

AIDA-Project

Telekom Customer Churn Prediction

Table of content

Table of content	0
Project	1
Motivation	1
Task focus	1
Project Task	1
Exploratory Data Analysis (EDA)	1
Statistical distributions on features, split on classification “churn”	2
Feature Selection	5
Data Preparation	6
Results	7
Metrics	7
Standard Classifiers	8
Neural Network Classifier	11
Simple Neural Network	11
Hidden layer variations	12
Conclusion	13
Appendix on NN results	14

1. Project

1.1. Motivation

The data contain the customer churn data of a national telecommunications service provider (within 50 states) offering national plans for voice and mail, as well international plans.

The task is to estimate the customers with churn tendency from the given data to enable reliable churn prevention measures. Therefore the prediction of customers with potential churn tendency needs to be as accurate as possible.

1.2. Task focus

M. Sauer: data preparation, feature selection, GitHub support
J. E. Dietert: statistical distributions, standard classifiers, neural networks
T. Habermann: metrics, knn-classifier, neural networks

1.3. Project Task

The final objective is to build a classifier to predict customer churn among the customers in the data. The output of the system would be an insight for the companies about the important factors that accelerate the churn process. To do that, the following tasks to be accomplished are mentioned within the specific chapters.

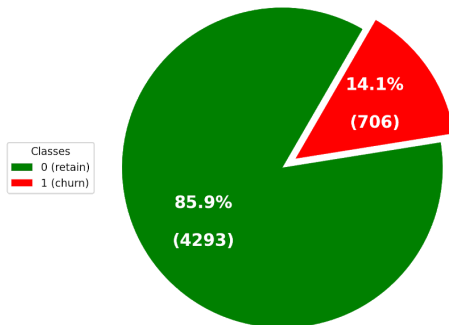
2. Exploratory Data Analysis (EDA)

The features contain:

- basic customer information
 - phone number, state, account length, area code
 - binary information on international plan, voice & mail plan
- call records per day, evening, night, as well international calls
 - minutes, charges, calls
 - number of customer service calls

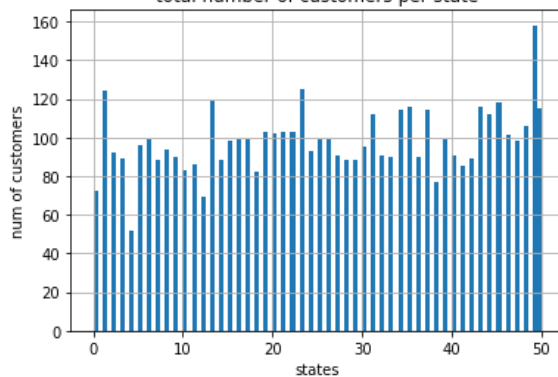
2.1. Statistical distributions on features, split on classification “churn”

Raw data class distribution



artificial dataset with churn 1/7 of 5000 customers ~14% churn rate

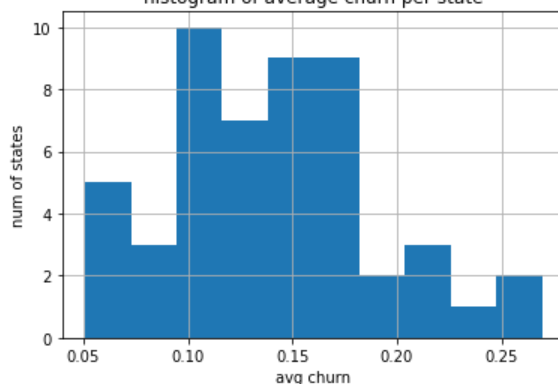
total number of customers per state



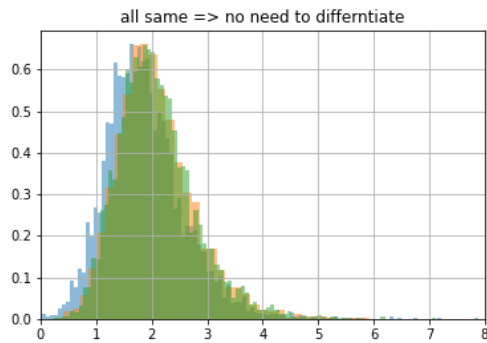
One initial distribution is done over the number of customers selected per state which is not balanced, but within 50...125.

These numbers are too low to build a reliable state-based classification model since there might be zero data sets of churn.

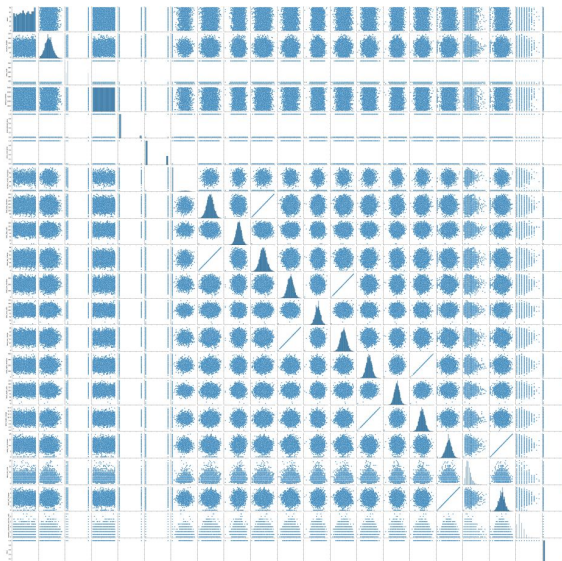
histogram of average churn per state



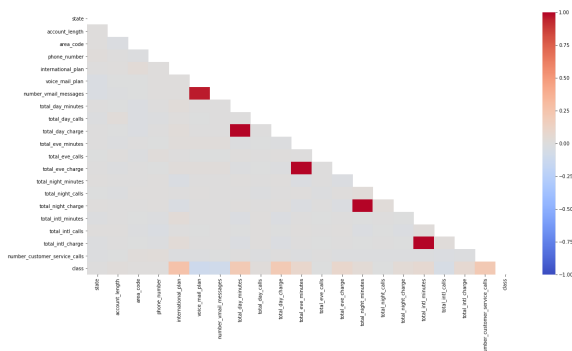
The average churn rate per state also shows some differentiation between states. Since the overall average churn rate is 14%, there exist some states with significantly higher churn >27%, while some only have ~5%



On the left, the average call duration time per day, evening or night is depicted, but the difference is tiny: evening and night call have 2.02min/call, while at day 1.88min/call. This is reasonable since the day charge is 17ct/minute, while 8.5ct in evening and 4.5ct/minute at night. We had the idea, (but not realized so far) to weight the day minutes with the charge in order to get some quadratic influence/importance..

