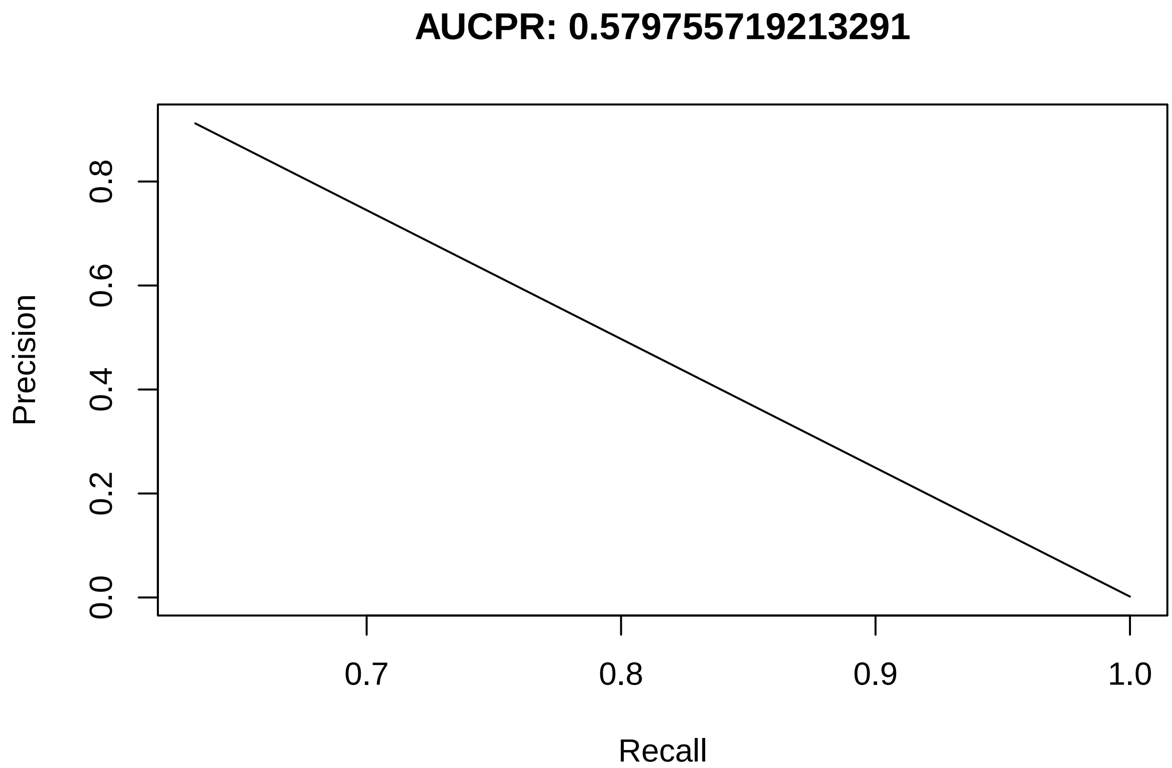
4.3 KNN - K-Nearest Neighbors

A KNN Model with k=5 can achieve a significant improvement with respect to the previous models, as regard AUCPR of **0.58** at the expense of a little drop off AUC, that is **0.81**.



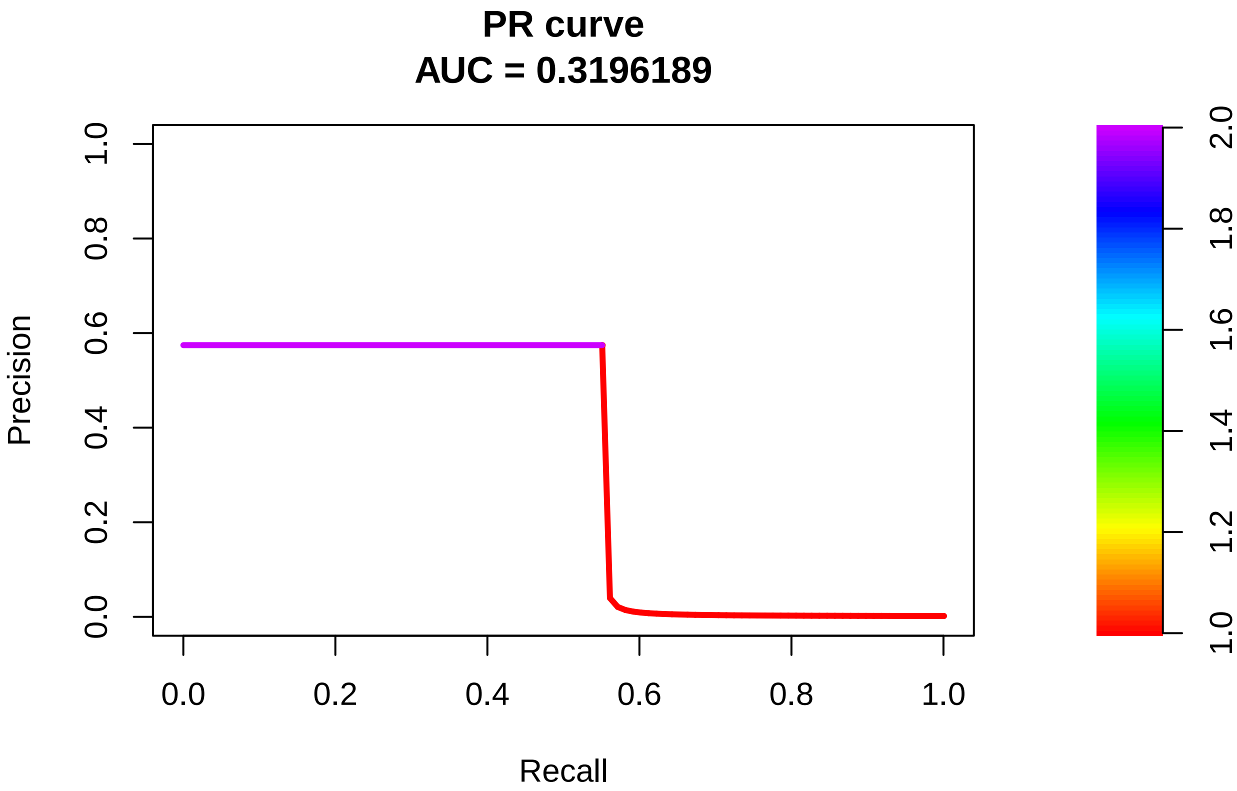

AUC: 0.816273772228058
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Specificity

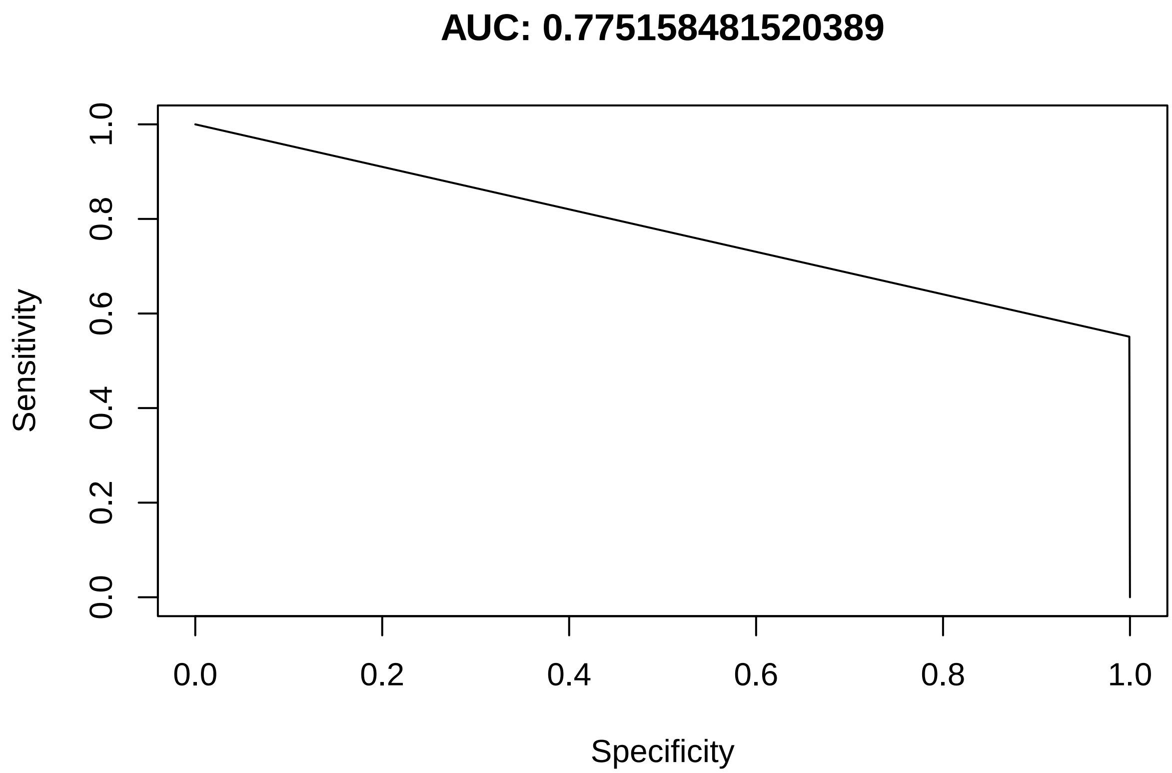
Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5000000	0.0000000
Naive Bayes	0.9175977	0.0548969
K-Nearest Neighbors k=5	0.8162738	0.5797557



