Università di Pisa

Master Degree in Artificial Intelligence and Data Engineering

DataMining

Streaming QoS Dataset Analysis

Github repository:
https://github.com/TommyTheHuman/DataMining-QoSDataAnalysis

Tommaso Amarante
Edoardo Morucci

Chapter 1

Introduction

This analysis begins with the breakdown of the entire dataset that contains different parameters that describe the quality of a streaming service. The dataset is made up of 3 classes: NoStall, MildStall and SevereStall, moreover the dataset is heavily imbalance since NoStall has 75% of the tuples, MildStall has $\sim$ 20% and the presence of SevereStall is less than 5%.
Our aim is to create a model built with the entire dataset; this model will be able to predict the *Stall level* having a completely new QoS parameters as input.
The service owner will exploit these predictions in order to have the means for implementing targeted methods, thus the user experience is going to be improved.

Chapter 2

Analysis

We decided to adopt a **5-fold cross validation** with 4 different classifiers, we adopted the *sci-kit learn* python library implementation. We collected data and put them in some tables. In particular we decided to use:

- **Binary Decision Tree**
- **KNN**
- **Gaussian Bayes Classifier**
- **Random Forest**

Since the dataset is heavily imbalanced we decided to test the performances also after some rebalanceing techniques. We adopted a RandomUndersampling, a RandomOversampling and a SMOTEENN algorithm for the hybrid sampling method. We used *Imbalanced-learn* python library in order to implement this methods.

We also decided to test the performances after a *Feature selection*, for this purpose we used the **SelectKBest** from the *sci-kit learn* python library. We used the *Chi square* function in order to chose the right attributes.

We evaluate every classifier using *Precision, Recall and F1 measure* for each class in order to state the ability of the classifier to identify a stall. We have collected the model complexity for the tree-based classifier in order to compare them, moreover we have measured the time spent on the classification for each classifier on an *Apple M1* processor.

BDT Classifier

We decided to adopot the Information Gain as splitting criterion and we tried with a depth of 6 and a depth of 11.

KNN

For the KNN we evaluated the prediction with 5 and 10 neighbours. We used an heuristic approach as we tried different parameter values and the ones with the best results were 5 and 10 neighbour.

Gaussian Bayes Classifier

For this classifier we used the default settings offered by the sci-kit learn implementation.

Random Forest Classifier

We used Information Gain as splitting criterion but we did not limit the depth of the trees.

UnderSampling

We carried out 2 tests with different settings, in the first case we sampled with *NoStall*: 2000, *MildStall*: 2000, *SevereStall*: 635, in the second case we sampled with *NoStall*: 6000, *MildStall*: 6000, *SevereStall*: 635. In both cases we used 50 as *RandomState* and we allowed replacement during the sampling.

OverSampling

Also in this case we carried out 2 test, in the first case we sampled with *NoStall*: 40624, *MildStall*: 20000, *SevereStall*: 3000, wharease in the second we sampled with *NoStall*: 40624, *MildStall*: 20000, *SevereStall*: 10000. In both cases we used 10 as *RandomState*.

Hybrid Sampling

In order to implement this strategy we adopted the SMOTEENN implementation of the *imblearn* python library we decided to use 10 as RandomState, 4 as number of jobs and an *automatic* sampling strategy.

Feature Selection

In order to make a more precise analysis on the dataset we decided to adopt a feature selection using the **SelectKbest** implementation of the *sci-kit learn* python library combined with the *chi-squared* function. We set K at 20 so the dataset will have 20 attributes instead of 29. Also for this choice we adopted an heuristic approach. We tried different numbers of attributes and we noticed that by reducing the number of attributes by a small quantity the complexity and the values are insignificantly reduced, whereas by reducing the attributes by a significant amount the complexity is reduced but the classifiers lose the capacity of recognise the classes. We ended up with a K value of 20 and we collected the following results.

In the following chapter we will analyse the results collected.

Chapter 3

Results

In the following tables we collected the **F1 measure**, **Precision** and **Recall** values for each class in the dataset using different classifiers. By doing that we can evaluate the ability of each classifier to recognise each class.
In the tables the highest value for each column is marked in italic bold, whereas the second highest value is marked only in bold.