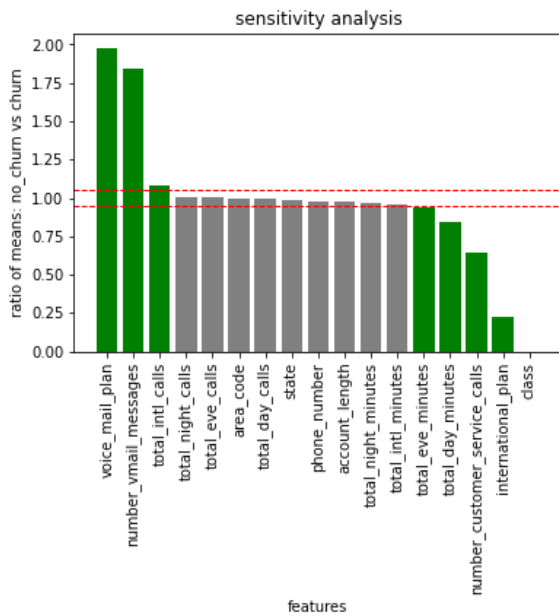


The classical intro to exploratory data analysis is the sns-pairplot. As is seen, there exist high correlation (line) in the relations of charge and minutes. These correlations are also seen in next correlations matrix below as high values red items. Some binary features (e.g. international plan yes/no lead to sparse lines without any correlation). These are expected as a relevant decision tree question.



Due to pay-as-use-tariffs, here: charge per minute, there is a clear correlation of charges and minutes.

The highly correlated features charges were excluded, minutes were kept.

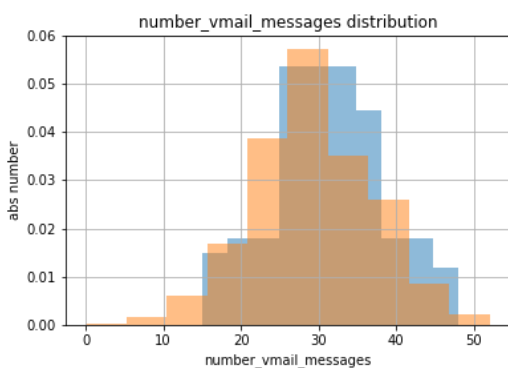


In order to filter/rank some of the features, we simply performed a sensitivity analysis by grouping according to class churn, and build the ratio of feature means:

$$\text{mean (no_churn)} / \text{mean (churn)}$$

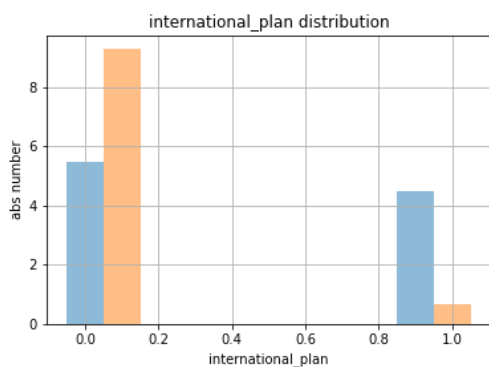
which results in some features with high effects > 1.05 or < 0.95 (as **green**), while **other features** have similar averages in their distributions.

- no_churn > churn:
 - voice_mail_plan
 - number_vmail_messages
 but also correlated
- slight influence: total_intl_calls
- no_churn < churn:
 - total_eve_minutes
 - total_day_minutes
 - number_customer_service_calls
 - international_plan



Since the other features were discussed later in the results chapter, on the left, the distributions of the “number of vmail messages” is depicted with clear thresholds for **churn** or **no_churn**

#vmail_messages <= 30: true: no churn
#vmail_messages <= 30: false: churn



Interestingly, customers without international plan churn less, which might come from some mis-alignment of international charges.

As a result, the charges per minute (27ct/min) do not differ for customers with or without international plan, which would obviously be a reason for churn in case of additional charges of international plan.

2.2. Feature Selection

TASK: Pre-process the data and do feature selection to extract the most important features.