4.4 SVM - Support Vector Machine

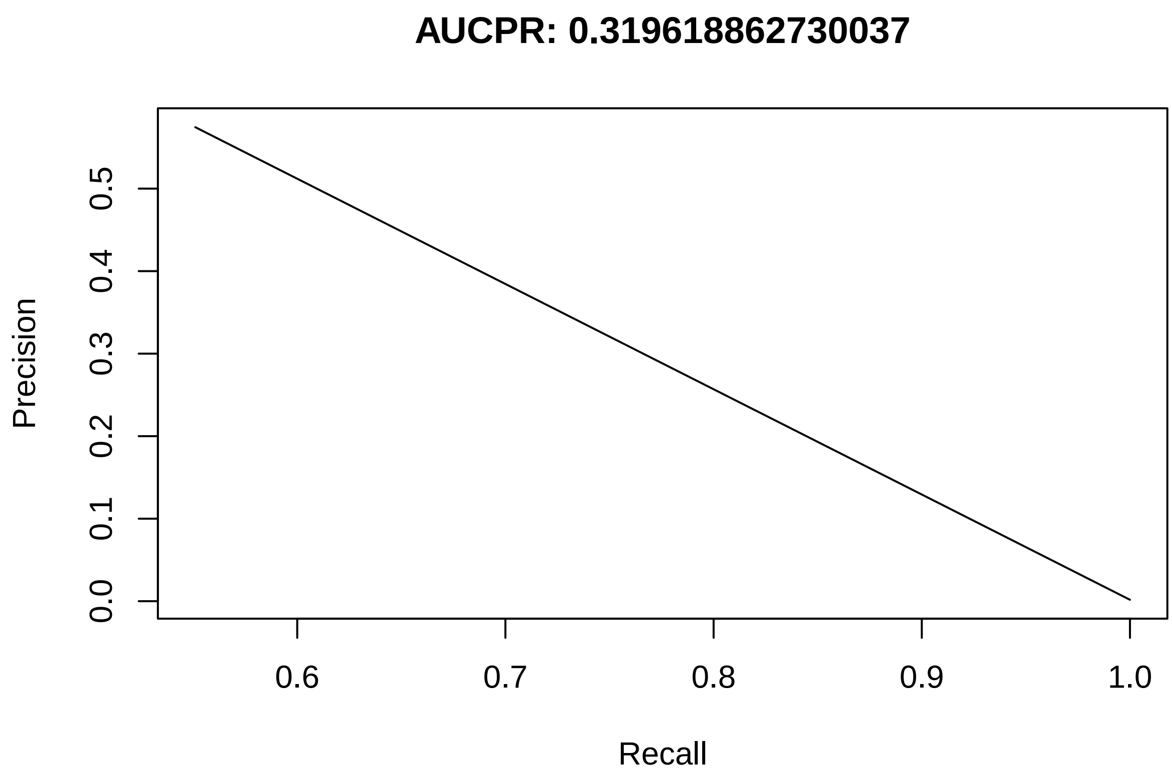
The SVM Model with a Sigmoid Kernel represent a step back on all fronts because the AUCPR is **0.32** and AUC is **0.77**.





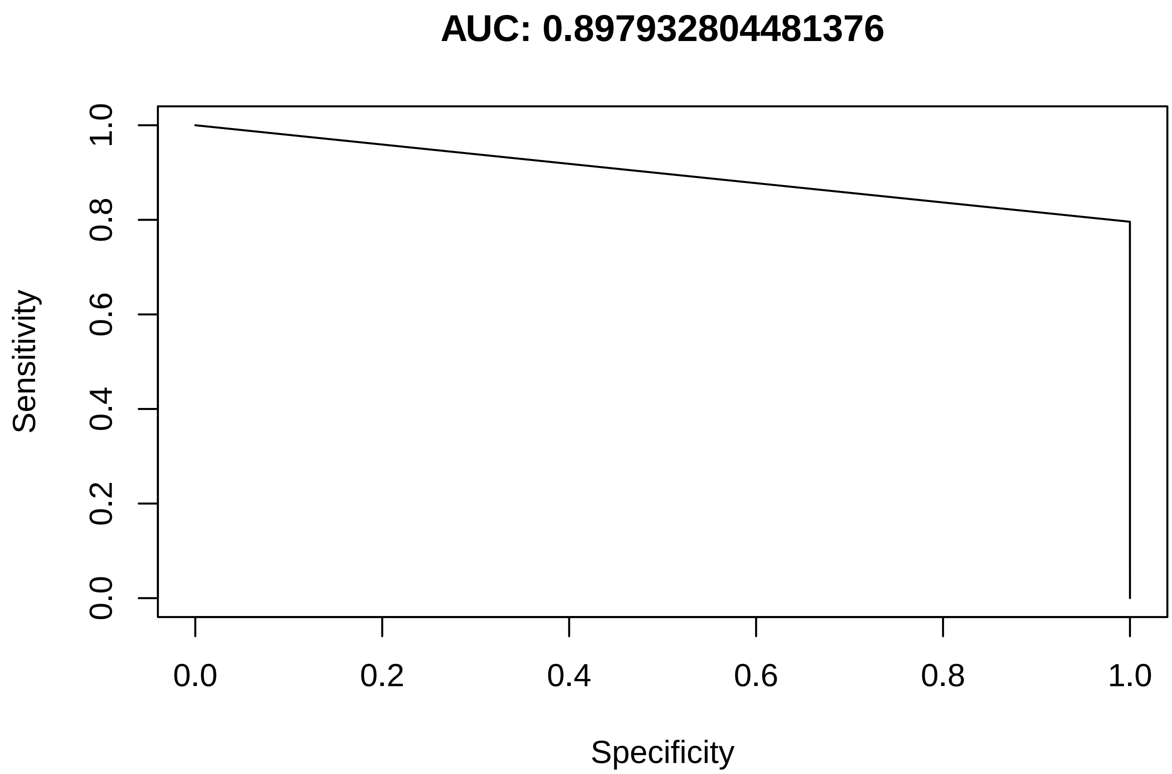
AUC: 0.775158481520389
Sensitivity
Specificity
0.0
0.2
0.4
0.6
0.8
1.0