Table 3.1 summarizes the results obtained on the entire dataset. Since the dataset is heavly imbalanced we decided to adopt different sampling strategies, we started with an **undersample**. The results are collected in table 3.2 and 3.3. We tried with different sampling configuration and we noticed that sampling with NoStall: 6000, MildStall: 6000, SevereStall: 635 and NoStall: 2000, MildStall: 2000, SevereStall: 635 were the best ones in terms of performances and model complexity. We adopted the undersample with 6000 on the majority classes because the performances on the minority class are overall better even though the complexity of the tree-based classifiers is significantly higher than in the sample with 2000 on the majority classes.

In tables 3.4 and 3.5 the results using the **Oversample** approach are collected. The performances remained more or less the same, referring to the general dataset, while the general complexity increased significantly in the tree-based classifiers.

We also decided to use an hybrid approach between undersampling and oversampling, in order to improve this way we used the **SMOTEENN** algorithm. With this algorithm we have an increment in the *recall* column but there is a decrease in the *precision* column. We can notice from the speed column that this algorithm takes from 100 to 1000 times more than with the other approaches.

After this analysis we also decided to adopt a **feature selection** for a deeper study of the entire dataset. The aim of this approach is to speed up the performances while not changing the general results. As we can notice in the dedicated section, the model complexity for the tree-based classifier only decreases in the undersampled data shown in table 3.8 and 3.9. We can assert that in the other cases we waste time and resources since neither the performances nor the model complexity are better.

3.1 Results without feature selection

ORIGINAL DATASET													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	0,9020	0,9335	0,9173	0,7588	0,6935	0,7227	**0,7599**	0,3993	0,5206	0,8677	60,2	119,4	5,0
BDT-11	**0,9102**	0,9316	**0,9206**	**0,7646**	**0,7136**	**0,7369**	0,6490	0,5719	**0,6059**	**0,8733**	739,6	1478,2	8,0
KNN-5	0,7647	0,8779	0,8174	0,3674	0,2197	0,2749	0,3592	**0,6041**	0,1001	0,7049			51,6
KNN-10	0,7604	**0,9423**	0,8416	0,4347	0,1422	0,2142	0,4001	0,1422	0,2142	0,7334			55,9
Gaussian	0,8763	0,4256	0,5721	0,2707	0,1837	0,2186	0,2199	***0,8980***	0,4294	0,3709			0,1
RF	***0,9164***	***0,9540***	***0,9348***	***0,8249***	***0,7370***	***0,7777***	***0,8681***	0,5429	***0,6643***	***0,8950***	337984,8	675869,6	114,6

Table 3.1

UNDER DATASET 2000 2000 635													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	**0,9485**	**0,8182**	**0,8784**	**0,6013**	**0,8102**	**0,6899**	0,3478	0,7456	0,4675	**0,8154**	52,6	104,2	0,5
BDT-11	0,9346	0,8073	0,8661	0,5768	0,7834	0,6638	0,3755	0,7292	**0,4951**	0,8005	254,6	508,2	0,7
KNN-5	0,8122	0,5909	0,6839	0,3168	0,5263	0,3953	***0,9514***	0,4004	0,1536	0,5727			7,1
KNN-10	0,8136	0,6354	0,7134	0,3284	0,5005	0,3964	0,1097	0,4092	0,1728	0,5993			7,0
Gaussian	0,8802	0,4137	0,5619	0,2673	0,1786	0,2139	0,0217	***0,9106***	0,0423	0,3609			0,1
RF	***0,9541***	***0,8659***	***0,9078***	***0,6768***	***0,8451***	***0,7511***	**0,5469**	**0,8275**	***0,6566***	***0,8951***	40283,2	80466,4	8,0