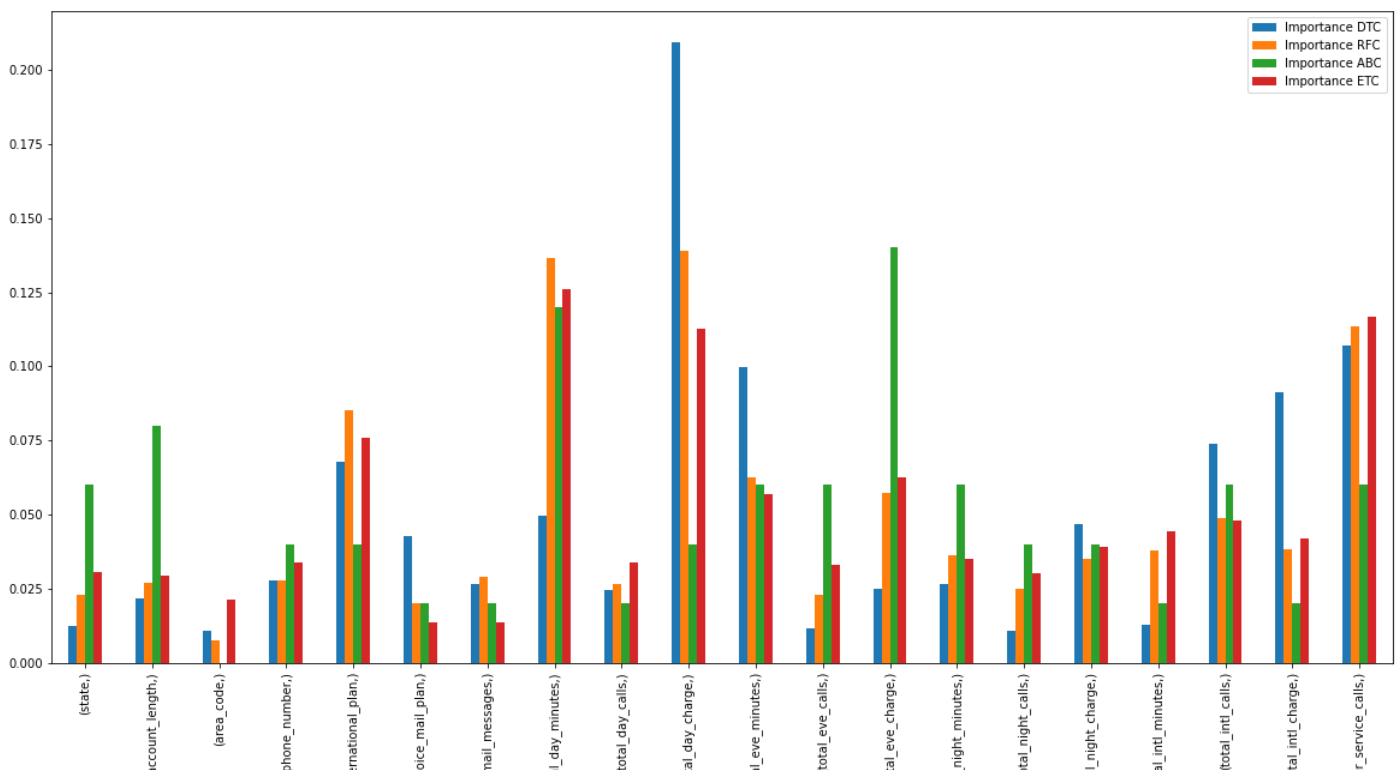
We compared 3 different ways of feature selection:

- Feature Extraction with RFE
- Feature Extraction with PCA
- Feature Importance with different Classifiers

For us, the most efficient way seemed to be the Feature Importance and the different comparable Outputs.

We compared the Feature Importance of following Classifiers:

- Decision Tree Classifier (DTC)
- Random Forest Classifier (RFC)
- Adaboost Classifier (ABC)
- Extra Trees Classifier (ETC)



After comparing the different outputs and checking the training time of some first early models, we decided against a feature reduction based on any of these processes.

What we are removing, are linear correlated values (charge/minutes).

The small amount of given data allows us to use all source data, without losing a lot of time.

2.3. Data Preparation

Instead of reducing the given data, we opted for a different approach. We set up different bulks of data with different structures, to compare the different outputs.

So we set up following different data bulks:

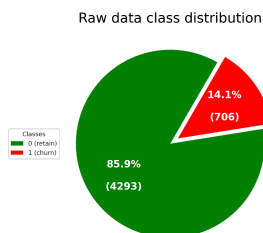
- original
 - the complete data set as provided
- dropped
 - the original data set minus some features:
 - state / area_code / phone_number / total_day_charge / total_eve_charge / total_night_charge / total_intl_charge
- scaled
 - original data set which received a scaling (standard Scaler) on some features:
 - account_length / number_customer_service_calls / number_vmail_messages / total_day_minutes / total_day_calls / total_day_charge / total_eve_minutes / total_eve_calls / total_eve_charge / total_night_minutes / total_night_calls / total_night_charge / total_intl_minutes / total_intl_calls / total_intl_charge
- encoded
 - original data set where specific features have been “one hot” encoded
 - state / area_code
- encoded and scaled
 - a mix of encoded and scaled data set
- scaled and dropped
 - a mix of scaled and dropped data set

There is no “scaled / encoded / dropped” data set, since all encoded features are dropped from the data set.

3. Results

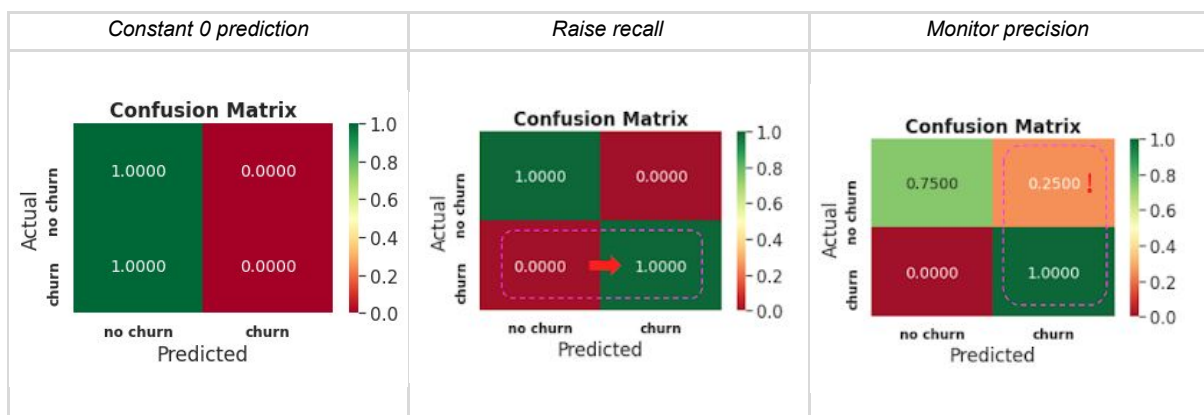
3.1. Metrics

The given task is a **binary classification** with the following class labels:



- 0 : retained customer
- 1 : churned customer

Raw data show an **imbalanced class distribution** with a much higher amount of retaining customers. Since we have an imbalanced class distribution accuracy is not the optimal metric. E.g. if a model would predict always 0 (no churn), we already have approx. 0.86 accuracy but would totally fail to predict churns.



It is necessary to observe a **high recall value** i.e. to have nearly all churners in the set of customers identified for potential churn.

Unfortunately this could also increase the number of retaining customers seen as churners by mistake, this could be crucial in further handling churners if the number of retaining customers seen as churners is significantly higher than the real customers. So also the **precision must be monitored** to have an acceptable number of customers being seen as churners but are not.