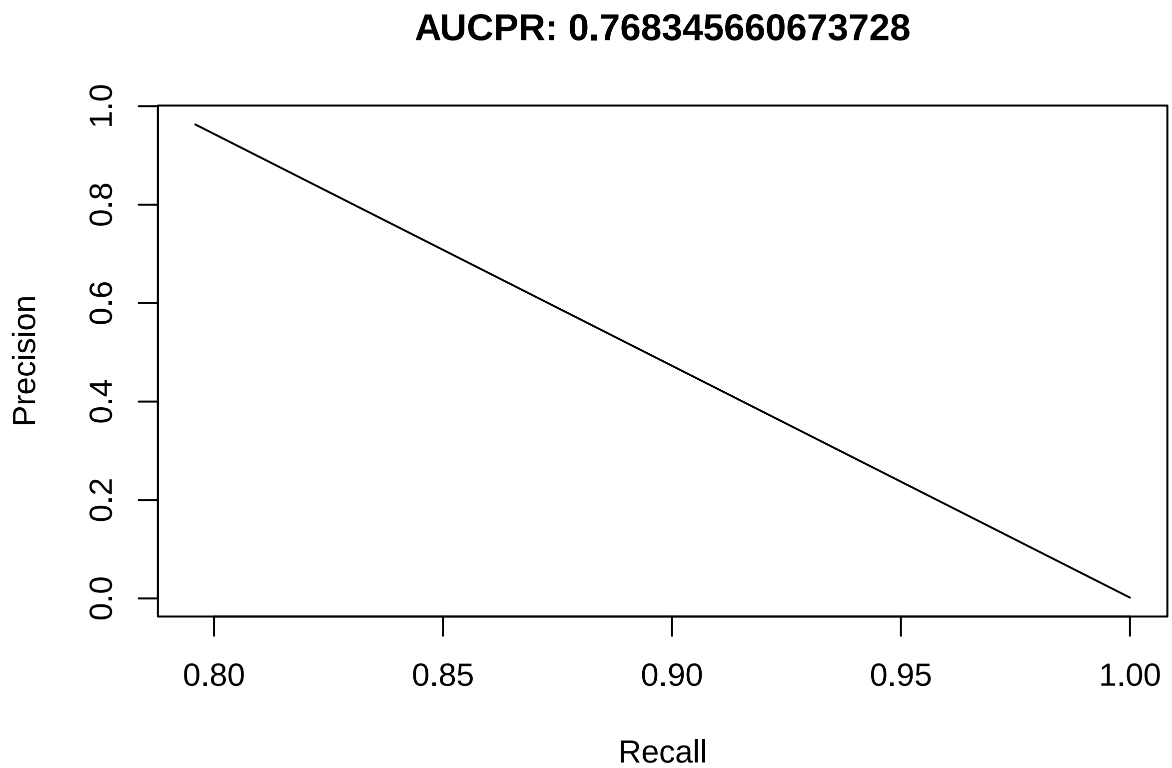
Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5000000	0.0000000
Naive Bayes	0.9175977	0.0548969
K-Nearest Neighbors k=5	0.8162738	0.5797557
SVM - Support Vector Machine	0.7751585	0.3196189



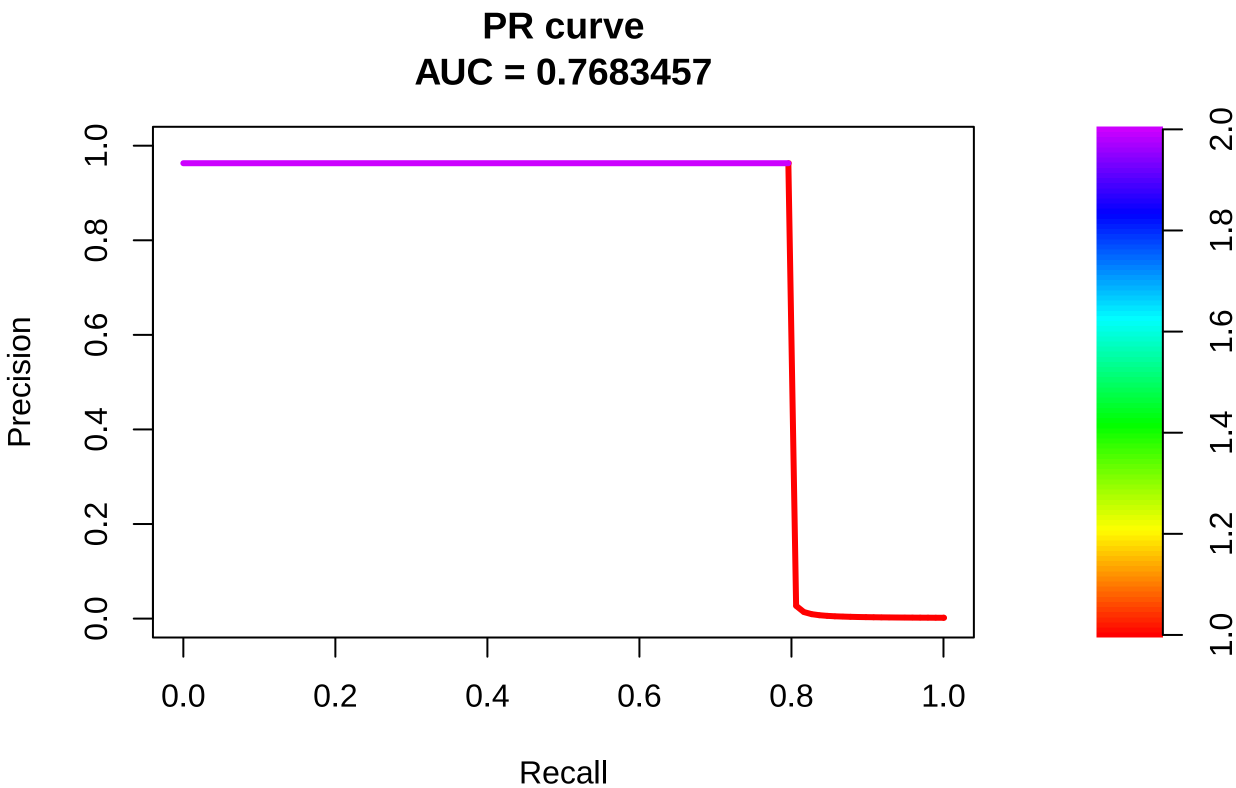
4.5 Random Forest

The ensemble methods are capable of a significant increase in performance. At the expense of another little drop off in terms of AUC (**0.9**) respect to the Naive Bayes model, there is a huge step forward in terms of AUCPR, that is **0.77**. This model doesn't reach the desired performance (AUCPR > 0.85), but it's close to it. As the plot and the table below suggest, there are few predictors like **V17**, **V12** and **V14** that are particularly useful for classifying a fraud.




AUCPR: 0.768345660673728
Precision
1.0
0.8
0.6
0.4
0.2
0.0
0.80
0.85
0.90
0.95
1.00
Recall

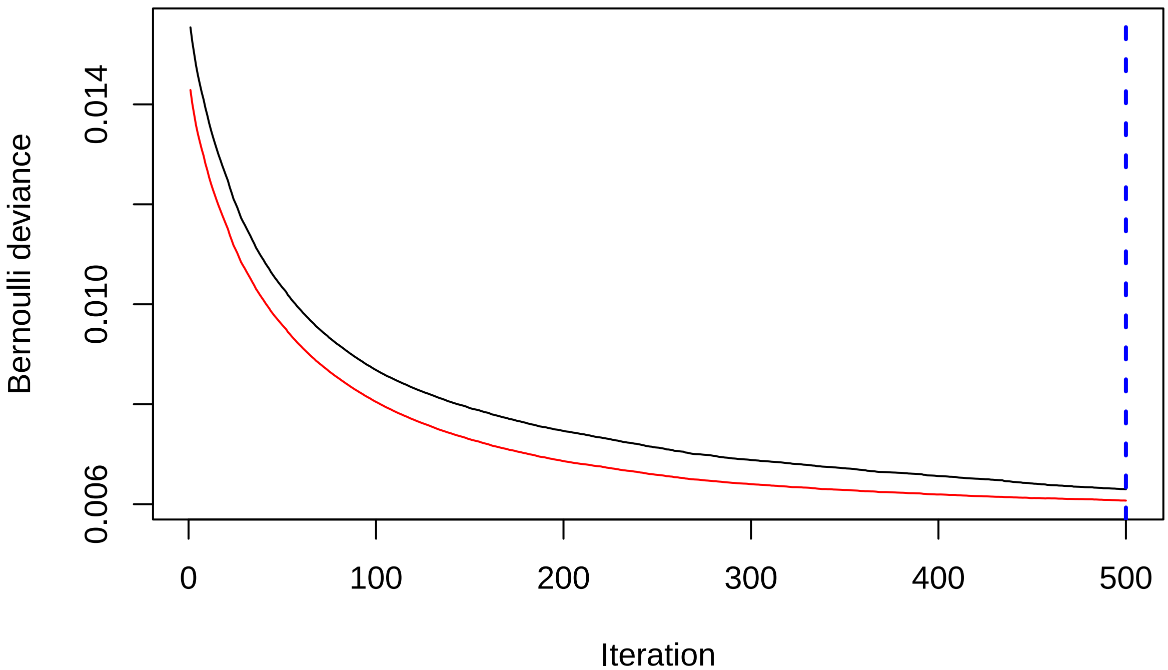
Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5000000	0.0000000
Naive Bayes	0.9175977	0.0548969
K-Nearest Neighbors k=5	0.8162738	0.5797557
SVM - Support Vector Machine	0.7751585	0.3196189
Random Forest	0.8979328	0.7683457