Table 3.2

UNDER DATASET 6000 6000 635													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	**0,9457**	**0,8317**	**0,8848**	**0,6280**	**0,8335**	**0,7106**	**0,5518**	**0,6675**	**0,5958**	**0,8302**	60,6	120,2	1,2
BDT-11	0,9433	0,8249	0,8799	0,6079	0,8252	0,6993	0,5103	0,6474	0,5678	0,8229	442,4	883,8	1,9
KNN-5	0,8068	0,5916	0,6824	0,3174	0,5619	0,4055	0,1391	0,2027	0,1635	0,5798			19,0
KNN-10	0,8080	0,6510	0,7208	0,3559	0,5274	0,4101	0,1689	0,1901	0,1781	0,6150			17,5
Gaussian	0,8800	0,4136	0,5618	0,2697	0,1907	0,2231	0,2195	***0,9042***	0,4285	0,3638			0,1
RF	***0,9544***	***0,8753***	***0,9130***	***0,6921***	***0,8647***	***0,7683***	***0,7404***	0,6538	***0,6905***	***0,8701***	94871,0	189642,0	23,2

Table 3.3

OVER DATASET 40924 20000 3000													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	0,9117	**0,9189**	0,9153	0,7383	0,7087	0,7228	0,5430	**0,7217**	**0,6160**	0,8644	59,0	117,0	5,3
BDT-11	**0,9206**	0,9179	**0,9192**	**0,7404**	**0,7416**	**0,7404**	0,5366	0,6083	0,5699	**0,8705**	739,8	1478,6	8,4
KNN-5	0,7807	0,7744	0,7775	0,3381	0,3296	0,3337	0,1081	0,2228	0,1452	0,6565			58,2
KNN-10	0,7759	0,8523	0,8123	0,3709	0,2353	0,2878	***0,9851***	0,2543	0,1416	0,6921			63,4
Gaussian	0,8776	0,4212	0,5684	0,2682	0,1775	0,2134	0,2182	***0,9068***	0,4261	0,3662			0,2
RF	***0,9220***	***0,9470***	***0,9340***	***0,8120***	***0,7550***	***0,7820***	**0,8170**	0,6110	***0,6950***	***0,8951***	375445,8	750791,6	131,4

Table 3.4

OVER DATASET 40924 20000 10000													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	***0,9260***	0,8819	0,9034	0,6741	0,7232	0,6977	0,3448	**0,8552**	0,4908	0,8422	62,8	124,6	6,3
BDT-11	0,9195	**0,9112**	**0,9152**	**0,7268**	**0,7306**	**0,7280**	0,4810	0,6865	**0,5645**	***0,8637***	741,6	1482,2	9,3
KNN-5	0,7811	0,7730	0,7770	0,3375	0,3277	0,3224	0,1027	0,2354	0,1426	0,6562			74,9
KNN-10	0,7777	0,8448	0,8098	0,3683	0,2233	0,2779	***0,8875***	0,3512	0,1415	0,6846			77,4
Gaussian	0,8776	0,4221	0,5684	0,2677	0,1714	0,2088	0,2167	***0,9105***	0,4233	0,3648			0,2
RF	**0,9220**	***0,9460***	***0,9340***	***0,8100***	***0,7550***	***0,7810***	**0,8010**	0,6010	***0,6840***	**0,8632**	378908,4	757716,8	135,7

Table 3.5

SMOTEENN													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	***0,9616***	0,7456	0,8398	0,5287	0,8310	0,6459	0,3007	**0,9081**	0,4507	0,7686	63,0	125,0	1898,9
BDT-11	0,9584	**0,7682**	**0,8526**	**0,5518**	**0,8494**	**0,6686**	**0,4009**	0,8188	**0,5369**	**0,7889**	737,0	1473,0	1898,9
KNN-5	0,8585	0,4099	0,5540	0,2988	0,6620	0,4115	0,0511	0,4231	0,0912	0,4727			1865,1
KNN-10	0,8660	0,3990	0,5460	0,2990	0,6570	0,4110	0,0491	0,4770	0,0890	0,4643			1865,1
Gaussian	0,8862	0,3998	0,5500	0,2673	0,1970	0,2266	0,0218	***0,9106***	0,0426	0,3552			1937,1
RF	**0,9602**	***0,8238***	***0,8866***	***0,6211***	***0,8692***	***0,7240***	***0,5576***	0,8087	***0,6593***	***0,8348***	265744,2	531388,4	2042,5