3.2. Standard Classifiers

TASK:

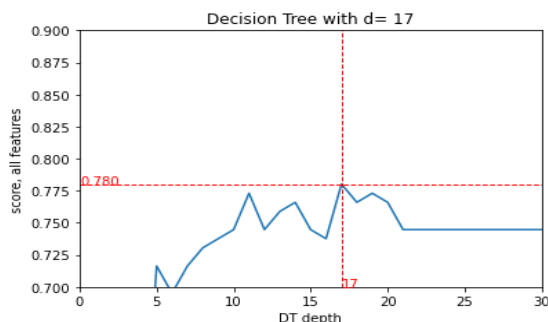
Classification method of your choice

- Use a classifier of your choice to classify the records into one of the two categories.
- Analyze the impact of hyperparameters on the performance of the model on validation and test sets.
- Analyze the outcomes.

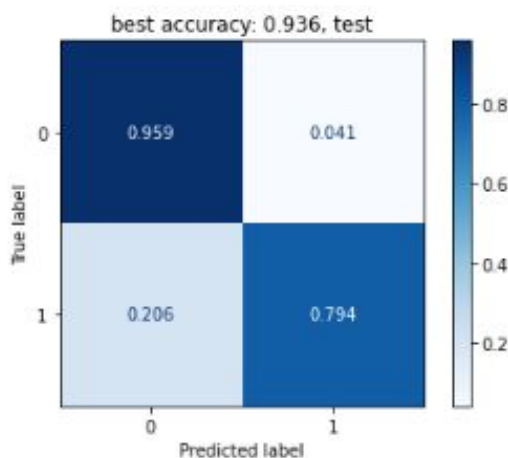
Different algorithms were investigated in the project:

- classical classifier with individual parameterization:
 - logistic regression, sigmoid
 - k-Nearest Neighbour
 - Decision Tree
 - Random Forest
 - Multinomial Bayes
 - Gradient Boost
 - XGBoost
 - SVC

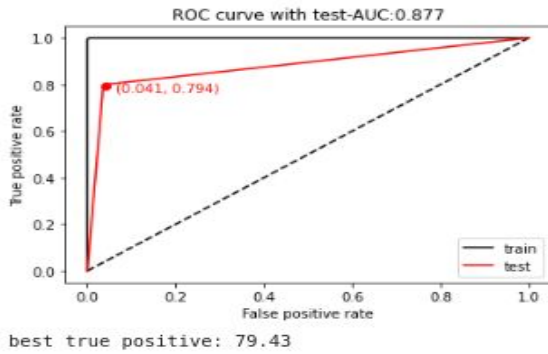
The decision tree classifier received best results.



The hyperparameter tuning of the DT classifier comes up with a degree of 17, slightly higher than the input feature dimension of 13.

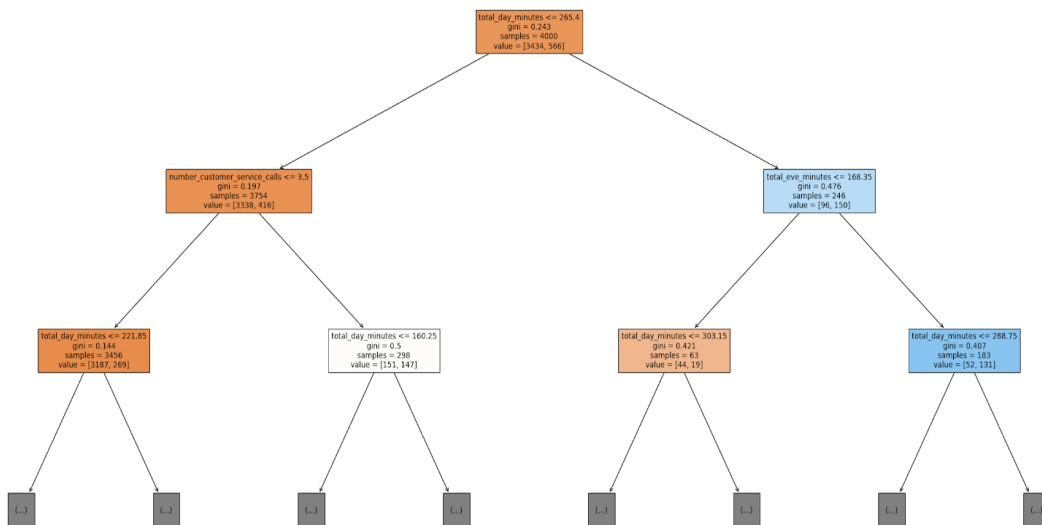
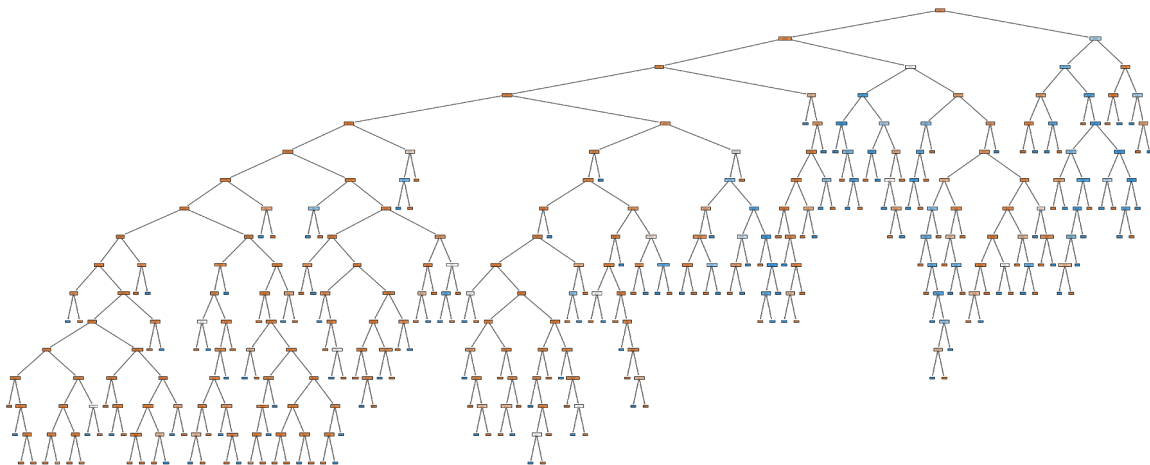


The classifier comes with an overall test set **accuracy of 93.6%** and a **true positive rate of 79.4%** which predicts a high portion of the churn customers correctly. Still ~20% were classified as no-churners despite they were churned customers. These 28 customers will be lost. On the opposite, 869* 4%= 35 customers are misclassified as potential churners, which would get some gratifications to stay. **It needs to be weighted externally by Telekom provider due to costs, image factors, etc. which “error” is more accepted.**



On the left, an alternative visualization is shown as ROC plot with the ideal target prediction as black line with the top-most left corner as the ideal point.