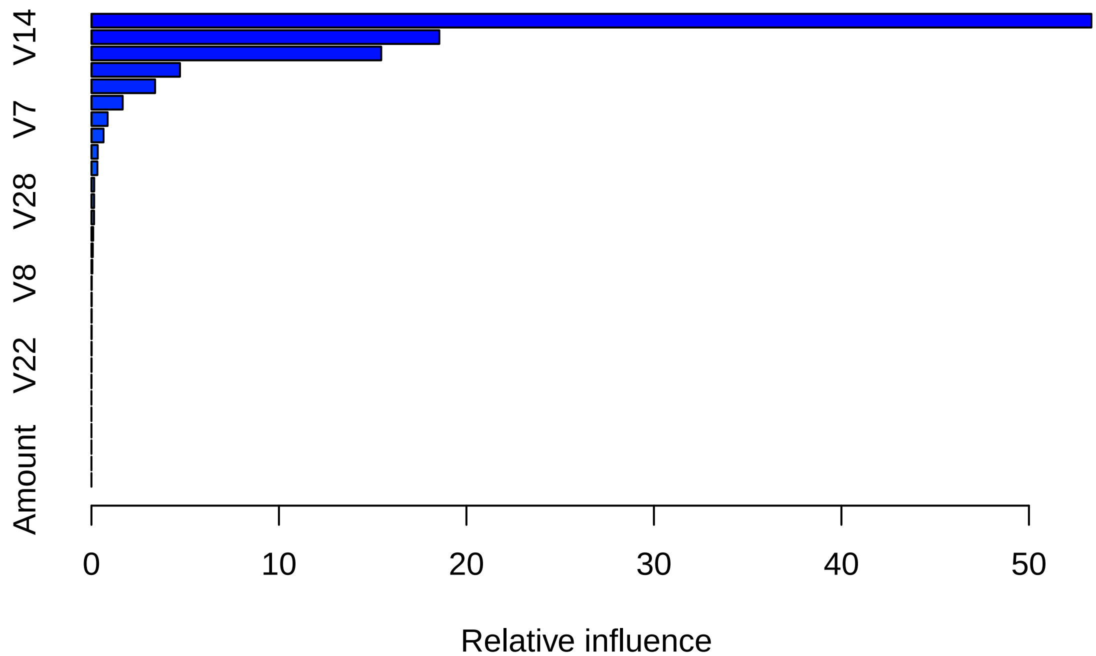


	MeanDecreaseGini
V1	8.708982
V2	7.784292
V3	8.985490
V4	17.257080
V5	7.772203
V6	8.821890
V7	19.072039
V8	7.013489
V9	23.520504
V10	43.772484
V11	44.997607
V12	73.056009
V13	6.829304
V14	63.479173
V15	6.388524
V16	40.124086
V17	105.084852
V18	16.236771
V19	8.041600
V20	8.359602
V21	10.723973
V22	5.886333
V23	4.705090
V24	6.127916
V25	5.290926
V26	10.888757
V27	9.216603
V28	6.266699
Amount	7.974071

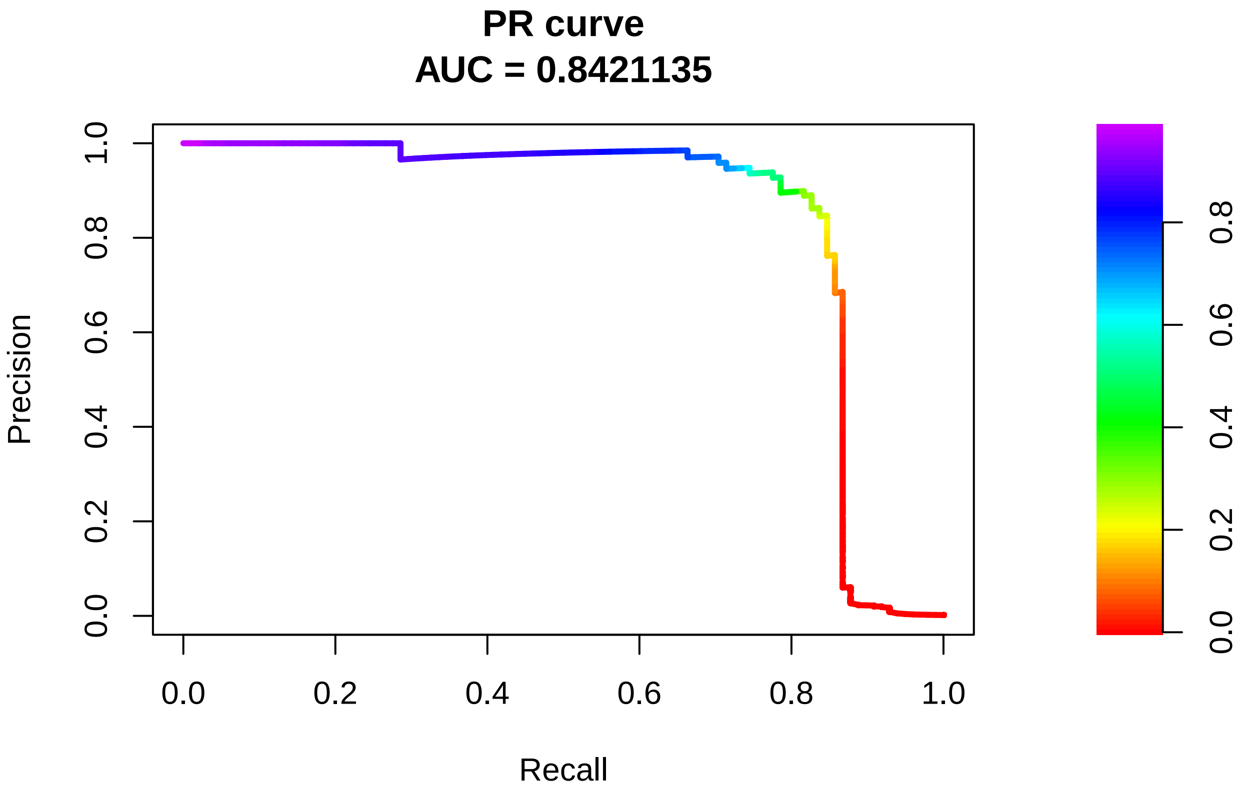
4.6 GBM - Generalized Boosted Regression

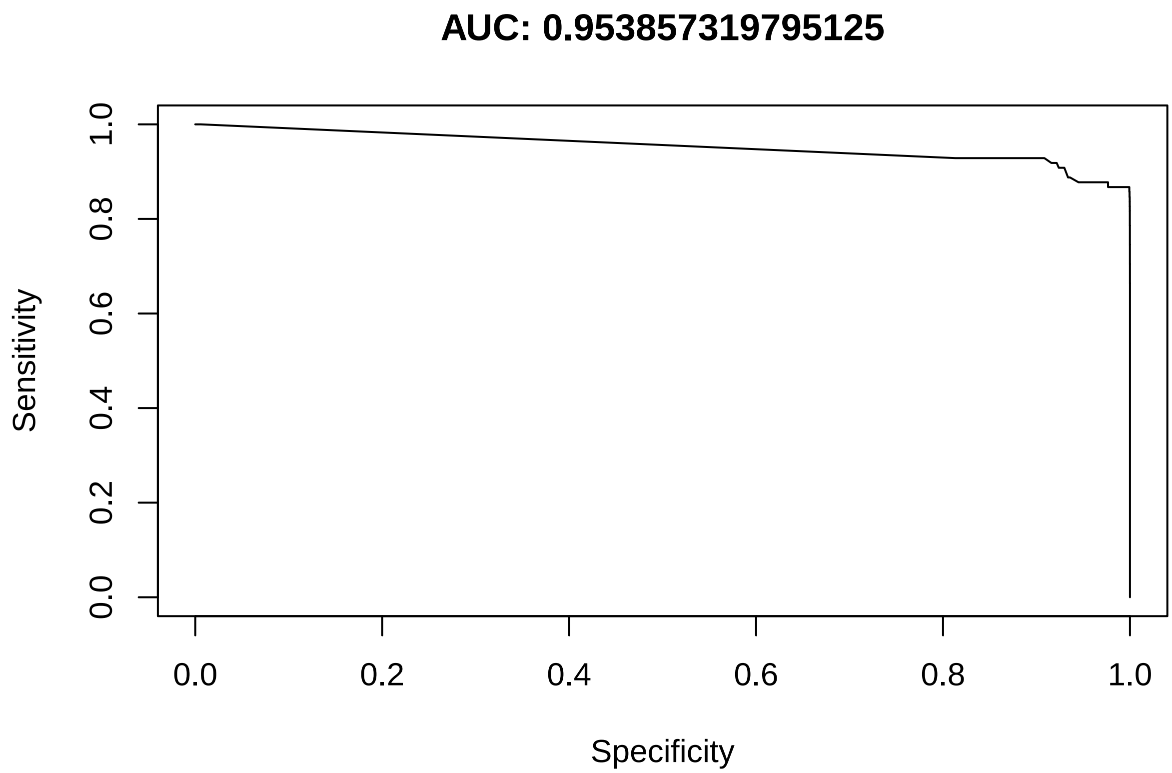
The GBM performance is really good: with an AUC of **0.95** and AUCPR of **0.94**, It doesn't achieve the target for a breath. As the Random Forest model shows, the **V17** and **V14** are still relevant to predict a fraud.