Table 3.6

3.2 Results with feature selection

ORIGINAL DATASET													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	0,8930	0,9241	0,9079	**0,7313**	0,6663	0,6937	**0,7675**	0,3667	0,4898	**0,8535**	62,4	123,8	3,3
BTD-11	**0,9028**	0,9184	**0,9104**	0,7263	**0,6958**	**0,7098**	0,6266	**0,4560**	**0,5228**	0,8525	821,6	1642,2	5,2
KNN-5	0,7647	0,8779	0,8174	0,3675	0,2198	0,2749	0,3592	0,0604	0,1001	0,7049			52,4
KNN-10	0,7600	***0,9420***	0,8420	0,4350	0,1420	0,2140	0,4000	0,0642	0,1080	0,7334			56,0
Gaussian	0,8763	0,4256	0,5721	0,2707	0,1837	0,2186	0,0220	***0,8980***	0,0429	0,3709			0,2
RF	***0,9060***	**0,9380**	***0,9220***	***0,7750***	***0,7110***	***0,7410***	***0,8550***	0,4030	***0,5440***	***0,8759***	404778,8	809457,6	87,5

Table 3.7

UNDER DATASET 2000 2000 635													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	**0,9417**	0,7909	0,8595	0,5652	**0,8101**	0,6651	0,3633	0,6500	0,4583	0,7940	52,8	104,6	0,5
BTD-11	0,9181	***0,8935***	***0,9054***	***0,6918***	0,7381	***0,7130***	***0,4608***	0,5429	***0,4970***	**0,7941**	279,4	557,8	0,6
KNN-5	0,8123	0,5910	0,6840	0,3168	0,5263	0,3954	0,0951	0,4005	0,1536	0,5727			7,2
KNN-10	0,8136	0,6355	0,7135	0,3284	0,5006	0,3965	0,1098	0,4093	0,1728	0,5993			7,1
Gaussian	0,8802	0,4137	0,5619	0,2673	0,1786	0,2139	0,0217	***0,9106***	0,0423	0,3609			0,1
RF	***0,9470***	**0,8152**	**0,8757**	**0,5974**	***0,8125***	**0,6875**	**0,3737**	**0,7495**	***0,4970***	***0,8137***	40179,6	80259,2	6,7

Table 3.8

UNDER DATASET 6000 6000 635													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	**0,9422**	0,8045	0,8671	0,5865	**0,8366**	**0,6878**	**0,5508**	0,4964	***0,5130***	0,8089	59,6	118,2	0,9
BTD-11	0,9371	**0,8151**	**0,8714**	**0,5916**	0,8094	0,6824	0,4252	**0,5417**	**0,4759**	**0,8105**	477,6	954,2	1,3
KNN-5	0,8069	0,5917	0,6825	0,3174	0,5620	0,4055	0,1392	0,2027	0,1635	0,5798			18,8
KNN-10	0,8080	0,6511	0,7209	0,3359	0,5275	0,4101	0,1690	0,1902	0,1781	0,6150			20,2
Gaussian	0,8801	0,4136	0,5619	0,2697	0,1907	0,2232	0,0220	***0,9043***	0,0429	0,3638			0,2
RF	***0,9471***	***0,8254***	***0,8818***	***0,6120***	***0,8616***	***0,7150***	***0,8383***	0,3023	0,4434	***0,8284***	95124,2	190148,4	19,0

Table 3.9

OVER DATASET 40924 20000 3000													
	No Stall			Mild Stall			Severe Stall				Complexity		Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	**0,9177**	0,8752	0,8956	0,6607	**0,7476**	0,6999	**0,5114**	0,4925	0,4928	0,8390	62,0	123,0	3,7
BTD-11	***0,9181***	***0,8938***	***0,9056***	***0,6922***	0,7383	**0,7133**	0,4619	**0,5404**	**0,4969**	**0,8508**	791,2	1581,4	6,0
KNN-5	0,7812	0,7731	0,7770	0,3376	0,3278	0,3324	0,1028	0,2355	0,1426	0,6562			77,7
KNN-10	0,7777	0,8448	0,8098	0,3683	0,2233	0,2780	0,0888	0,3513	0,1415	0,6846			82,0
Gaussian	0,8776	0,4212	0,5684	0,2683	0,1775	0,2134	0,0218	***0,9068***	0,0426	0,3662			0,2
RF	0,9157	**0,8919**	**0,9035**	**0,6883**	***0,7550***	***0,7193***	***0,7831***	0,3880	***0,5183***	***0,8521***	375412,8	750725,6	103,4

