AI Assignment 1

NATHANIEL ANDRE K. THOMAS-COPELAND (SECTION A)

MIBA 2022-2023

1. Executive summary

- The goal of this project was to limit costs that the bank are suffering from due to churning customers
- In order to remedy this problem, we have decided to turn to machine learning in order to create a model which would be able to predict churning customers in time and retain them (which is 7 times cheaper than acquiring a new customer)
- In order to do so, we trained two types of models: an Ensemble/Random Decision Forest model and a Deepnet model
- As our goal was to detect as many churning customers as possible, the performance metric we were pushing to optimize in our models was the Recall measure
- After carefully calculating the profits which would result from implementing both the optimized Deepnet and Ensemble models, we decided to select and propose the implementation of the Deepnet model. Both models allow to create higher profits than our current situation
- Aditionnaly, our models have helped us identify the key features that determine whether a client churns or not. The telling features we have detected are the age, the country of the client, the number of products he/she possesses, and his/her bank balance. Based on this, we have implemented business solutions that leverage these insights to diminish the churn rate

2. Creating the model

1. The Problem : what is it? What do we want to improve?

- The European bank is is losing clients (churn rate= 20.37%) and are currently spending 70 € per client (=Customer Acquisition Cost) to replace them, whereas the Customer Retention Cost is of 10 €.
- **Therefore, the bank is looking for a way to detect departing clients in time in order to save money in order to cut costs** and allocate the remainder of their spending budget towards growing their clientele and revenues.
- This would allow to dedicate minimize costs, maximize profits and the growth of the clientele.

2. Which kind of model will be used? What kind of problem?

- The type of model used will be a **Supervised model**, as we will be using past data of customers with certain characteristics (=inputs) that have left (=target output) in order to identify patterns and relationships between these variables to predict whether a customer is likely to leave (=target output).
- Because the Target (variable) is categorical (whether the client will leave or not), **we are solving a Classification problem**.

3. Is the interpretability of the ML model important in this context?

- **The interpretability of the ML model in this situation is not the priority but remains a valuable aspect of our model**. In this case, the goal is to detect as many churning customers as possible and not to understand why clients are leaving.

4. Does any variable have a strange distribution? Are there suspicious/corrupted values? Are there missing values or outliers?

- **Variables with strange distribution:**

Balance: 36.18% of the population have bank balances between 0 and 10,000€, and then another 44,29% of instances have bank account balances between 90,000€ and 150,000€.

- **Suspicious/corrupted values:**

Average day transactions: 2 instances make between 1,000 and 1,500 transactions per day, 1 between 3,000 and 3,5000 a day and 4 over 4000 a day → 0.07% of total instances (see appendix figure 1)

Average day expenditure: 2 extreme outliers above 500€ (see appendix figure 2)

4. How did we deal with these problematic values?

- **Suspicious/corrupted values:**

Anomalies : used BigML's personalized anomalies detector to identify anomalies above 60% anomaly score (see appendix figure 5 & 6)

- **Missing values:**

None

- **Outliers:**

Common practice: assumed that values below and above 99th percentile are outliers. (see appendix figure 3 & 4)

4. How did we deal with these problematic values?

- **Numerical rescaling:**

In order to avoid biases due to the different scales of the numerical values, we have decided to normalize all the numerical values for them to be scaled between 0 and 1. To do this, we used BigML's 'add field feature' and selected the Normalization function (see appendix figure 7).

IMPORTANT: anomalies, outlier removal was carried out on the training set only to avoid data leakage. The normalization was done on the training set and test set separately to avoid target leakage. however, the same normalization parameters were used (values between 0 and 1).

5. Was there any feature causing a Target leakage?

- None of the features in the dataset caused Target leakage. All of the feature values can be determined before the customer churns.

6. Were all the features useful to predict the target variable?

- Yes. the features which we have deemed useless are the following : Surname, CustomerId, RowNumber. These features were removed during the dataset creation (view appendix figure 8).

7. Perform the train-test split. Which percentages did you choose?

- **Percentage split - 80/20 split (see figure):**

We use this measure as it is common practice. This split is inspired from the Pareto Principle (also called the 80/20 rule). The general point is that, in most cases, 80% of effects come from 20% of causes. While the preface of the principle was based on wealth distribution, it statistically does come close to explaining many human, machine, and environmental phenomena. For this reason, and typically without knowledge of the source, many use the 80/20 split for training and testing.

- **The share of churned clients (see figure)**

The share for the test set is of 20.75% and of 19.28% (see figure) for the training set. It's important that both datasets present the same share of positive cases, as if not the model will be training to predict instances that occur at different frequencies in real life than in the training set. This will result in a biased model, and biased predictions.

8. Simple model training: brief metrics & feature importance analysis

- 84.25% Accuracy : The accuracy represents the rate at which the model's overall predictions concerning whether a client would churn were correct
- 64.45% Precision: out of all the the predictions that a client was going to churn made by the model, it was correct 64.45% of the time
- 53.73% Recall: out of all the customers that churned in the test set, the model accurately identified 53.73%
- ROC AUC = 0.8436 : the model falsely predicts a customer will churn 7.76 % of the time, and detects that a customer will leave 53.73% of the time
- The features with the highest predictive power were Age (18.13 %), NumOfProd (14.83 %), and Balance (13.86%)
- Least important features are HasCrCard (0.51%), Customer service calls (3.54%), Average day transactions (4.37%)

9. Feature engineering: Could new features be created by combining the original ones? If so, did these have a higher predictive power?

We tried to add 2 features which were combinations of original features:

Expenditure by transaction ratio performance (based on intuition): how much a customer (on average) spends per transaction (average) in a day (see appendix figure 13)

- When analyzing explaining power of the variable for the model created, we can see that the explaining power of the new variable (7.14%) is higher than the two it replaced (5.37% for average day expenditure and 4.37% for average day transactions), but doesn't make it one of the most explanative variables (see appendix figure 15)

9. Feature engineering: Could new features be created by combining the original ones? If so, did these have a higher predictive power?

2. BalancexNOP = Balance*NumOfProd (see appendix figures 18-21):

- After analyzing the correlation between each variable in the training set using the scatterplot feature, we noticed a strong correlation between Balance and NumOfProd (See appendix figure 18)

- This combination (14.59% field importance) did result in a higher predictive power than NumOfProd (13.86% field) had in the original model but not higher than Balance (14.83%)

10. Did we do feature selection using the remaining variables we decided to go forward with?

We decided to remove the following features in reason of their consistently low field importance (See figure 22):

- Customer service calls: low predictive power
- HasCrCard: low predictive power

11. If applicable, did the new features or the removal of some improve the performance of your model?

- When compared with our first model (see appendix figure 12), removing HasCrCard and Customer service calls improved the precision, recall, and ROC AUCH (see appendix figure 23)

- When comparing the overall performance of the model with the first simple model created (see appendix figure 12), we can notice that including the BalancexNOP variable does not improve the performance of the model (lower precision, recall and accuracy, and ROC AUCH). Therefore, we decided not to include BalancexNOP going forward and to keep the original variables

- Regarding overall model performance using the new variable, this model has slightly lower precision, recall and accuracy than the simple model we first created. For this reason, we opted against using this feature in our model going forward (see appendix figure 16 & 17)

- **In order to compare the performances of each model discussed, we identified the positive class as when the target variable (Excited) was equal to 1**

12. Training another model with a different algorithm

- In order to compare our algorithms, we decided to look at the following metrics: Accuracy, Precision, and Recall. We also analyzed and compared the performances of each model on the same test set in order to assess which has a more favorable behavior for us

- We decided to create a Decision Net (see appendix figure 25) model in order to evaluate whether our algorithm choice was the right one (see appendix figure 23)

- When comparing both models, we noticed the Deep Net presented significantly lower Precision (69%) and Accuracy (38.7%) than the model we created using the Random Decision Forest algorithm (84.4% and 64.27% respectively). However, the Deepnet model's Recall (84.58%) and ROC AUCH (0.8613) are higher for the Deepnet model than the Random Decision Forest (62.17% and 0.8503 respectively)

12. Training another model with a different algorithm

- When comparing the behaviors of both models on the same test set, the deepnet identified less False Negatives (churning customers that the model predicted would stay), however it also identified an alarmingly high number of False Positives (staying customers that the model predicted would churn). In the case of our Randomized Decision Forest, it identified more False Negatives but significantly less False Positives. Even though False Negatives cost significantly more to the bank than false positives (since they force us to pay 70 euros to replace the departed client, when a False Positives only makes us pay 10 euros), the massive share of False positives identified by the Deepnet, and overall low accuracy which results in low TP and TN identification rates makes the Deepnet model less interesting than the Randomized Decision forest model.