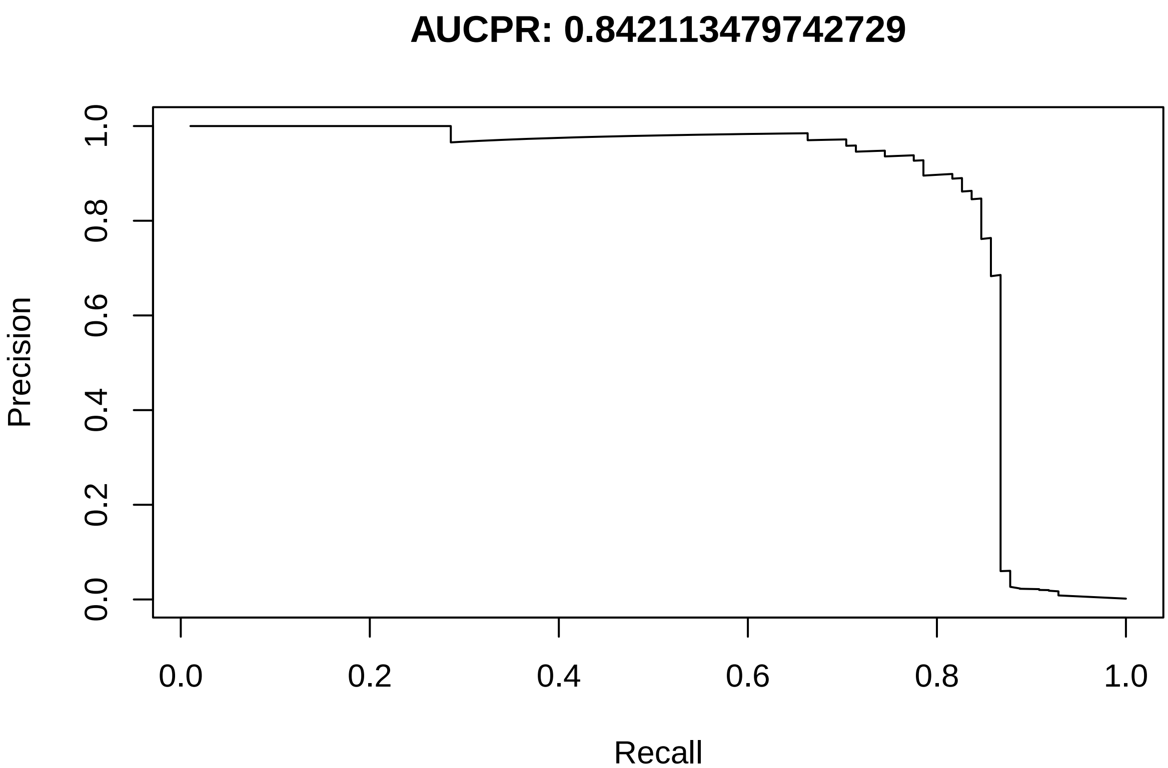


V14
V7
V28
V8
V22
Amount
0
10
20
30
40
50
Relative influence


PR curve
AUC = 0.8421135
Precision
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Recall
0.8
0.6
0.4
0.2
0.0


AUC: 0.953857319795125
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Specificity

Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5000000	0.0000000
Naive Bayes	0.9175977	0.0548969
K-Nearest Neighbors k=5	0.8162738	0.5797557
SVM - Support Vector Machine	0.7751585	0.3196189
Random Forest	0.8979328	0.7683457
GBM - Generalized Boosted Regression	0.9538573	0.8421135

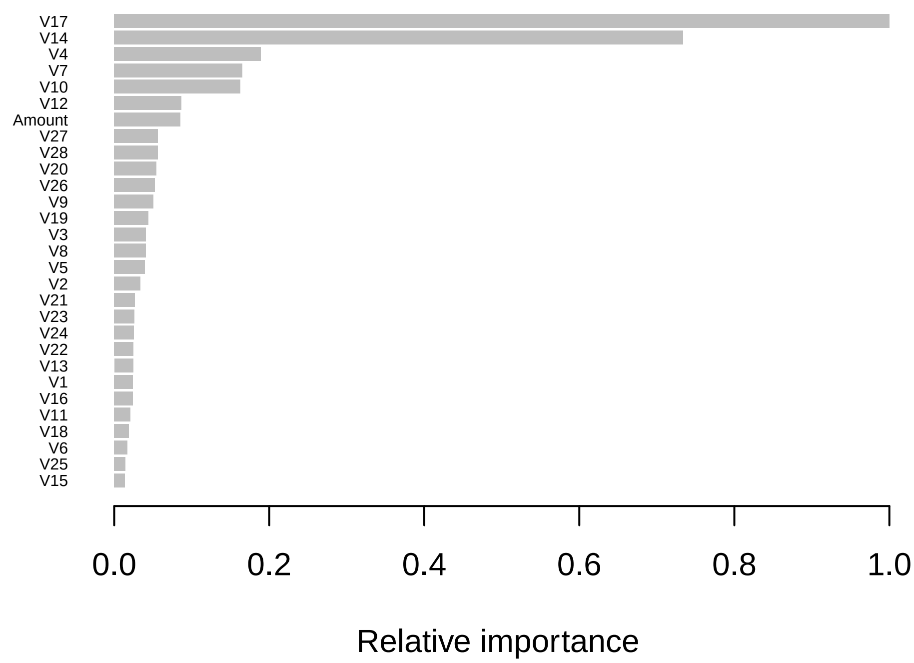


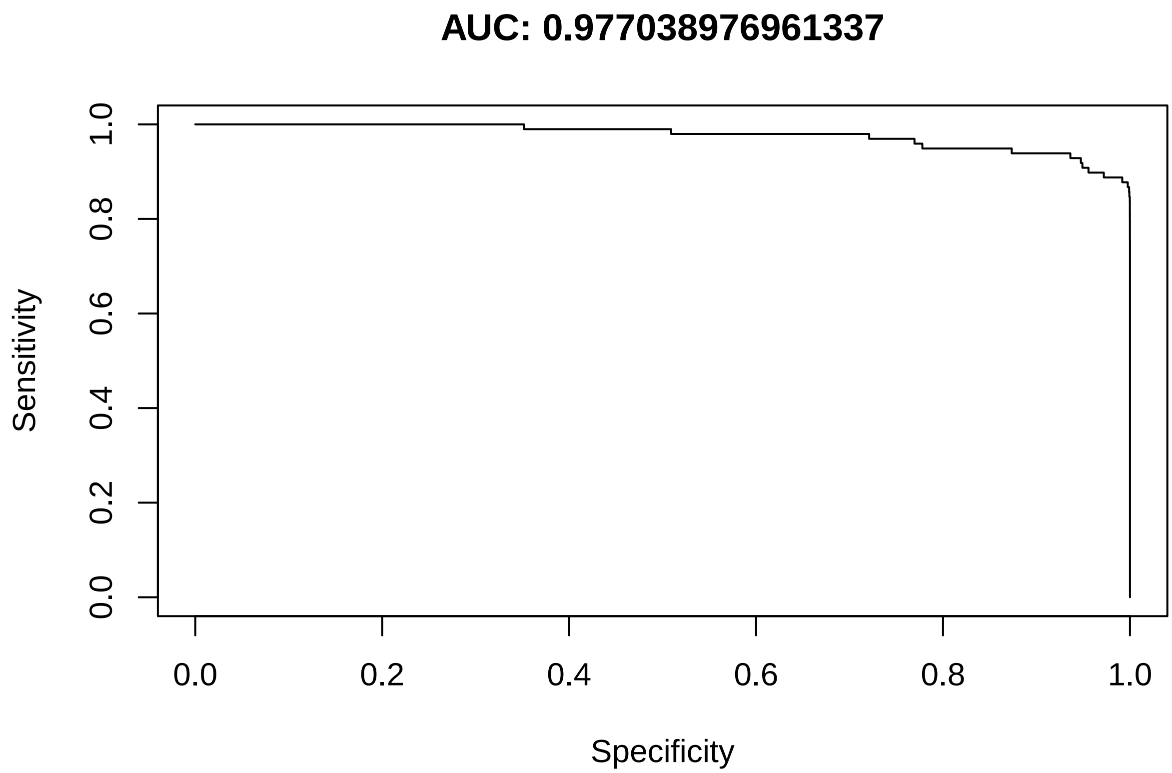
	var	rel.inf
V17	V17	53.3300209
V14	V14	18.5530357
V12	V12	15.4550412
V10	V10	4.7219307
V20	V20	3.3949817
V11	V11	1.6650329
V7	V7	0.8612551
V9	V9	0.6445507
V4	V4	0.3346926
V26	V26	0.3156347
V3	V3	0.1467431
V28	V28	0.1435442
V18	V18	0.1392624
V16	V16	0.0918682
V27	V27	0.0711635
V25	V25	0.0489084
V8	V8	0.0172958
V5	V5	0.0155866
V6	V6	0.0147381
V15	V15	0.0134430
V21	V21	0.0114564
V22	V22	0.0074806
V19	V19	0.0019186
V1	V1	0.0004148
V2	V2	0.0000000
V13	V13	0.0000000
V23	V23	0.0000000
V24	V24	0.0000000
Amount	Amount	0.0000000