Table 3.10

OVER DATASET 40924 20000 10000													
	No Stall			Mild Stall			Severe Stall			Complexity			Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	0,9127	0,8848	0,8980	0,6664	0,6503	0,6547	0,2477	**0,8187**	0,3798	0,8257	63,0	125,0	3,8
BTD-11	***0,9190***	***0,8920***	***0,9050***	***0,6880***	***0,7270***	***0,7060***	***0,3760***	0,6250	***0,4690***	***0,8476***	825,8	1650,6	6,2
KNN-5	0,7810	0,7730	0,7770	0,3380	0,3280	0,3320	0,1030	0,2350	0,1430	0,6562			69,8
KNN-10	0,7777	0,8448	0,8098	0,3683	0,2233	0,2780	0,0888	0,3513	0,1415	0,6846			77,6
Gaussian	0,8780	0,4210	0,5680	0,2680	0,1710	0,2090	0,0217	***0,9110***	0,0423	0,3648			0,3
RF	**0,9162**	**0,8912**	**0,9034**	**0,6821**	**0,6808**	**0,6810**	**0,2976**	0,8086	**0,4342**	**0,8379**	379110,2	758120,4	105,9

Table 3.11

SMOTEENN													
	No Stall			Mild Stall			Severe Stall			Complexity			Speed
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Accuracy	Leaves	Nodes	
BDT-6	***0,9590***	0,6850	0,7980	0,4710	0,7870	0,5880	0,1900	**0,8840**	0,3130	0,7127	62,4	123,8	1763,2
BTD-11	0,9568	**0,7438**	**0,8366**	**0,5235**	**0,8245**	**0,6399**	**0,2767**	0,7797	**0,4083**	**0,7642**	797,6	1594,2	1710,7
KNN-5	0,8580	0,4100	0,5540	0,2990	0,6620	0,4110	0,0511	0,4230	0,0912	0,4727			1841,3
KNN-10	0,8661	0,3994	0,5456	0,2992	0,6573	0,4109	0,0491	0,4773	0,0890	0,4643			1819,2
Gaussian	0,8862	0,3998	0,5501	0,2673	0,1970	0,2266	0,0218	***0,9106***	0,0426	0,3552			1725,1
RF	**0,9570**	***0,7921***	***0,8665***	***0,5802***	***0,8604***	***0,6924***	***0,4701***	0,7305	***0,5704***	***0,8083***	327799,2	655498,8	1982,3

Table 3.12

3.3 Graphs

In order to understand better our results we decided to build up different **Recall-Precision graphs** only for the test without the feature selection. In all the couples the graph on the left represents the Mild Stall, whereas on the right the Sever Stall is shown.

From the graphs we can assert that KNN-5, KNN-11 and Gaussian Bayes are not capable of recognising the Mild Stall since they all are in the lower left of the graph. These classifiers are not good also for the Severe Stall since they do not ever have high Recall and Precision at the same time.