12. The impact of using K-fold Cross-validation and why do we use it

- We decided to use k-fold cross validation for both the Deepnet model and the Randomized Decision Forest in order to add robustness to our results. Instead of training the model using a single training set/ test set split (as we have been doing until now), the dataset is split into 5 different test set/training set splits which will produce 5 different models with their own performance metrics
- The average of the performance of the 5 subsequent models will be the performance of the final model. This method will help prevent overfitting, which can happen when you only build a model using 1 training set
- Cross validated models tend to perform better, as they are trained on multiple datasets and their performance tends to translate better to new data

12. The impact of using K-fold cross validation and why do we use it

- **In the case of the Deepnet (see appendix figure 27),** we can observe that using cross validation has significantly improved the accuracy of the model (from 69% to 80.03%) and precision has improved considerably well (from 38.7% to 51.56%). Subsequently, the F-measure for Deepnet using cross-validation is higher as well (from 0.531 to 0.6003). However, the recall for our new model is lower than that of our original Deepnet (from 84.58% to 73.20%)

- In practical terms, the cross-validated Deepnet will identify more churning and non-churning customers. It will also misidentify more churning customers as staying (as the recall is lower) but will also predict fewer staying customers as staying (as precision has improved as well).

12. The impact of using K-fold cross validation and why do we use it

- **In the case of the Random Decision Forest (see appendix figure 29),** we can observe that the using cross validation has slightly increased the accuracy (from 84.4% to 85.71%) and has significantly increased precision (from 62.47% to 72.73%). Recall has diminished noticeably (from 62.17% to 42.91%) and the F-Measure for the cross validated model is also lower than that of the regular Random Decision Forest (from 0.6232 to 0.5495).

- In practical terms, these differences mean that given the same dataset, the cross-validated Random Decision Forest model will more accurately identify churning and non-churning customers (higher accuracy). It will also misidentify more churning customers as staying (as the recall is lower) but will also predict fewer staying customers as churning (as precision has increased)

13. Model optimization

In order to attain our final models, we ran a series of models and toggled the hyperparamters according to our results in order to obtain models with which we were satisfied. The final hyperparameters chosen for our models were the following :

- Deepnet model hyperparameter final selection:

auto structure suggestion, 2 hidden layers, adam, learning rate=0.01, 684-iteration, beta1=0.9, beta2=0.999, epsilon=1e-08, tree embedding=True, missing values, max. training time=879, balanced

- Random Decision Forest final hyperparameter selection:

random decision forest, 286-node, random candidate ratio: 0.7, 44-model, deterministic order, balanced

14. Is this new model overfitted?

- Overfitting occurs a given model is trained to work perfectly on the training set only. **This entails a drastic dip in performance evaluation when this model is applied to an unknown dataset**

- In order to prevent overfitting, we used Cross-validation as the Evaluation model for both the Deepnet and Ensemble models. Using Cross-validation allows for the models to be trained on multiple splits of the dataset separately and trained on other splits of the dataset, which assures that the model works on "unknown datasets"

14. Is this new model overfitted?

- In order to further confirm that our models were not overfitted, we evaluated both of them using the 20% split of the dataset (as we conducted the Cross-validation on the training set only). This allowed us to use the 20% split as a "new dataset" to evaluate how well our models would perform
- Although their performance was lower (which is to be expected), both of our models kept similar performance levels, which further proves that both models are not overfitted (see figures 34-35 & figure 39)

15. Feature interpretation

The features with the highest predictive power were **Age, NumOfProducts, Balance, and Geography** for both the Random Decision Forest and Deepnet Models (see appendix figure 33 and 38)

- Age plays a significant role in the determining whether a customer will churn or not. According to dataset analysis, adults aged between 29 and 44 are the most common churners
- We also believe that NumOfProducts's presence amongst the variables with the highest predictive power, as there is a clear trend that only customers with between 2 and 4 products churn
- Balance's high predictive power is mostly related to its high correlation with the NumOfProducts variable, as no distinguishable trend was identified between the Balance and the target variable
- Finally, the Geography attribute's influence comes as no surprise, as data analysis shows certain countries (such as Germany) have a very high proportion of churning customers

16. Interpreting the confusion matrix of the Random Decision Forest

- **TP (TRUE POSITIVE):**

Describes an instance accurately predicted as the positive instance. In our case, this would translate into a customer which our model correctly predicts will churn.

- **TN (TRUE NEGATIVE):**

Describes an instance accurately predicted as a negative instance. In our case, this would translate into a customer which our model correctly predicts won't churn.

- **FP (FALSE POSITIVE):**

Describes an instance which the model predicts to be a positive instance when it is actually a negative instance. In our case, this would translate into a customer which our model falsely identifies as churning.

- **FN (FALSE NEGATIVE):**

Describes an instance which the model predicts to be a negative instance when it is in reality a positive instance. In our case, this would translate into a customer which our model falsely identifies as a staying customer.

16. Interpreting the confusion matrix of the Random Decision Forest

- **Accuracy:**

The accuracy metric reflects the rate at which the model correctly identifies both classes of the target variable. It is translated into the following formula:

$$\textbf{Accuracy} = (TP + TN)/(TN + TP + FP + FN)$$

In business terms (for this case), the accuracy would predict the rate at which the model correctly identifies churning and non-churning customers over all the customers in the dataset.

- **Recall:**

The recall metric reflects the percentage of correctly predicted true positive instances over the total amount of true positive instances. It is translated into the following formula:

$$\textbf{Recall} = TP/(TP+FN)$$

For this case, the recall is the percentage of churning customers predicted by the model over the total number of customers that actually churned

16. Interpreting the confusion matrix of the Random Decision Forest

- **Precision:**

The precision metric reflects which percentage of instances identified as True positives by the model are actual True Positives. This translates into the following formula:

Precision = TP/(TP+FP)

In our case, the precision would be equivalent to the percentage of the total customers that were predicted to churn by our model which actually churned

16. Interpreting the confusion matrix of the Random Decision Forest (see appendix figure 35)

- In the dataset used to evaluate this model, we can see that the total number of customers is of 2000 clients
- Of the total amount of Clients, the model mistakenly predicted 103 churning customers as staying (false negatives)
- Of the total amount of clients, the model mistakenly predicted 344 clients would churning when they were actually staying (false positives)
- The model correctly predicted 312 churning customers (true positives), as well as 1,241 staying customers (true negatives)

17. Which metric is most important to us?

- In our case, our goal is to predict as many churning customers as possible before they leave, as not catching them on time will cost us 70 euros

- This translates into the **Recall metric,** which represents the percentage of churning customers our model accurately predicted out of all churning customers

- **The F-1 Measure** is a way to evaluate the overall accuracy of a model. Its value is calculated through a harmonic mean of the Precision and Recall metrics. Despite it being a good way to measure accuracy, **the F-1 measure is generally considered a good metric to consider if False Negative instances and False Positive instances come at the same cost** (so if acquiring new customers came at the same cost as retaining existing customers). In our case, False Negatives (misidentifying churning clients) comes at a higher cost than misidentifying staying customers. This means our goal is to identify as many churning customers as possible (and thus reduce the amount of False Negatives)

18. Translating the components of our confusion matrix into financial results

Based on the information provided to us concerning the different prices and costs by the bank, we have attached the following values to each type of outcome of our confusion matrix:

Instance	Cost(€)	Revenue (€)	Total net impact (=value of each instance) (€)
True Positive	10	60	50
True Negative	0	60	60
False Negative	70	0	-70
False Positive	10	60	50