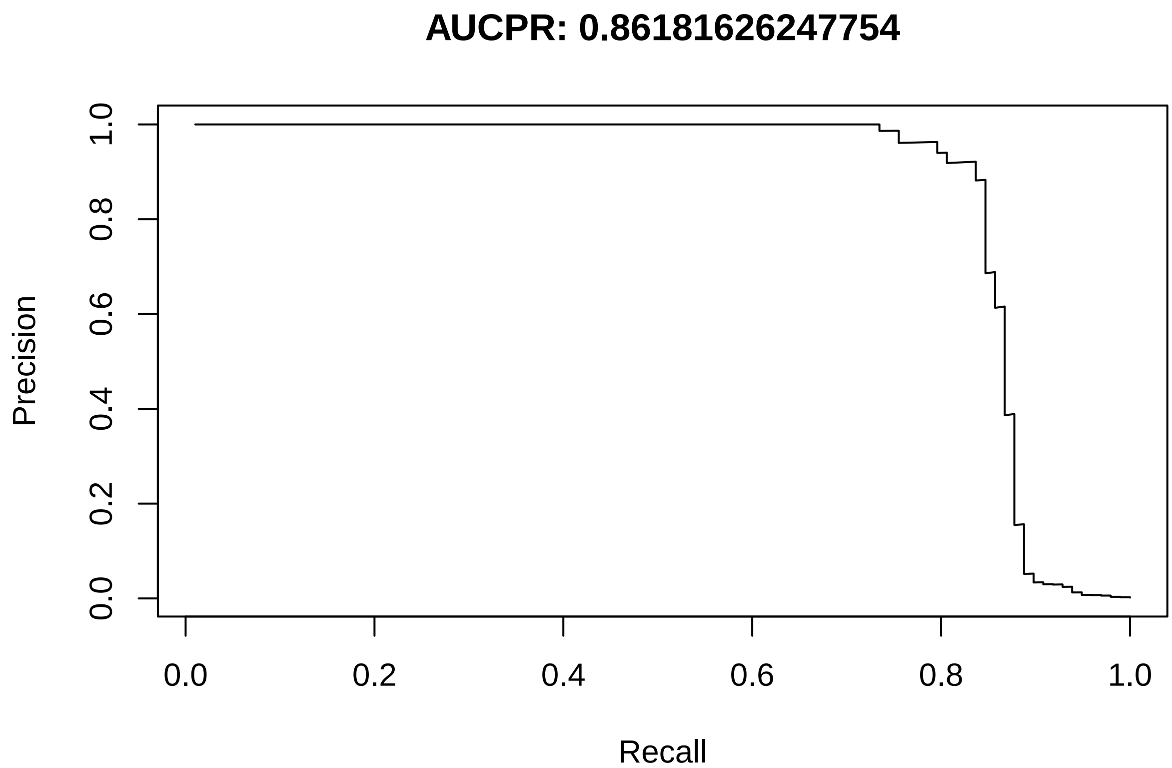
4.7 XGBoost- eXtreme Gradien Boosting

XGBoost is eXtreme Gradien Boosting package .This technique has been shown to produce models with high predictive accuracy.With an AUC of **0.98** and an AUCPR of **0.86** it has met expected metrics . As the previous model shown, **V17** and **V14** are still relevant to predict a fraud.

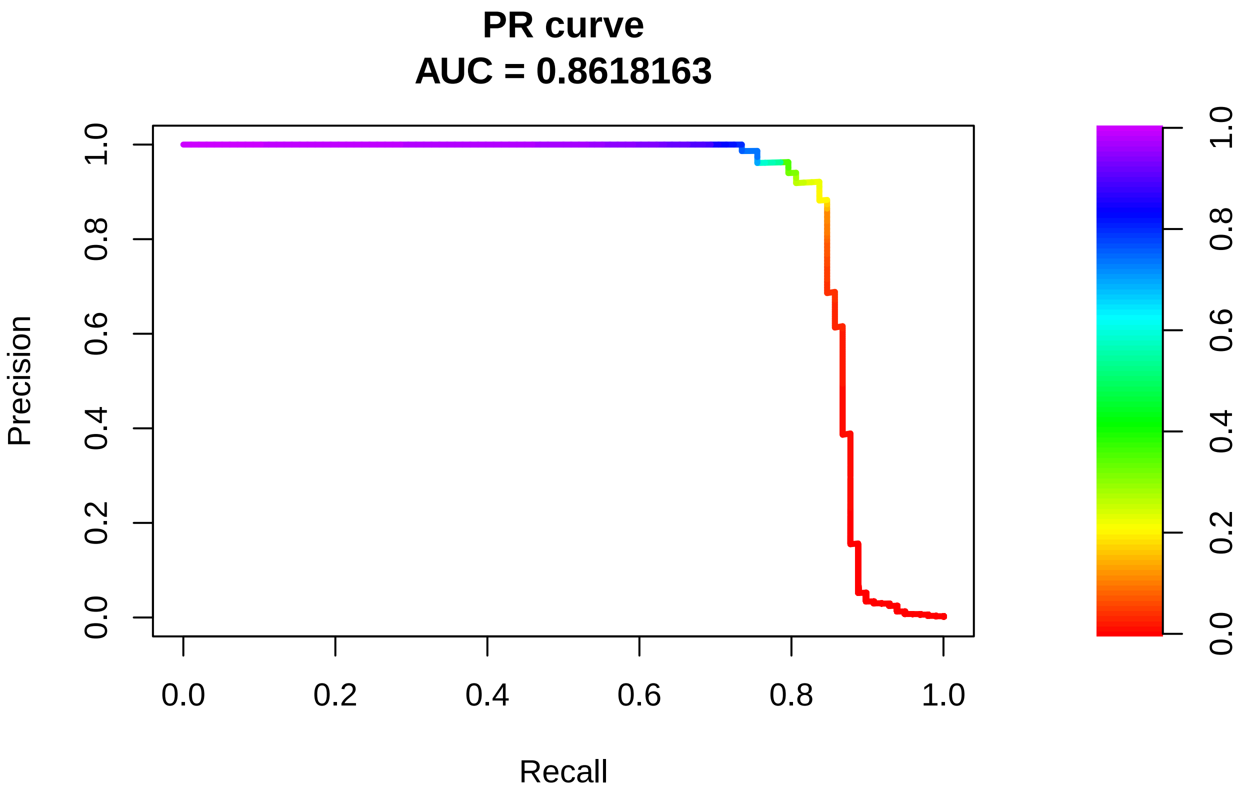
```
## [1]  test-aucpr:0.658215 cv-aucpr:0.651097
## Multiple eval metrics are present. Will use cv_aucpr for early stopping.
## Will train until cv_aucpr hasn't improved in 40 rounds.
##
## [101]    test-aucpr:0.857385 cv-aucpr:0.877270
## [201]    test-aucpr:0.862116 cv-aucpr:0.886406
## Stopping. Best iteration:
## [190]    test-aucpr:0.861816 cv-aucpr:0.887686
```