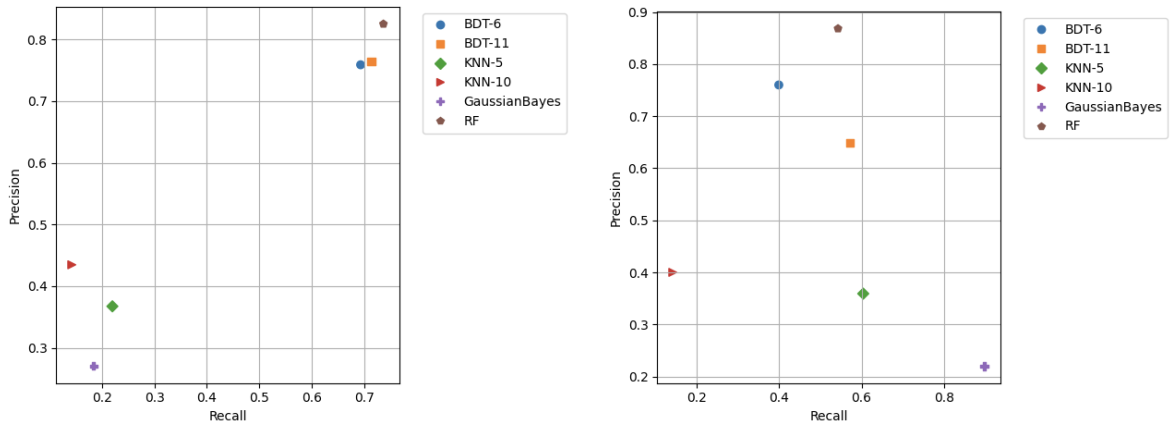

Figure 3.1: Mild and severe stall for original dataset

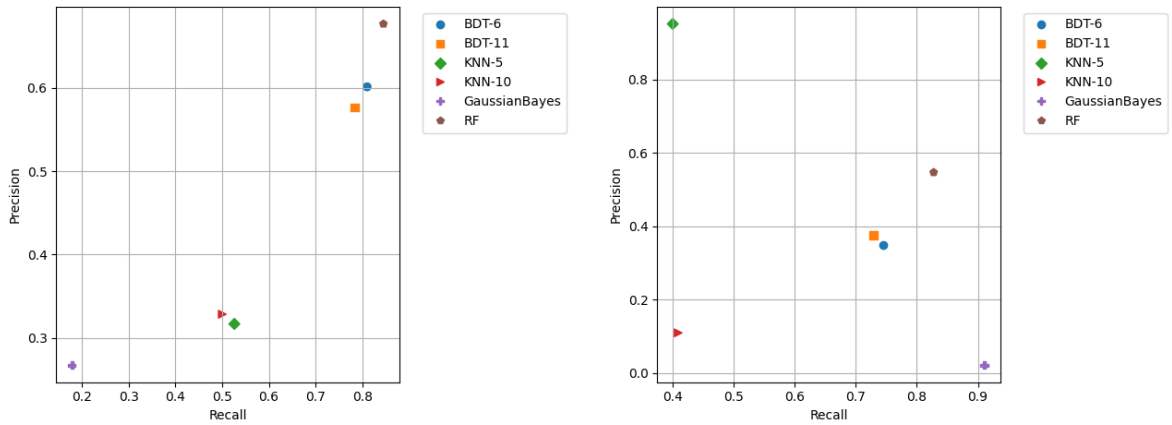

Figure 3.2: Mild and severe stall for undersampling 2000 2000 635

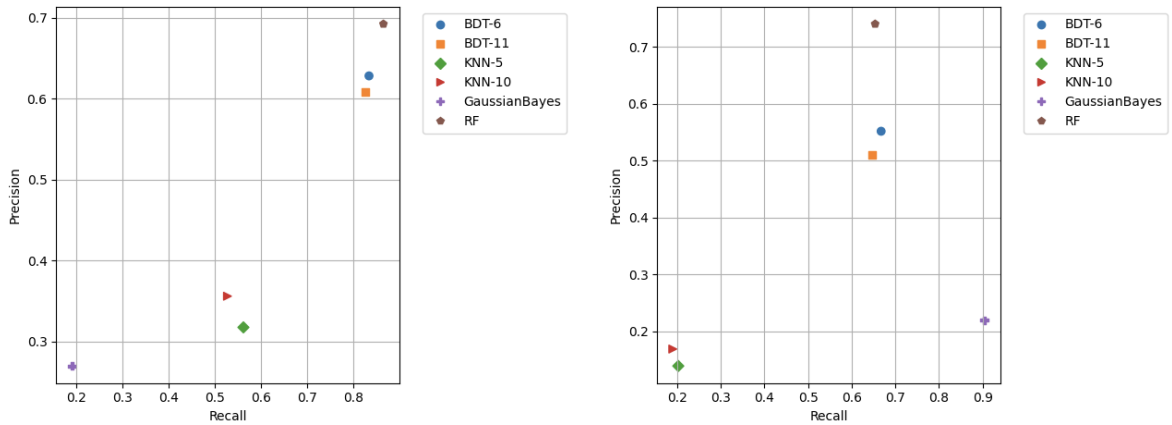

Figure 3.3: Mild and severe stall for undersampling 6000 6000 635

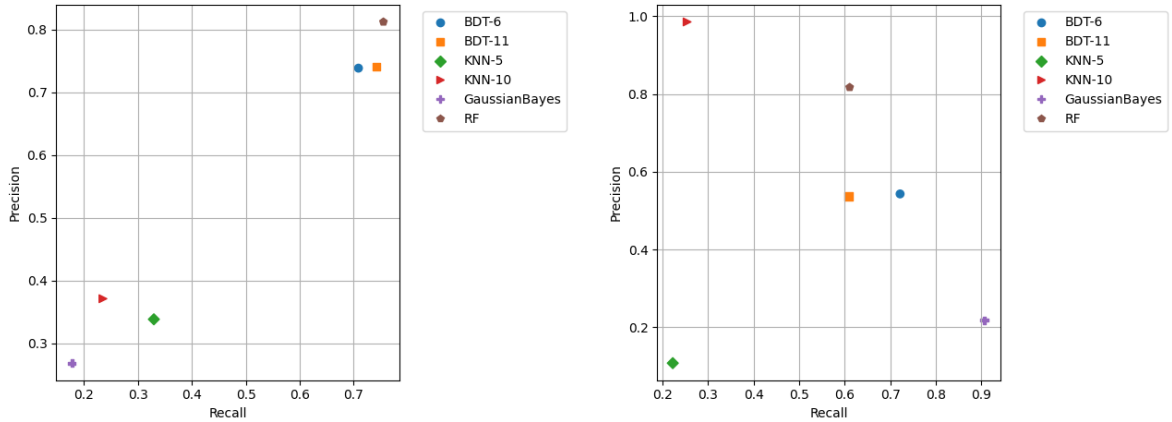

Figure 3.4: Mild and severe stall for oversampling 40924 20000 3000

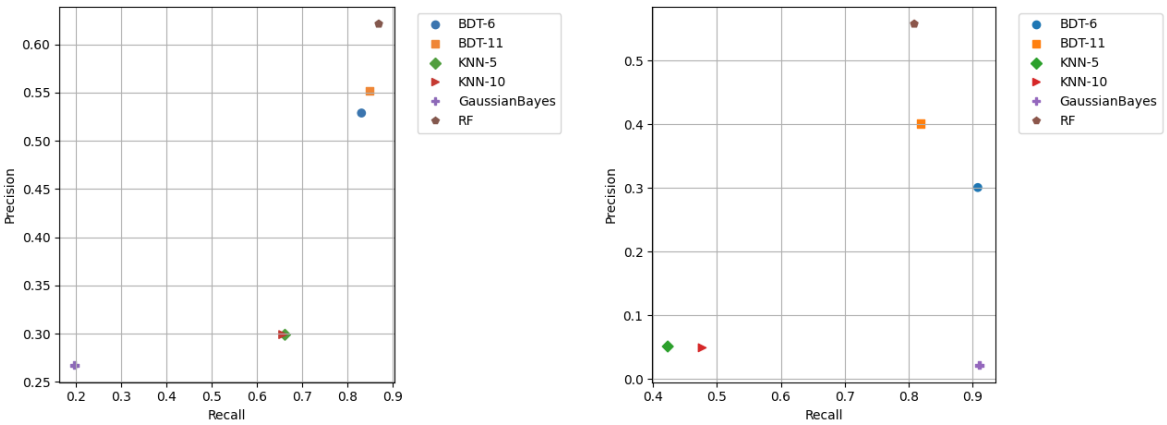

Figure 3.5: Mild and severe stall for sampling with smote

Chapter 4

Conclusions

To sum up first of all *Feature selection* is worth only combined with the under sampling method otherwise the complexity will increase without any significant performance increasement.

RandomForest has the highest values for F1-measure and accuracy for all the test we had evaluated without the feature selection, all these measurement are more or less the same. It is important to notice that, among all the RF test we carried out, the lowest complexity model is in the undersampling (2000 - 2000 - 635). In this case, considering the speed column, the value that we have collected is about 10 times smaller than the other RF values.

The only problem with the RF classifier is that it has low interpretability. BDT-6 has slightly worse values than the RF previously selected but the binary decision tree is more interpretable. The best values of the BDT-6 are in the undersampling (6000 - 6000 - 635) as shown in table 3.2 and in graph 3.3. BDT-6 proved to be to optimal choice also in presence of time constraints, since it has the optimal trade off between execution time and total accuracy.