18. Translating the components of our confusion matrix into financial results

- Based on the information provided to us concerning the different prices and costs by the bank, we have attached the following values to each type of outcome of our confusion matrix:
- If the model correctly predicts a customer is leaving, the bank will pay 10€ to retrieve him and will receive 60€ or revenue. Therefore, the net impact of detecting a churning customer is of 50
- If the model correctly predicts a customer is staying, the bank will not pay anything to acquire/retain the customer. His cost will be of 0€ and he will receive 60€ of revenue. The net impact of accurately detecting a staying customer is of 60€
- If the model falsely predicts a customer is staying, the bank will not pay 10€ to retain him. Therefore, the client will leave, and the bank will have to pay 70€ to retrieve him. Therefore, the net impact of not detecting a leaving customer is of 70€
- If the model falsely predicts a customer is leaving, the bank will pay 10€ in an attempt to retain him. However, the customer is staying regardless, and the bank will receive its 60€ revenue. Therefore, the net impact of falsely detecting a staying customer is of 50€

19. Comparing the profits of both optimized models

- Given a customer sample of 2000 students, the net profits from implementing the Deepnet model (see figure 39) will have the following impact on revenues:

Instance	Number	Cost (€)	Revenue (€)	Total Cost (€)	Total revenue (€)	Net profit (€)
TP	338	10	60	3380	20,280	16,900
TN	1,050	0	60	0	63,000	63,000
FP	535	10	60	5350	32,100	26,750
FN	77	70	0	5,390	0	-5390
Total	2,000			14,120	115,380	101,260

- Assuming our model gave proportionate results for our full clientele, the net profit realized using this model would be of 5,063,000 euros

19. Comparing the profits of both optimized models

- Given a customer sample of 2000 students, the net profits from implementing the Random Decision Forest model (see figure 35) will have the following impact on revenues:

Instance	Number	Cost (€)	Revenue (€)	Total Cost (€)	Total revenue (€)	Net profit (€)
TP	312	10	60	3,120	18,720	15,600
TN	1,241	0	60	0	74,460	74,460
FP	344	10	60	3,440	20,640	17,200
FN	103	70	0	7,210	0	-7,210
Total	2,000					100.050

- Assuming our model gave proportionate results for our full clientele, the net profit realized using this model would be of 5,002,500 euros

19. Comparing the profits of both optimized models

- After comparing both models' revenues, we have concluded that the Deepnet model generates the highest profits
- Considering the current monthly profits of the bank are the following:

(1-0.2037)*100,000*60 – (0.2037)*70*100,000= 4777800-1425900 = 3,352,900€

- Therefore, the impact of implementing the Deepnet model will increase the revenues by:

5,063,000 – 3,352,900 = 1,710,100€

20. Suggested business activities based on the insights provided by model

- **Create age-targeted marketing campaigns:** our models have revealed the importance of the age of our customers in predicting whether they'll churn. When analyzing the age of churning clients, we realized that most of them were aged between 35 and 45 on average. Therefore, creating targeted ads for this age group could help reduce the churn rate
- **Prevent clients from owning too many products:** our models have also demonstrated the predictive power of the number of products in determining whether a client will churn. After taking a closer look at the data, we realized that clients that had between 2 and 4 products were more likely to churn. Therefore, limiting product advertisements/targeted offers for clients that already possess more than one product should be avoided
- **Increasing retention rates for German clients**: As for the previous two measures, this one is inspired by the predictive power of Geography in our model. After taking a look at the data, we noticed that German clients were those that churned the most
- **Cutting the German branch of the bank:** this measure might seem extreme, but German customers are the highest churners despite the fact that they only account for 25% of our customers and churn as much as the French, who account for 50% of our clientele. Cutting our activities in Germany to focus our efforts on growing a seemingly more loyal French clientele could be the key to reducing our churn rate on the long term

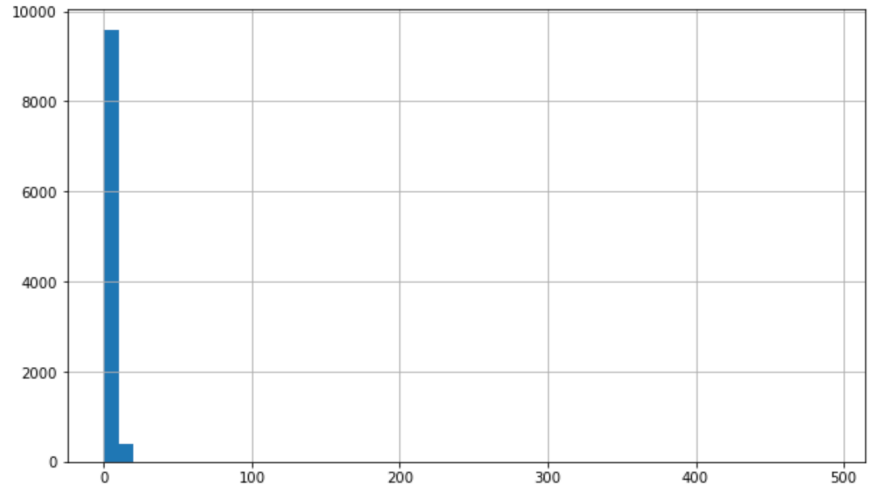
3. Appendix

Figure 1

- As we can see, out of 10,000 instances, only 7 have average day transactions exceeding 20. All these instances have abnormally high values.

```
df['Average day transactions'].hist(bins=np.arange(start=0, stop=500, step=10), figsize=[10,6])
```

```
<AxesSubplot:>
```




```
In [10]: df[df['Average day transactions']>20]
```

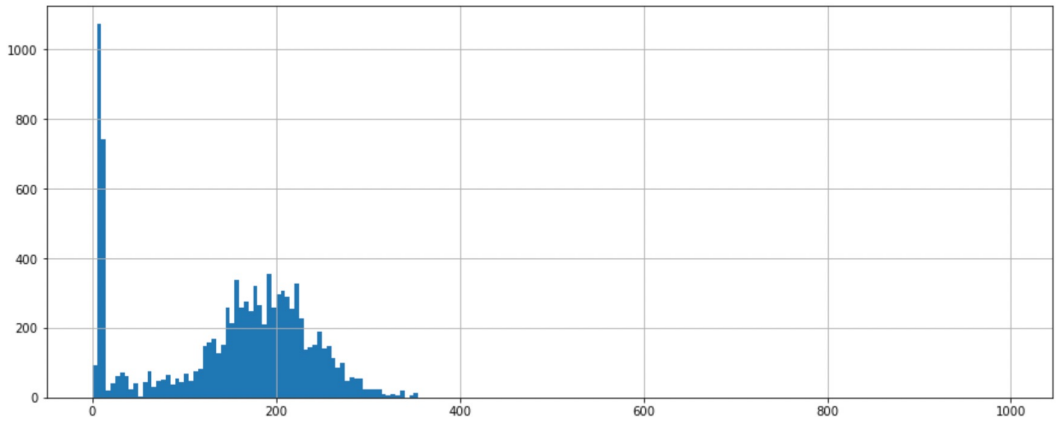
ustomer service calls	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Average day expenditure	Average day transactions	EstimatedSalary	Exited
	1	135515.760000	1	1	0	8.400000	1187	86013.960000	0
	4	113709.280000	1	1	1	282.200000	1728056	153887.930000	1
	8	130701.290000	1	1	0	220.600000	173910	59354.240000	1
	2	176139.500000	2	1	0	245.000000	10000	129654.220000	0
	9	84673.170000	2	0	1	238.500000	18970	82081.350000	0
	2	100431.840000	1	1	0	166.600000	3561	35140.490000	1
	9	134208.250000	1	1	1	199.500000	1000	13455.430000	0

Figure 2

- As we can see, out of 10,000 instances, only 2 have average day expenditures exceeding 400. All these instances have abnormally high values.

```
df['Average day expenditure'].hist(bins=np.arange(start=0, stop=1000, step=5), figsize=[15,6])
```

```
<AxesSubplot:>
```




```
In [14]: df[df['Average day expenditure']>400]
```