AUC: 0.977038976961337
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Specificity


AUCPR: 0.86181626247754
Precision
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Recall

Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5000000	0.0000000
Naive Bayes	0.9175977	0.0548969
K-Nearest Neighbors k=5	0.8162738	0.5797557
SVM - Support Vector Machine	0.7751585	0.3196189
Random Forest	0.8979328	0.7683457
GBM - Generalized Boosted Regression	0.9538573	0.8421135
XGBoost	0.9770390	0.8618163

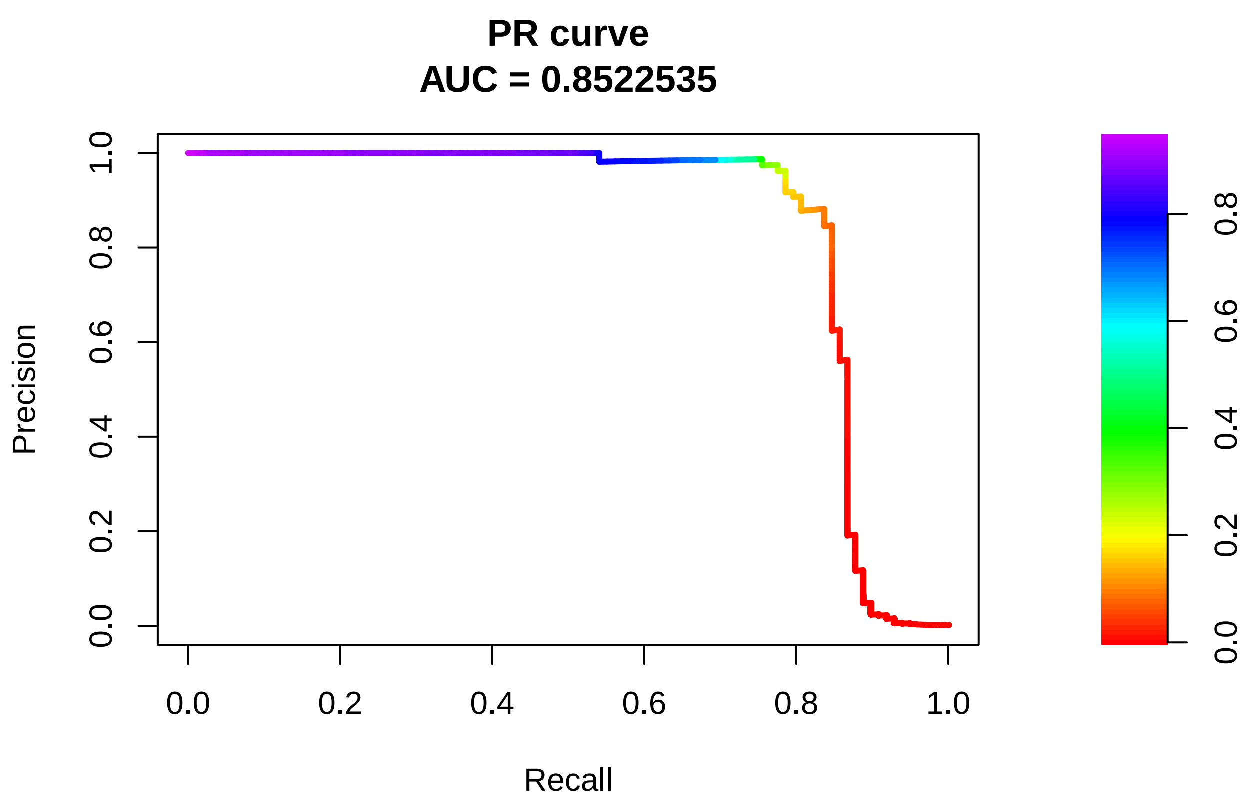


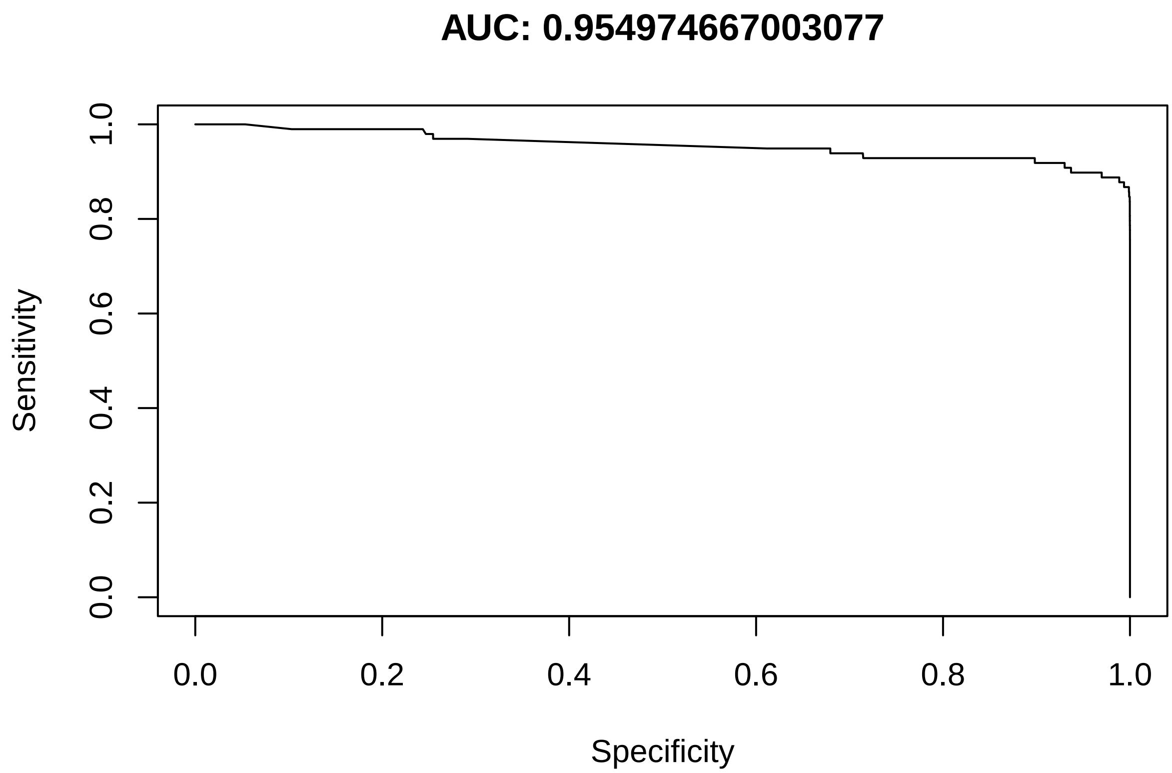
Feature	Gain	Cover	Frequency	Importance
V17	0.3171657	0.3376840	0.0590406	0.3171657
V14	0.2328285	0.4247761	0.0974170	0.2328285
V4	0.0600361	0.0149544	0.0900369	0.0600361
V7	0.0524206	0.0016778	0.0487085	0.0524206
V10	0.0515966	0.0024414	0.0442804	0.0515966
V12	0.0274032	0.1442810	0.0457565	0.0274032
Amount	0.0270669	0.0014754	0.0568266	0.0270669
V27	0.0179538	0.0006398	0.0265683	0.0179538
V28	0.0178111	0.0008319	0.0324723	0.0178111
V20	0.0171806	0.0008593	0.0250923	0.0171806
V26	0.0166046	0.0006860	0.0332103	0.0166046
V9	0.0161372	0.0059450	0.0265683	0.0161372
V19	0.0139521	0.0008483	0.0346863	0.0139521
V3	0.0129482	0.0014248	0.0391144	0.0129482
V8	0.0128923	0.0008873	0.0280443	0.0128923
V5	0.0125336	0.0188990	0.0324723	0.0125336
V2	0.0106854	0.0006103	0.0228782	0.0106854
V21	0.0084312	0.0007444	0.0191882	0.0084312
V23	0.0083561	0.0280382	0.0265683	0.0083561
V24	0.0079779	0.0005232	0.0250923	0.0079779
V22	0.0079069	0.0011115	0.0228782	0.0079069
V13	0.0077632	0.0008035	0.0243542	0.0077632
V1	0.0076040	0.0006159	0.0295203	0.0076040
V16	0.0076017	0.0069315	0.0258303	0.0076017
V11	0.0066428	0.0006218	0.0177122	0.0066428
V18	0.0060901	0.0004219	0.0199262	0.0060901
V6	0.0054157	0.0004609	0.0169742	0.0054157
V25	0.0045781	0.0004818	0.0169742	0.0045781
V15	0.0044156	0.0003236	0.0118081	0.0044156

4.8 LightGBM : Light Gradient Boosting Machine

LightGBM is the efficient and complex implementation of GBM. It has lot of parameters and because of this it has a steep learning curve. With a small change of the parameters, the LightGBM model is able to reach the performance of XGBoost,performance is bit worse: AUC of **0.95** and AUCPR of **0.85**.