The area below the curve is 0.877 which is close to optimal value 1.0, compared to the poor 0.5 as dice-solution.



On the top, the full Decision tree is plotted, but not readable.

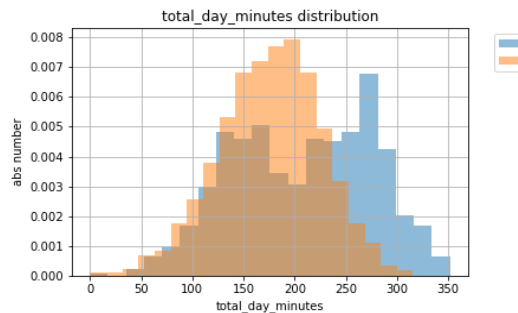
So below, the top 3 layers were depicted.

The main 3 decisions are discussed in the following

1. : total day minutes <= 265.4

1.1 : number customer calls <= 3.5

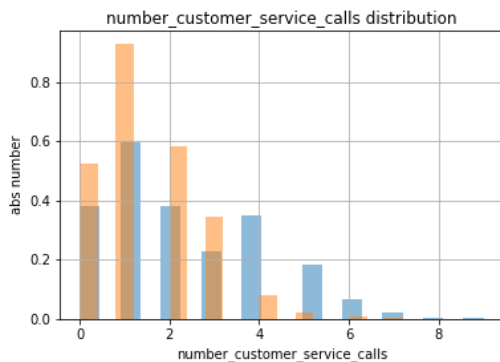
2.1 : total eve minutes <= 168.35



1. : **total day minutes** ≤ 265.4

This decision is valid since also seen in the statistics of **no_churn** vs **churn** customers.

This result is reasonable, since customers with higher minutes pay accordingly more and might churn for cheaper offers. Potential caps in charges or flat-offers might keep customers.

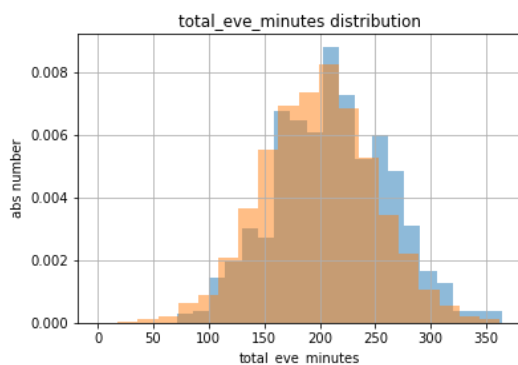


1.1 : **number customer calls** ≤ 3.5

This decision is valid since also seen in the statistics of **no_churn** vs **churn** customers.

If a customer calls less than 3.5 times the hotline to complain, they could be identified as “happy”, so they stay.

On the opposite, “angry” customers call more often hotline to complain, and leave in case of >3 calls



2.1 : **total eve minutes** ≤ 168.35

This decision is valid since also seen in the statistics of **no_churn** vs **churn** customers.

Again, this result is reasonable, since customers with higher minutes pay accordingly more and might churn for cheaper offers.

The following decisions also repeatedly focussed on different thresholds of minutes and customer service calls, as well as other features, but have not been validated here.

We also investigated, to include or exclude the ‘state’ as a feature. The feature importance of ‘state’ is low (1%). Only ‘voice_mail_plan’ is lower with 0.35%.

As expected, highest importance is with ‘total_day_minutes’ (26.7%), followed by ‘total_eve_minutes’ (15%) and ‘number_customer_service_calls’ (11%)

3.3. Neural Network Classifier

TASK:

Feed Forward based classification

- Build a feed forward neural network to classify the records into one of the two categories.
- Compare the performance of different activation functions and loss functions on the proposed task.
- Analyze the outcomes.

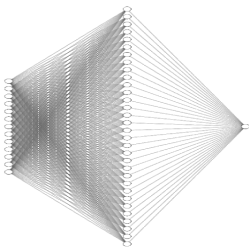
Used dataset: The performance of the neural network is better with scaled data. Additionally simple tests showed no negative impact if the reduced data set is used. Another important point is to improve runtime since fitting of our neural network consumes for each run several minutes of time.

*I.e. scaled and reduced data was used for further analysis. **

Loss and activation function: Because we have a binary classification problem *binary cross entropy loss function was used for all further investigations*. Instead of this other combinations of hidden layers are analyzed. Activation in hidden layers was tried with sigmoid instead of relu but the result is a constant prediction of 0 which is not desired. *So relu is always used in hidden layers. **

*See jupyter notebook *Neural_Network_classification_simple*

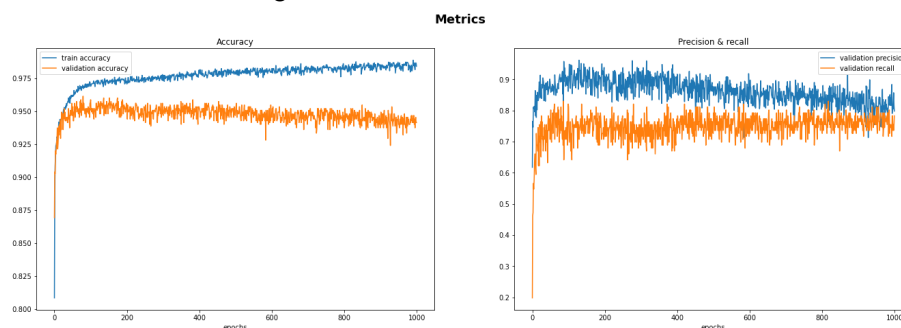
3.3.1. Simple Neural Network



As a first try a simple fully connected neural network was set up. The hidden layer was chosen with 32 neurons and activation relu and a single output was done with sigmoid activation for binary classification.