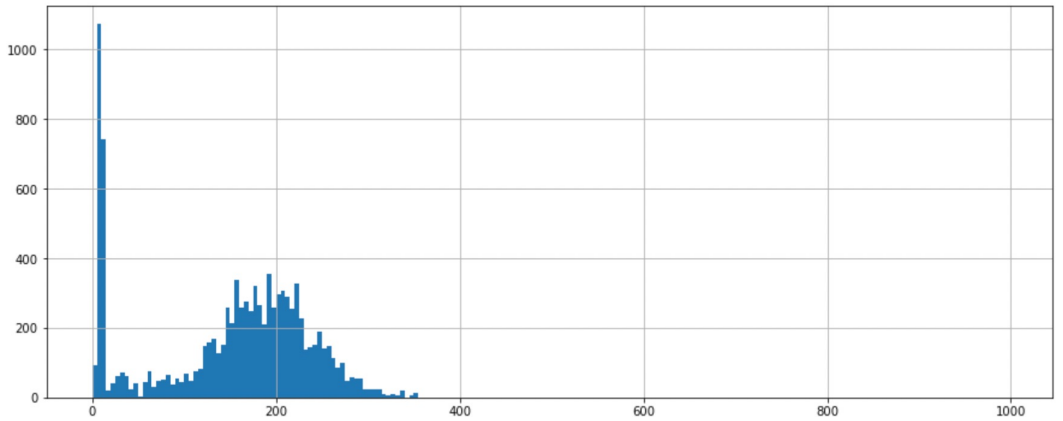
stomer service calls	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Average day expenditure	Average day transactions	EstimatedSalary	Exited
	10	0.000000	1	1	1	187620.000000	4	31299.710000	0
	10	75497.510000	1	0	0	1895620.000000	5	149682.780000	0

Figure 2

- As we can see, out of 10,000 instances, only 2 have average day expenditures exceeding 400. All these instances have abnormally high values.

```
df['Average day expenditure'].hist(bins=np.arange(start=0, stop=1000, step=5), figsize=[15,6])
```

```
<AxesSubplot:>
```




```
In [14]: df[df['Average day expenditure']>400]
```

stomer service calls	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Average day expenditure	Average day transactions	EstimatedSalary	Exited
	10	0.000000	1	1	1	187620.000000	4	31299.710000	0
	10	75497.510000	1	0	0	1895620.000000	5	149682.780000	0

Figure 3

Identifying the upper and lower limits to remove outliers:

- In order to detect the values below and above the 99th percentile of our training dataset for each attribute, we used the Pandas library on Python

```
In [15]: df.quantile(0.005)
```

```
RowNumber                      50.995000
CustomerId                 15567332.975000
CreditScore                  418.000000
Age                           19.000000
Customer service calls         0.000000
Tenure                         0.000000
Balance                        0.000000
NumOfProducts                  1.000000
HasCrCard                      0.000000
IsActiveMember                 0.000000
Average day expenditure        4.379800
Average day transactions       0.000000
EstimatedSalary              823.345500
Exited                         0.000000
Name: 0.005, dtype: float64
```

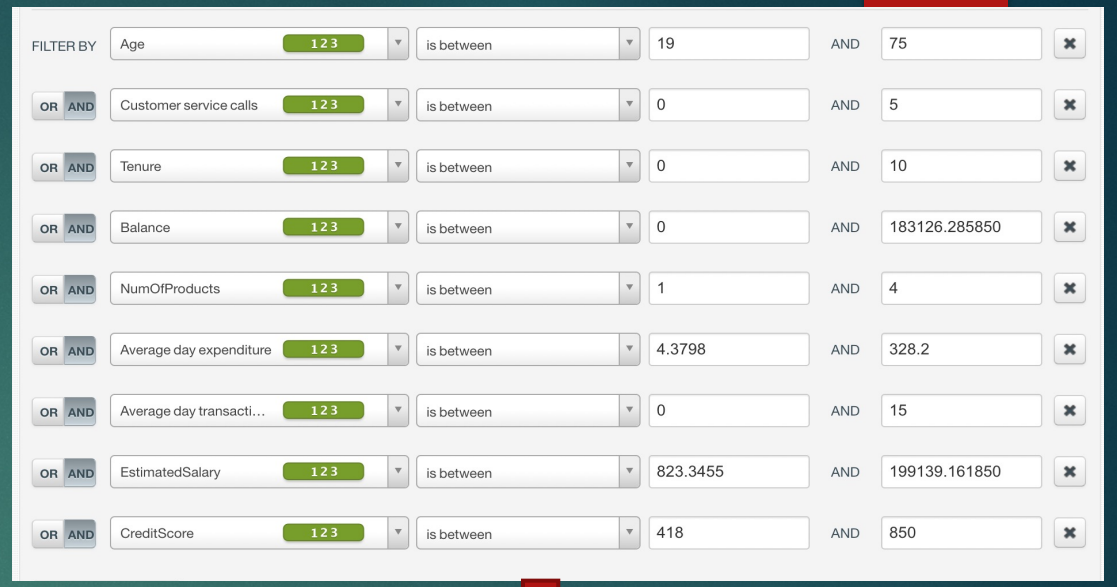
```
In [16]: df.quantile(0.995)
```

```
RowNumber                      9950.005000
CustomerId                 15814267.005000
CreditScore                    850.000000
Age                             75.000000
Customer service calls           5.000000
Tenure                          10.000000
Balance                     193126.285850
NumOfProducts                    4.000000
HasCrCard                        1.000000
IsActiveMember                   1.000000
Average day expenditure        328.200000
Average day transactions        15.000000
EstimatedSalary             199139.161850
Exited                           1.000000
Name: 0.995, dtype: float64
```

Figure 4

Eliminating the outliers from the training set:

- After identifying the values above and below the 99th percentile, we used BigML's filter feature to eliminate all values below and above the limits established for the training set. We did not touch the test set.



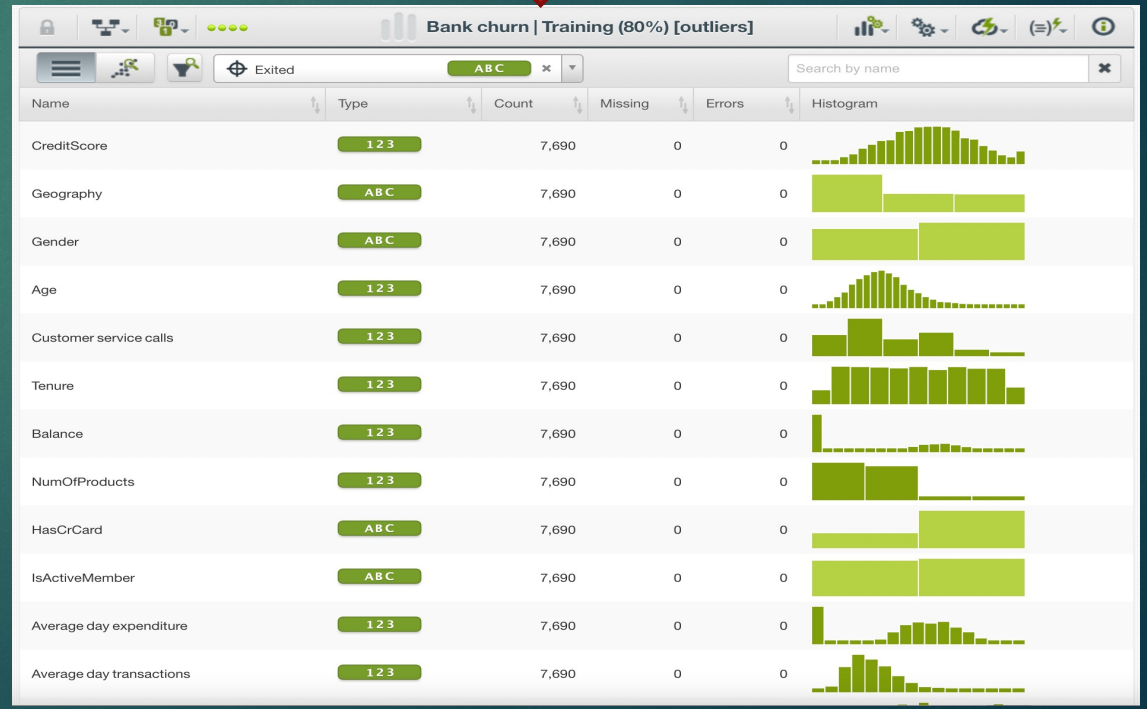


Figure 5 & 6

Figure 5:

- After filtering the outliers, we used BigML's anomaly detector to detect the 36 anomalies with the highest scores**. This procedure was carried out on the cleaned training set only (after having removed outliers)**

Figure 6

- Once the anomalies were obtained, we decided to discard all instances with an anomaly score exceeding 60% (according to rules of good practice stated by the BigML documentation)

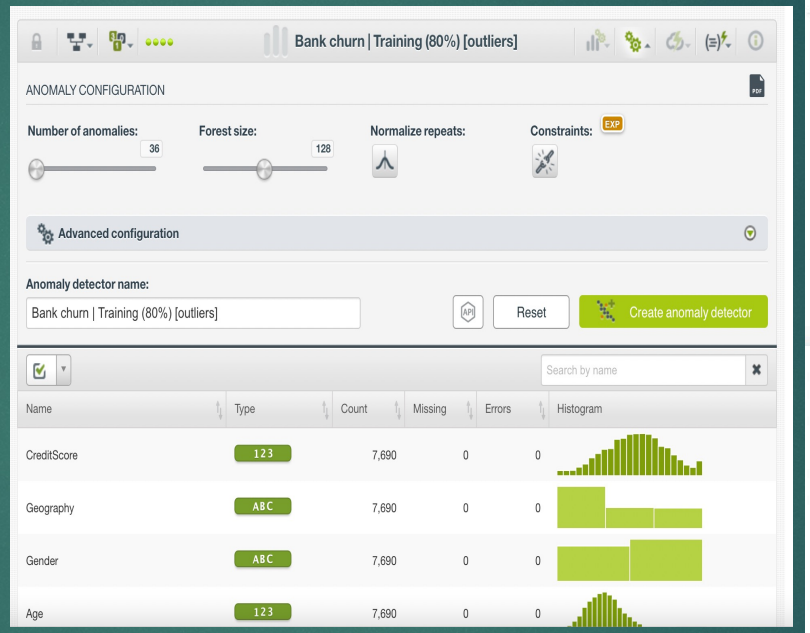


Figure 5

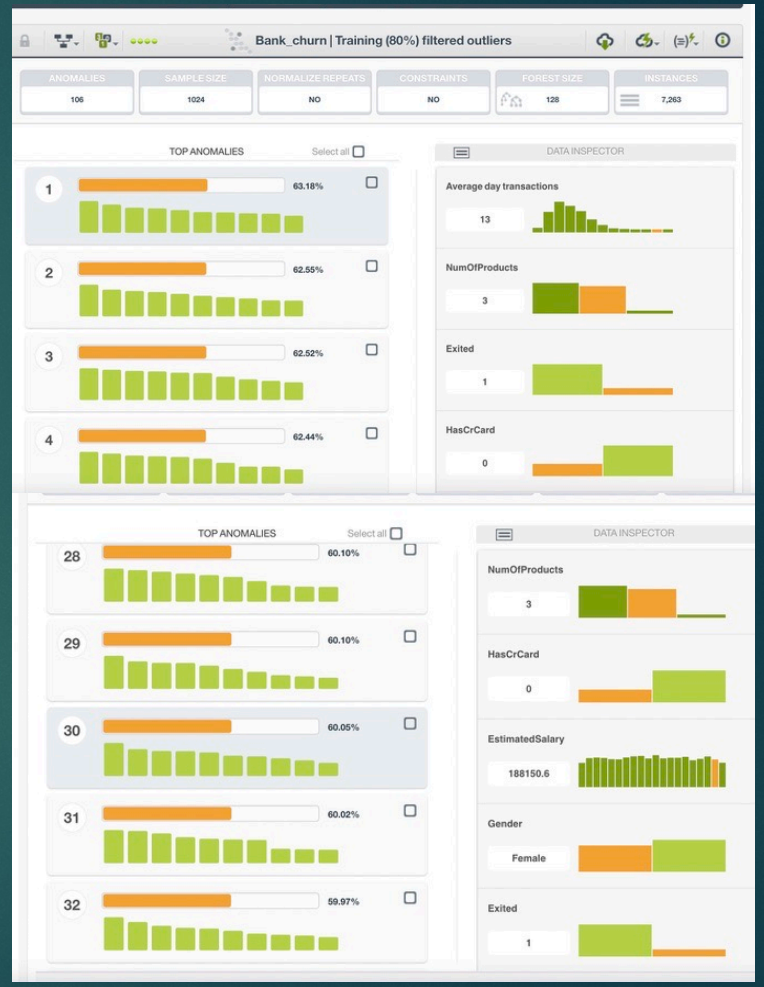


Figure 6

Figure 7

Normalizing the training and test sets:

- After cleaning the training set (outliers + anomalies), we proceeded to normalizing all numerical values between 0 and 1 using the "ADD FIELDS" feature on BigML.
- The same process was realized on the test set. **However, we did not filter outliers and anomalies from the test set**

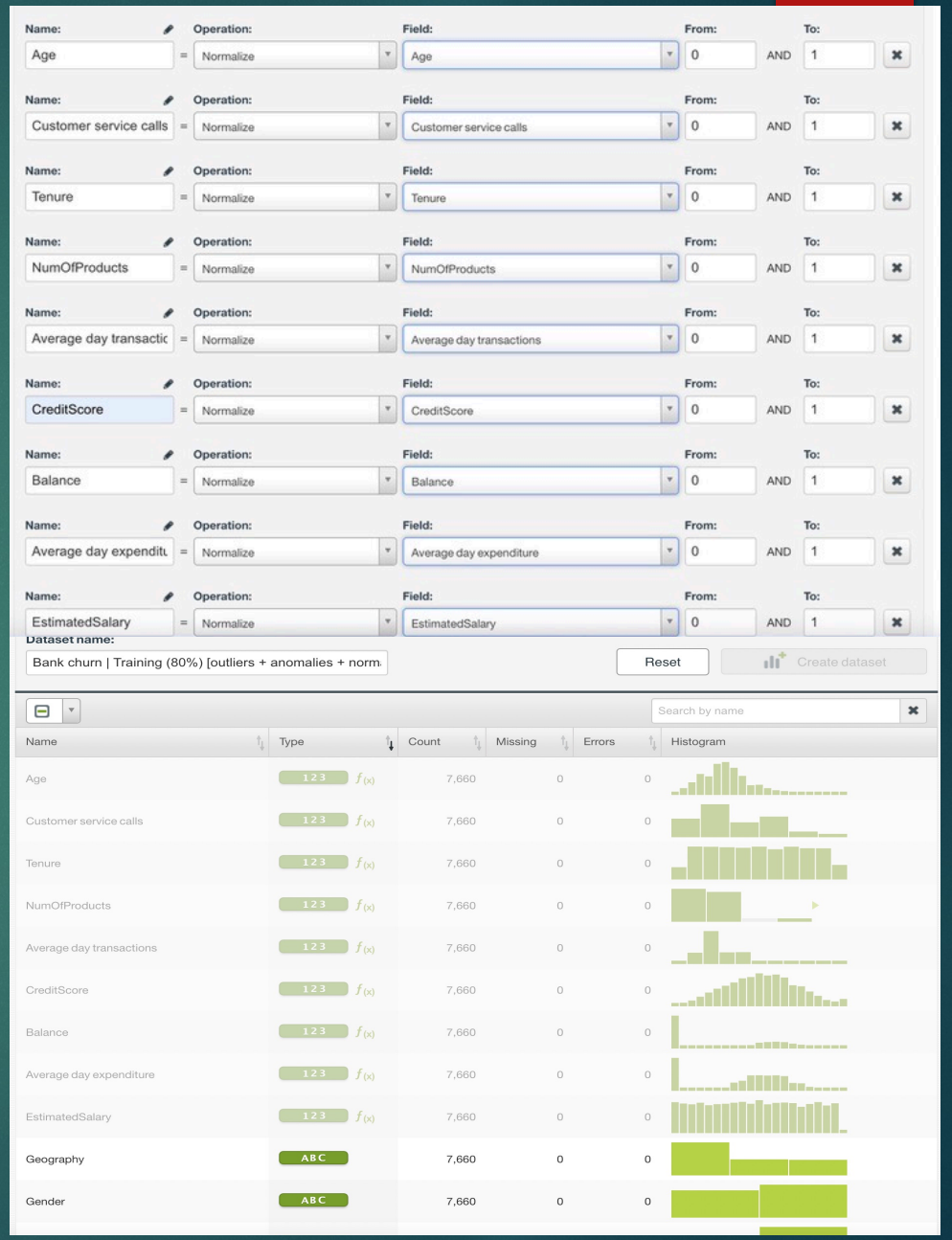


Figure 8

Filtering the unnecessary attributes during Dataset creation:

- After uploading the Bank Churn csv file as the source of the project, we eliminated the undesired attributes (RowNumber, CustomerId, and Surname) before creating the dataset

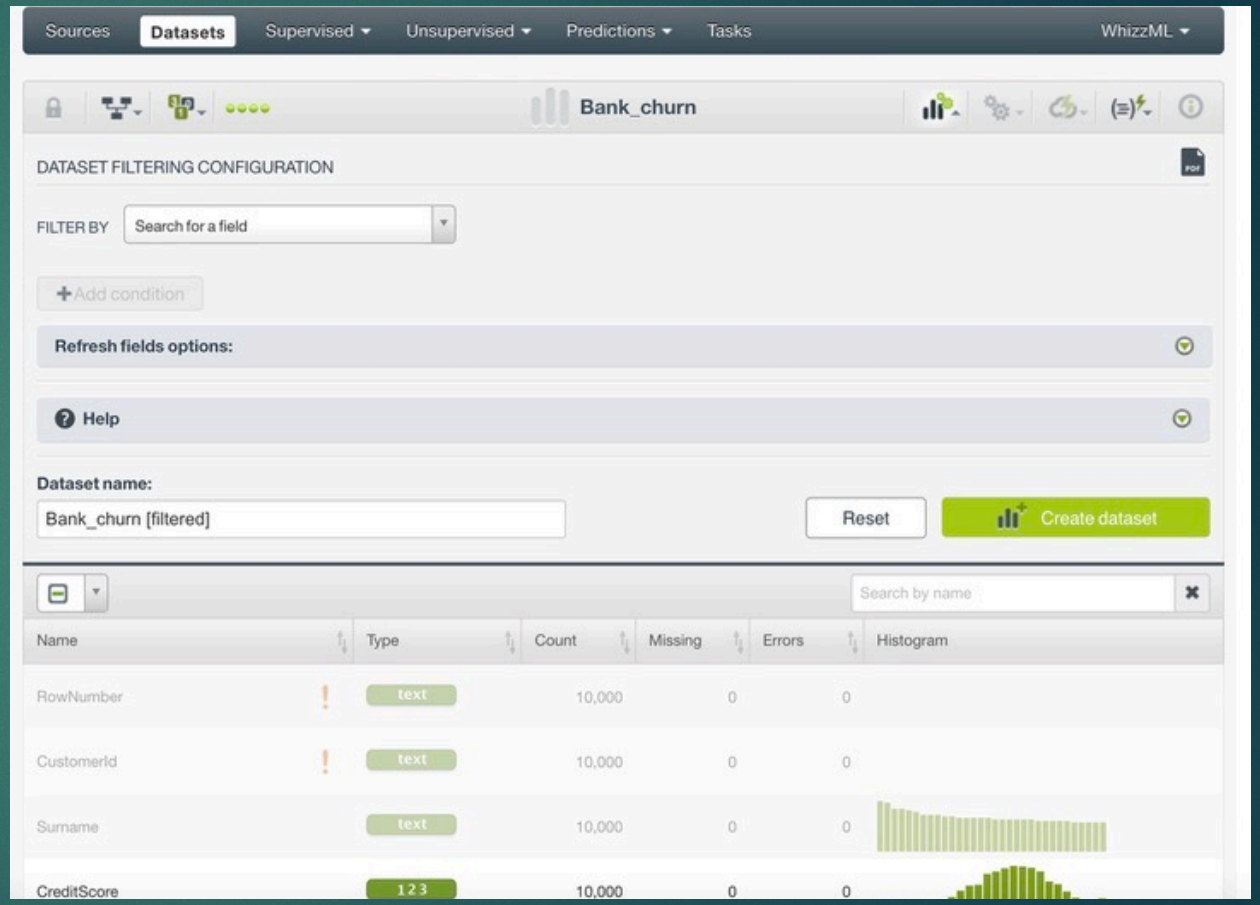


Figure 9

Creating the training/test set split:

- After removing the undesired attributes from the source to configure the dataset (see figure 8), we split it into a practice and training set.

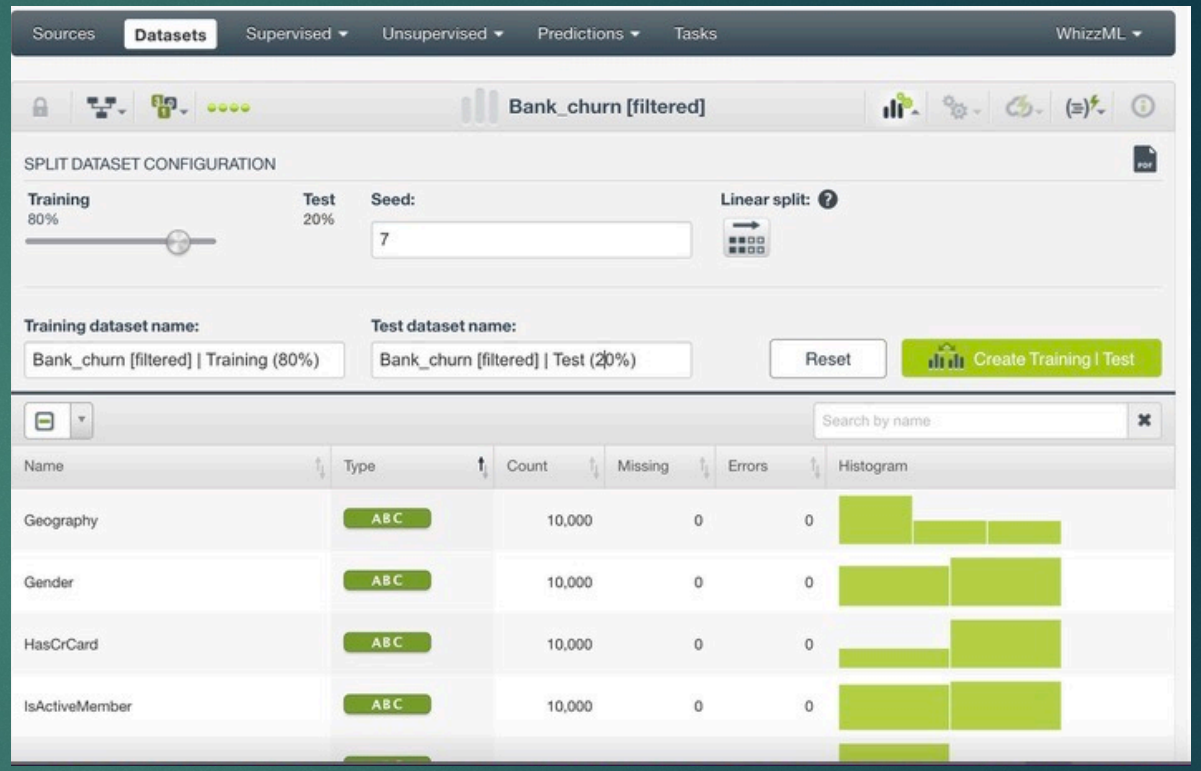


Figure 10

Creating the simple model:

- Training set used: Filtered (outliers then anomalies + Normalized numerical attributes