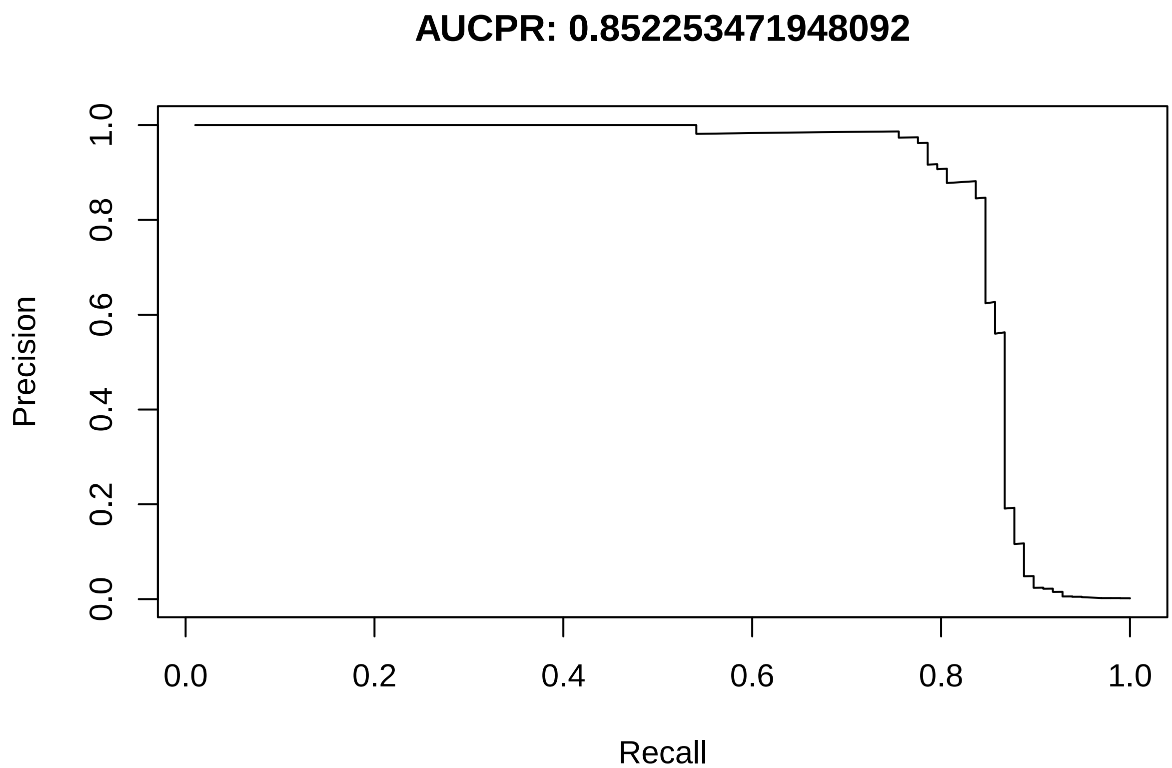
```
## [LightGBM] [Warning] verbosity is set=0, verbose=1 will be ignored. Current value: verbosity=0
## [LightGBM] [Warning] verbosity is set=0, verbose=1 will be ignored. Current value: verbosity=0
## [LightGBM] [Warning] verbosity is set=0, verbose=1 will be ignored. Current value: verbosity=0
## [LightGBM] [Warning] verbosity is set=0, verbose=1 will be ignored. Current value: verbosity=0
## [LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.023638 se
## You can set ‘force_col_wise=true‘ to remove the overhead.
## [LightGBM] [Warning] verbosity is set=0, verbose=1 will be ignored. Current value: verbosity=0
## [LightGBM] [Warning] verbosity is set=0, verbose=1 will be ignored. Current value: verbosity=0
## [1] "[1]:  test’s binary_error:0.00172048  cv’s binary_error:0.00172048"
## [1] "[21]:  test’s binary_error:0.00154492  cv’s binary_error:0.00149225"
## [1] "[41]:  test’s binary_error:0.000842682  cv’s binary_error:0.00080757"
## [1] "[61]:  test’s binary_error:0.00080757  cv’s binary_error:0.000772458"
## [1] "[81]:  test’s binary_error:0.000667123  cv’s binary_error:0.000719791"
## [1] "[101]:  test’s binary_error:0.000614456  cv’s binary_error:0.000632011"
## [1] "[121]:  test’s binary_error:0.000544232  cv’s binary_error:0.000561788"
## [1] "[141]:  test’s binary_error:0.000544232  cv’s binary_error:0.000544232"
## [1] "[161]:  test’s binary_error:0.00050912  cv’s binary_error:0.000544232"
```





AUC: 0.954974667003077
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Specificity

Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5000000	0.0000000
Naive Bayes	0.9175977	0.0548969
K-Nearest Neighbors k=5	0.8162738	0.5797557
SVM - Support Vector Machine	0.7751585	0.3196189
Random Forest	0.8979328	0.7683457
GBM - Generalized Boosted Regression	0.9538573	0.8421135
XGBoost	0.9770390	0.8618163
LightGBM	0.9549747	0.8522535



Feature	Gain	Cover	Frequency
V14	0.4290563	0.3499868	0.0855491
V7	0.2980483	0.0352024	0.0260116
V12	0.0345297	0.0208371	0.0549133
V26	0.0312950	0.0079741	0.0570328
V10	0.0249416	0.0059929	0.0381503
V4	0.0236295	0.2522238	0.0870906
V20	0.0198082	0.0435213	0.0385356
V1	0.0110255	0.0006144	0.0204239
V18	0.0086577	0.0016203	0.0283237
V13	0.0083061	0.0092839	0.0360308
V24	0.0082725	0.0027608	0.0342967
V2	0.0082198	0.0013769	0.0181118
Amount	0.0081963	0.0181656	0.0539499
V16	0.0080692	0.0066841	0.0208092
V9	0.0078733	0.0016156	0.0410405
V11	0.0074096	0.0377765	0.0342967
V21	0.0061725	0.0022685	0.0292871
V28	0.0058975	0.0024640	0.0315992
V15	0.0058189	0.0012221	0.0290944
V27	0.0056640	0.0335835	0.0433526
V3	0.0050209	0.0022023	0.0254335
V17	0.0047820	0.0342703	0.0157996
V5	0.0046542	0.0030638	0.0292871
V25	0.0045497	0.0003728	0.0096339
V23	0.0044318	0.0301643	0.0292871
V8	0.0044004	0.0157618	0.0248555
V19	0.0041902	0.0257136	0.0208092
V22	0.0039120	0.0524726	0.0211946
V6	0.0031670	0.0008040	0.0157996

Model	AUC	AUCPR
Naive Baseline - Predicts Legal	0.5000000	0.0000000
Naive Bayes	0.9175977	0.0548969
K-Nearest Neighbors k=5	0.8162738	0.5797557
SVM - Support Vector Machine	0.7751585	0.3196189
Random Forest	0.8979328	0.7683457
GBM - Generalized Boosted Regression	0.9538573	0.8421135
XGBoost	0.9770390	0.8618163
LightGBM	0.9549747	0.8522535

5 Results

Summary of all the models built for trained and validated.

6 Conclusion

The ensemble methods once again confirm themselves as among the best models out there. It easy to find them as a winners of numerous Kaggle's competitions or on TOP5 of them. Here, XGBoost model can achieve a very good AUCPR result of **0.86** and the others ensemble methods are very close to it. As the features importance plots and table show, there are few predictors like **V17** and **V14** that are particularly useful for classifying a fraud. The SMOTE technique (a technique for dealing with imbalanced data) could improve the performance a bit.

7 Appendix :

7.1 1b - Environment

```
##               _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          1.2
## year           2021
## month          11
## day            01
## svn rev        81115
## language       R
## version.string R version 4.1.2 (2021-11-01)
## nickname       Bird Hippie
```