Layer	units	activation
Input layer	13 (reduced features)	-
Hidden layer	32	relu
Output layer	1	sigmoid

The learning curve and metrics (see chapter [Metrics](#)) for 1000 epochs are given below. Obviously our model overfits soon with decreasing accuracy for validation set but validation sets precision and recall converge.

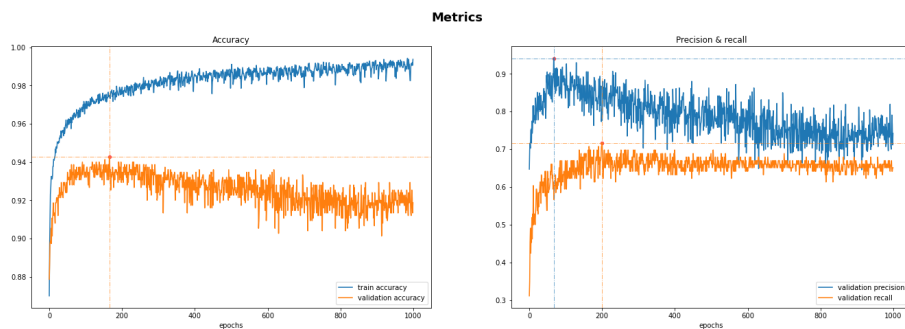


jupyter notebook: *Neural_Network_classification*

3.3.2. Hidden layer variations

As a special task the performance for different hidden layer sizes and more hidden layers was analyzed by choosing the “optimal” model by different metrics with the following approach:

- Different model architectures were trained over 800 epochs
- For each architecture the corresponding model was taken for
 - minimum loss
 - maximum accuracy
 - maximum recall
 - maximum precision

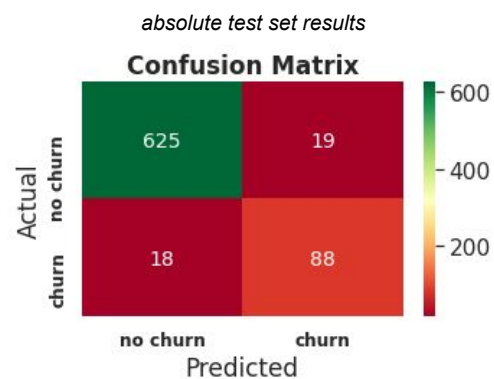
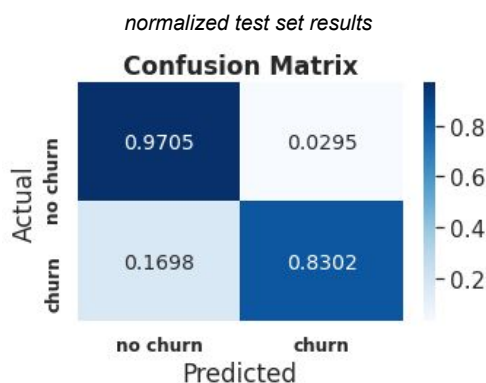


- The performance metrics for each of the models were measured on the test set

One hidden layer with different amount of units / Random hidden layer combinations:
(combinations of 0 Recall >0.98 and high 1 Recall >0.77, see appendix for more data and combinations)

Model-hidden layers	Model taken with best parameter	Accuracy	0 Precision	0 Recall	1 Precision	1 Recall
[16]	Loss	0.952000	0.963415	0.981366	0.872340	0.773585
[16]	Accuracy	0.954667	0.966361	0.981366	0.875000	0.792453
[32]	Accuracy	0.953333	0.963470	0.982919	0.881720	0.773585
[128]	Accuracy	0.956000	0.963581	0.986025	0.901099	0.773585
[256]	Precision	0.956000	0.963581	0.986025	0.901099	0.773585
(128, 32, 64)	Accuracy	0.953333	0.964885	0.981366	0.873684	0.783019
(128, 32, 128)	Recall	0.957333	0.972222	0.978261	0.862745	0.830189
(64, 64, 32, 32)	Accuracy	0.956000	0.969278	0.979814	0.868687	0.811321

The results for an **input-128-32-128-1** fully connected neural network with best recall outperforms the decision tree classifier for churn prediction.



4. Conclusion

TASK:

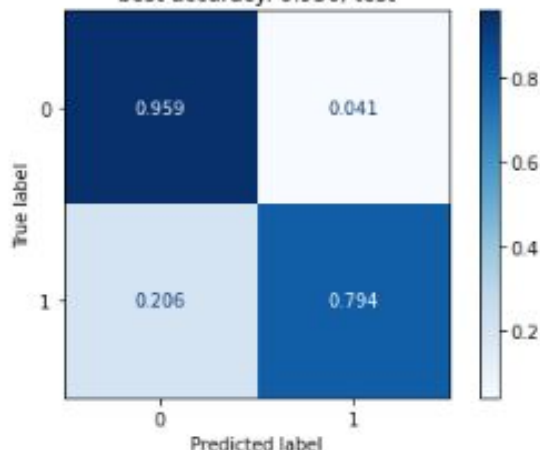
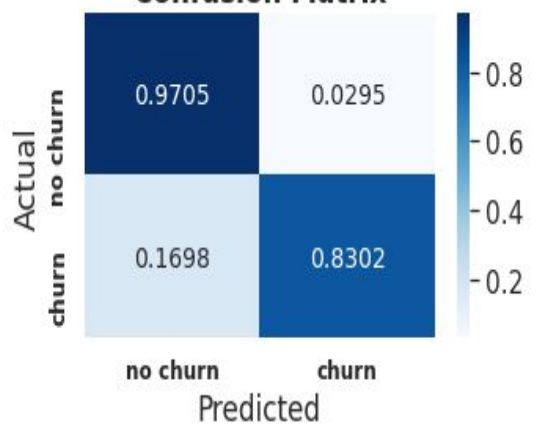
Compare the results from both classifiers from the accuracy point of view.