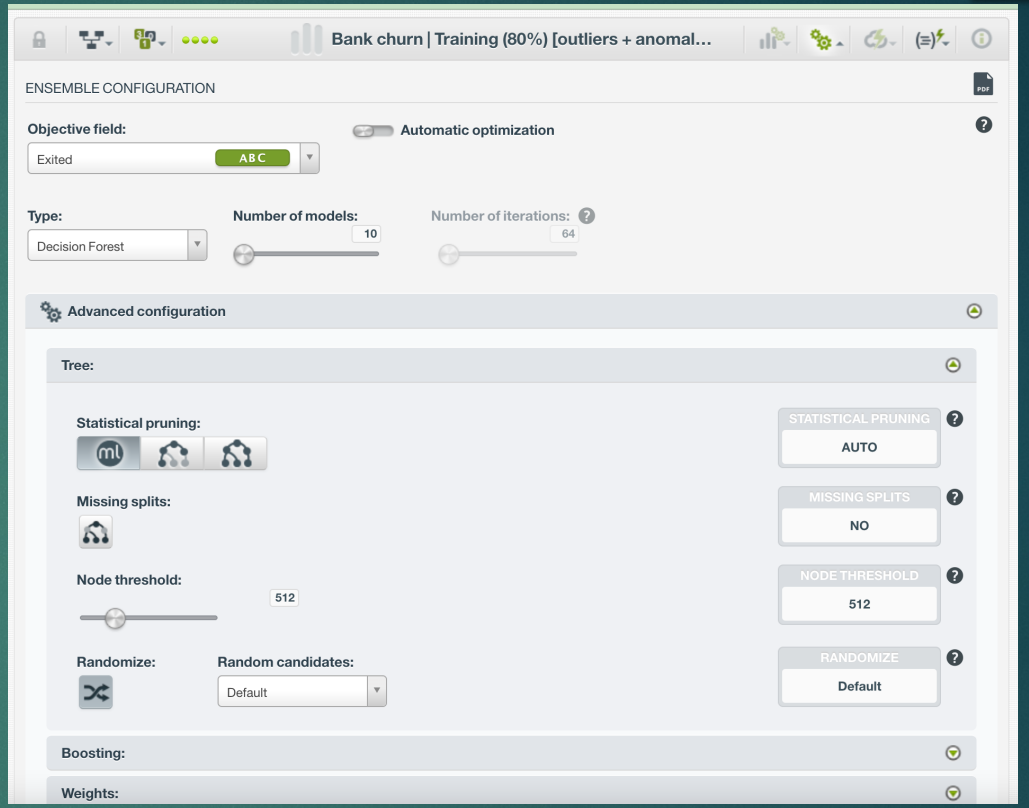


Figure 11 & 12

Figure 11: Ensemble randomized tree model summary

Figure 12: Model performance evaluation using test set

- Test set used: other 20% of original Dataset Normalized

Figure 11

Figure 12

Figure 13

Feature engineering 1:

- Using the "ADD FIELDS" feature, we used a Lisp flatline formula to create "Average day expenditure per transaction", which was equal to the quotient of Average day transactions and Average day expenditure
- We created this variable on the training and test set separately. However, we used the non-normalized versions of these datasets
- In the case of the training set, the test set we used was filtered for anomalies and outliers
- After creating the new variable, we removed the old variables to avoid issues related to collinearity

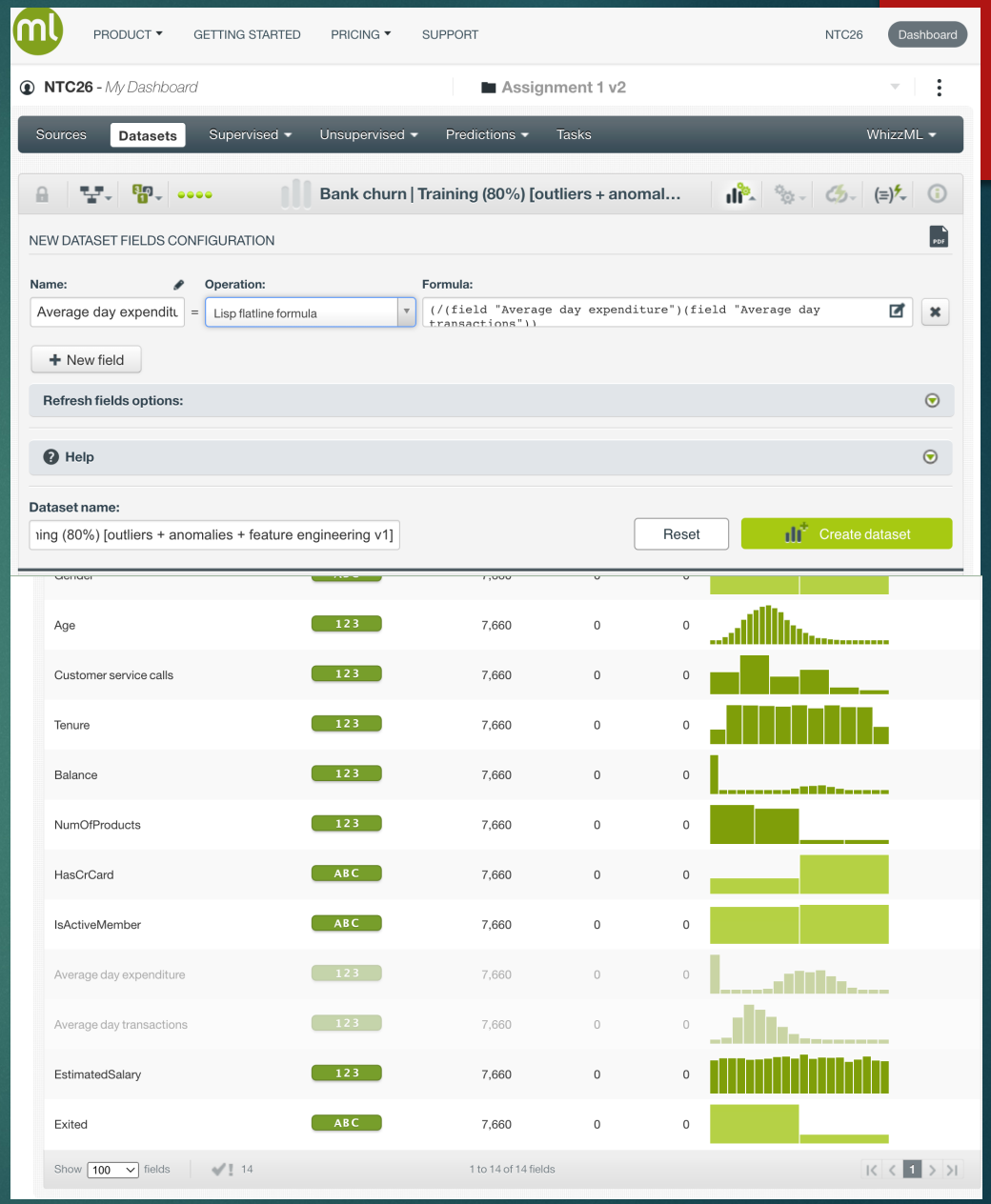


Figure 13

Figure 14 & 15

Feature engineering 1:

- Once the new variable was created, we noticed missing values in both the training and test. This was because some instances had Average day transactions worth 0, which made the ratio impossible to calculate (figure 14)
- Since we did not want to lose data and certain algorithms cannot deal with missing values, we used the "ADD FIELDS" feature to replace the missing values with the median of instances for this variable (figure 15)
- Once this was done, we proceeded to then normalizing the resulting training and test sets