We are able to predict churners with different models.

Business impact

It is very important to keep in mind that a decreasing recall value for 0 (no churn) will have a high impact on absolute values of customers.

So it might be acceptable not to identify churners (lower true-positive value) if this would reduce the rate of wrong identified churners (also lower false-positive value).

Classical Classifier Fit	Neural Network																		
<div>best accuracy: 0.936, test</div>  <p>A confusion matrix for a classical classifier fit. The y-axis is labeled 'True label' with values 0 and 1. The x-axis is labeled 'Predicted label' with values 0 and 1. The matrix values are: (0,0) = 0.959, (0,1) = 0.041, (1,0) = 0.206, (1,1) = 0.794. A color bar on the right indicates values from 0.2 to 0.8.</p> <table><tr><td>True label \ Predicted label</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0.959</td><td>0.041</td></tr><tr><td>1</td><td>0.206</td><td>0.794</td></tr></table>	True label \ Predicted label	0	1	0	0.959	0.041	1	0.206	0.794	<div>Confusion Matrix</div>  <p>A confusion matrix for a neural network. The y-axis is labeled 'Actual' with values 'no churn' and 'churn'. The x-axis is labeled 'Predicted' with values 'no churn' and 'churn'. The matrix values are: (no churn, no churn) = 0.9705, (no churn, churn) = 0.0295, (churn, no churn) = 0.1698, (churn, churn) = 0.8302. A color bar on the right indicates values from 0.2 to 0.8.</p> <table><tr><td>Actual \ Predicted</td><td>no churn</td><td>churn</td></tr><tr><td>no churn</td><td>0.9705</td><td>0.0295</td></tr><tr><td>churn</td><td>0.1698</td><td>0.8302</td></tr></table>	Actual \ Predicted	no churn	churn	no churn	0.9705	0.0295	churn	0.1698	0.8302
True label \ Predicted label	0	1																	
0	0.959	0.041																	
1	0.206	0.794																	
Actual \ Predicted	no churn	churn																	
no churn	0.9705	0.0295																	
churn	0.1698	0.8302																	
<p>The DT classifier comes with an overall test set accuracy of 93.6% and a true positive rate of 79.4% which predicts a high portion of the churn customers correctly. Still ~20% were classified as no-churners despite they were churned customers. These 28 customers will be lost. On the opposite, $869 \cdot 4\% = 35$ customers are misclassified as potential churners, which would get some gratifications to stay. <u>It needs to be weighted externally by Telekom provider due to costs, image factors, etc. which “error” is more accepted.</u></p>	<p>With neural networks a better accuracy compared to decision trees can be reached. Also a better recall rate is possible.</p> <p>Thus a neural network is recommended for predictions.</p> <p>In general it is possible to reduce the false positives and reach a better precision for churn predictions but this reduces the recall rate.</p> <p>This decision can be done by the business owner.</p>																		

Appendix on NN results

hidden layer neurons	Best Parameter	Accuracy	0 Precision	0 Recall	1 Precision	1 Recall
[16]	Loss	0.952000	0.963415	0.981366	0.872340	0.773585
[16]	Accuracy	0.954667	0.966361	0.981366	0.875000	0.792453
[16]	Precision	0.949333	0.959091	0.982919	0.877778	0.745283
[16]	Recall	0.948000	0.971919	0.967391	0.807339	0.830189
[32]	Loss	0.928000	0.962382	0.953416	0.732143	0.773585
[32]	Accuracy	0.953333	0.963470	0.982919	0.881720	0.773585
[32]	Precision	0.948000	0.956259	0.984472	0.885057	0.726415
[32]	Recall	0.918667	0.967897	0.936335	0.677165	0.811321
[64]	Loss	0.925333	0.959375	0.953416	0.727273	0.754717
[64]	Accuracy	0.949333	0.963303	0.978261	0.854167	0.773585
[64]	Precision	0.948000	0.954887	0.986025	0.894118	0.716981
[64]	Recall	0.942667	0.960184	0.973602	0.824742	0.754717
[128]	Loss	0.930667	0.952599	0.967391	0.781250	0.707547
[128]	Accuracy	0.956000	0.963581	0.986025	0.901099	0.773585
[128]	Precision	0.953333	0.960666	0.986025	0.898876	0.754717
[128]	Recall	0.909333	0.970588	0.922360	0.637681	0.830189
[256]	Loss	0.944000	0.953313	0.982919	0.872093	0.707547
[256]	Accuracy	0.949333	0.963303	0.978261	0.854167	0.773585
[256]	Precision	0.956000	0.963581	0.986025	0.901099	0.773585
[256]	Recall	0.948000	0.961832	0.978261	0.852632	0.764151

Model-hidden layers	Best Parameter	Accuracy	0 Precision	0 Recall	1 Precision	1 Recall
(128, 32)	Loss	0.936000	0.957055	0.968944	0.795918	0.735849
(128, 32)	Accuracy	0.940000	0.967239	0.962733	0.779817	0.801887
(128, 32)	Precision	0.938667	0.937135	0.995342	0.954545	0.594340
(128, 32)	Recall	0.945333	0.970359	0.965839	0.798165	0.820755
(128, 128)	Loss	0.942667	0.955994	0.978261	0.846154	0.726415
(128, 128)	Accuracy	0.953333	0.966309	0.979814	0.865979	0.792453
(128, 128)	Precision	0.940000	0.945022	0.987578	0.896104	0.650943
(128, 128)	Recall	0.937333	0.971564	0.954969	0.752137	0.830189
(64, 32)	Loss	0.920000	0.952012	0.954969	0.721154	0.707547
(64, 32)	Accuracy	0.945333	0.956127	0.981366	0.865169	0.726415
(64, 32)	Precision	0.940000	0.942393	0.990683	0.917808	0.632075
(64, 32)	Recall	0.928000	0.959502	0.956522	0.740741	0.754717
(64, 64, 64)	Loss	0.944000	0.960245	0.975155	0.833333	0.754717
(64, 64, 64)	Accuracy	0.952000	0.966258	0.978261	0.857143	0.792453
(64, 64, 64)	Precision	0.956000	0.958021	0.992236	0.939759	0.735849
(64, 64, 64)	Recall	0.948000	0.966102	0.973602	0.831683	0.792453
(128, 32, 64)	Loss	0.944000	0.961656	0.973602	0.826531	0.764151
(128, 32, 64)	Accuracy	0.953333	0.964885	0.981366	0.873684	0.783019
(128, 32, 64)	Precision	0.945333	0.945347	0.993789	0.945205	0.650943
(128, 32, 64)	Recall	0.908000	0.970540	0.920807	0.633094	0.830189
(128, 32, 128)	Loss	0.937333	0.961360	0.965839	0.786408	0.764151
(128, 32, 128)	Accuracy	0.953333	0.966309	0.979814	0.865979	0.792453
(128, 32, 128)	Precision	0.944000	0.954683	0.981366	0.863636	0.716981
(128, 32, 128)	Recall	0.957333	0.972222	0.978261	0.862745	0.830189
(32, 128, 32, 64)	Loss	0.941333	0.954545	0.978261	0.844444	0.716981
(32, 128, 32, 64)	Accuracy	0.954667	0.960725	0.987578	0.909091	0.754717
(32, 128, 32, 64)	Precision	0.954667	0.960725	0.987578	0.909091	0.754717
(32, 128, 32, 64)	Recall	0.930667	0.968354	0.950311	0.728814	0.811321
(64, 64, 32, 32)	Loss	0.940000	0.960061	0.970497	0.808081	0.754717
(64, 64, 32, 32)	Accuracy	0.956000	0.969278	0.979814	0.868687	0.811321
(64, 64, 32, 32)	Precision	0.952000	0.959215	0.986025	0.897727	0.745283
(64, 64, 32, 32)	Recall	0.934667	0.976000	0.947205	0.728000	0.858491
(64, 32, 64, 32)	Loss	0.938667	0.958589	0.970497	0.806122	0.745283
(64, 32, 64, 32)	Accuracy	0.956000	0.962179	0.987578	0.910112	0.764151
(64, 32, 64, 32)	Precision	0.956000	0.962179	0.987578	0.910112	0.764151

Best True Positive values:

AIDA-Project - Team C
Dietert, Habermann, Sauer

Telekom Customer Churn Prediction

structure	activations	1 st round	2 nd round	3 rd round
20, 1	['relu', 'sigmoid']	0.511	0.525	0.454
20, 20, 1	['relu', 'relu', 'sigmoid']	0.695	0.603	0.624
20, 20, 20, 1	['relu', 'relu', 'relu', 'sigmoid']	0.752	0.504	0.660
20, 20, 20, 20, 1	['relu', 'relu', 'relu', 'relu', 'sigmoid']	0.560	0.617	0.695
20, 20, 20, 20, 20, 1	'relu', 'relu', 'relu', 'relu', 'relu', 'sigmoid'	0.617	0.592	0.645
20, 20, 20, 20, 20, 20, 1	'relu', 'relu', 'relu', 'relu', 'relu', 'relu', 'sigmoid'	0.567	0.511	0.667

Best True Positive values:

structure	activations	1 st round	2 nd round	3 rd round
20, 1	['relu', 'sigmoid']	0.610	0.475	0.546
20, 40, 1	['relu', 'relu', 'sigmoid']	0.667	0.585	0.702
20, 40, 20, 1	['relu', 'relu', 'relu', 'sigmoid']	0.582	0.766	0.652
20, 40, 40, 20, 1	['relu', 'relu', 'relu', 'relu', 'sigmoid']	0.475	0.582	0.638
20, 40, 60, 40, 20, 1	'relu', 'relu', 'relu', 'relu', 'relu', 'sigmoid'	0.660	0.589	0.638
20, 40, 60, 60, 40, 20, 1	'relu', 'relu', 'relu', 'relu', 'relu', 'relu', 'sigmoid'	0.735	0.553	0.610

Best True Positive values:

structure	activations	1 st round	2 nd round	3 rd round
40, 1	['relu', 'sigmoid']	0.270	0.475	0.489
40, 40, 1	['relu', 'relu', 'sigmoid']	0.631	0.539	0.525
40, 40, 40, 1	['relu', 'relu', 'relu', 'sigmoid']	0.475	0.546	0.652
40, 40, 40, 40, 1	['relu', 'relu', 'relu', 'relu', 'sigmoid']	0.461	0.589	0.716
40, 40, 40, 40, 40, 1	'relu', 'relu', 'relu', 'relu', 'relu', 'sigmoid'	0.461	0.528	0.695
40, 40, 40, 40, 40, 40, 1	'relu', 'relu', 'relu', 'relu', 'relu', 'relu', 'sigmoid'	0.709	0.560	0.730

Feature Importance Overview

	DTC	RFC	ABC	ETC
state	0.017426	0.021991	0.06	0.027664
account_length	0.007132	0.007291	0.00	0.023061
area_code	0.028268	0.028290	0.04	0.027523
phone_number	0.067817	0.085376	0.04	0.069619
international_plan	0.000000	0.022899	0.02	0.029929
voice_mail_plan	0.020408	0.027820	0.08	0.033842
number_vmail_messages	0.103851	0.112297	0.06	0.119524
total_day_minutes	0.065863	0.023728	0.02	0.014229
total_day_calls	0.141229	0.137749	0.10	0.130760
total_day_charge	0.022365	0.025234	0.02	0.029411
total_eve_minutes	0.116979	0.144717	0.06	0.104908
total_eve_calls	0.076709	0.058538	0.10	0.051946
total_eve_charge	0.014940	0.024173	0.06	0.033247
total_night_minutes	0.053767	0.059020	0.10	0.053016
total_night_calls	0.036199	0.037496	0.06	0.037787
total_night_charge	0.013697	0.025316	0.04	0.030800
total_intl_minutes	0.032745	0.036582	0.04	0.040350
total_intl_calls	0.012743	0.039513	0.02	0.045121
total_intl_charge	0.076629	0.047136	0.06	0.051993
number_customer_service_calls	0.091232	0.034835	0.02	0.045269