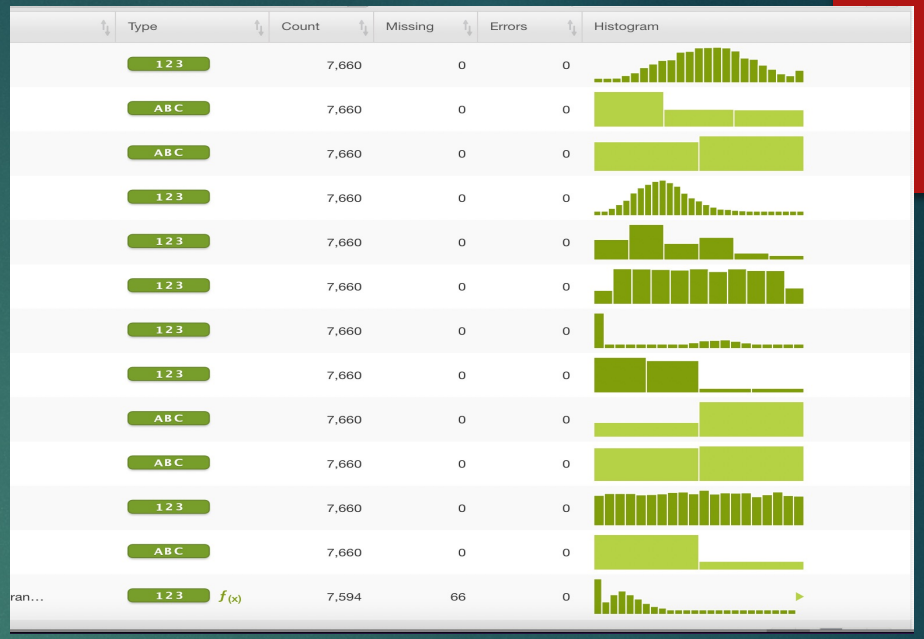


Figure 14

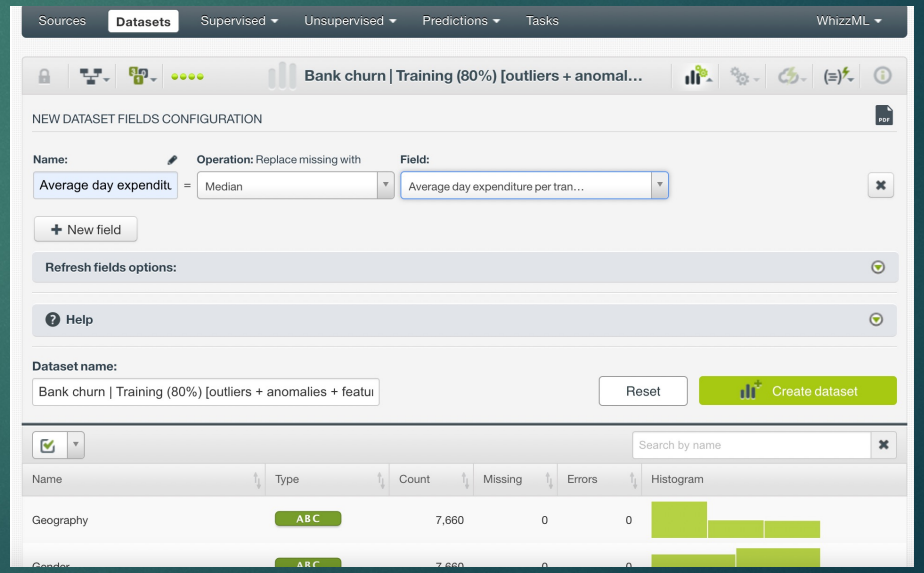


Figure 15

Figure 16 & 17

Feature engineering 1:

- This model was created using the same model creation parameters as the first simple model we created (see figure 10)

Figure 16

Figure 17

Figure 18 & 19

Feature engineering 2:

- We observed the correlation between training set variables using BigML's Linear regression feature. This is how we identified a significant correlation between NumOfProd and Balance (see figure 18)
- In order to combine these variables, we decided to create BalancexNOP, which is equal to the multiplication of Balance by NumOfProd (see figure 19)
- Since the correlation was spotted between the normalized features in the training set, we carried out the operation in figure 19 on the normalized training set and test set. Once again, in both sets, we deleted the original variables to avoid issues related to collinearity

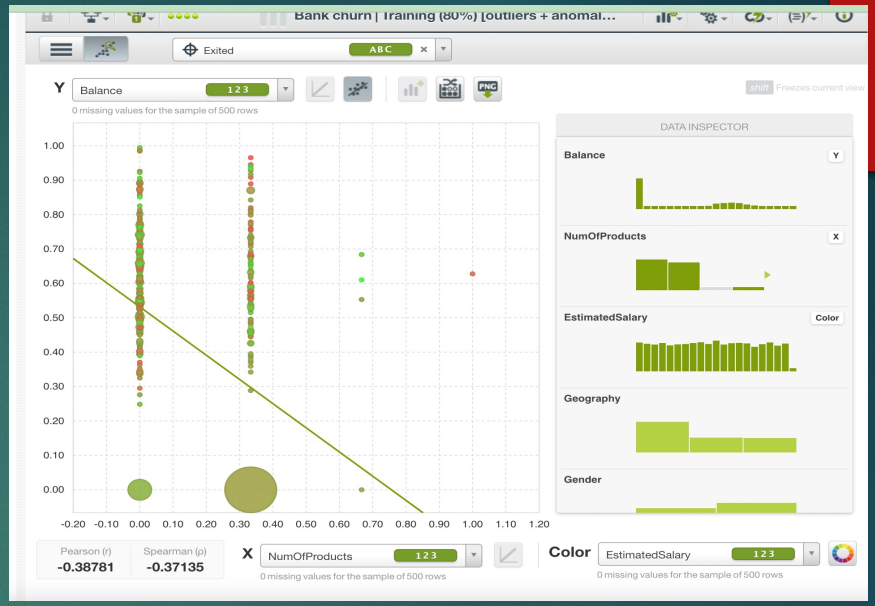


Figure 18

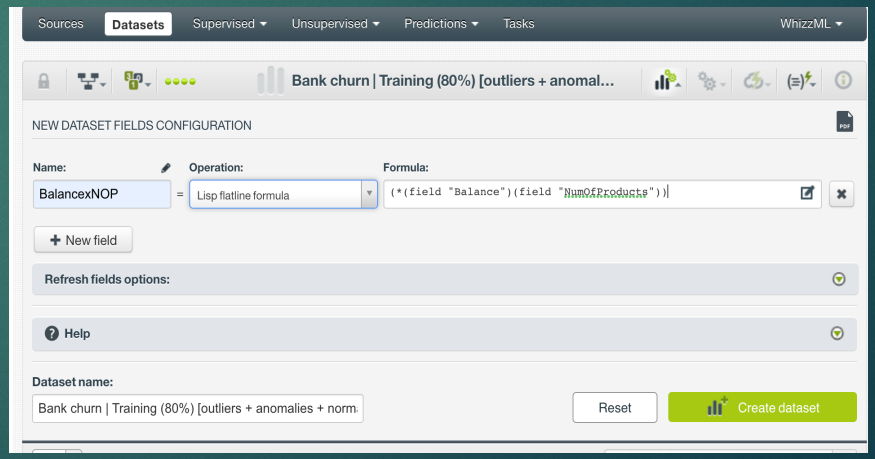


Figure 19

Figure 20 & 21

Feature engineering 2:

- This model was created using the same model parameters as the first simple model we created one (see figure 10)

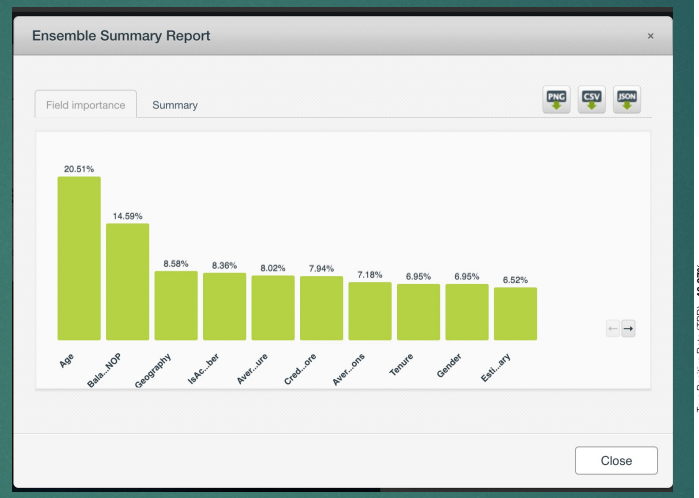


Figure 20

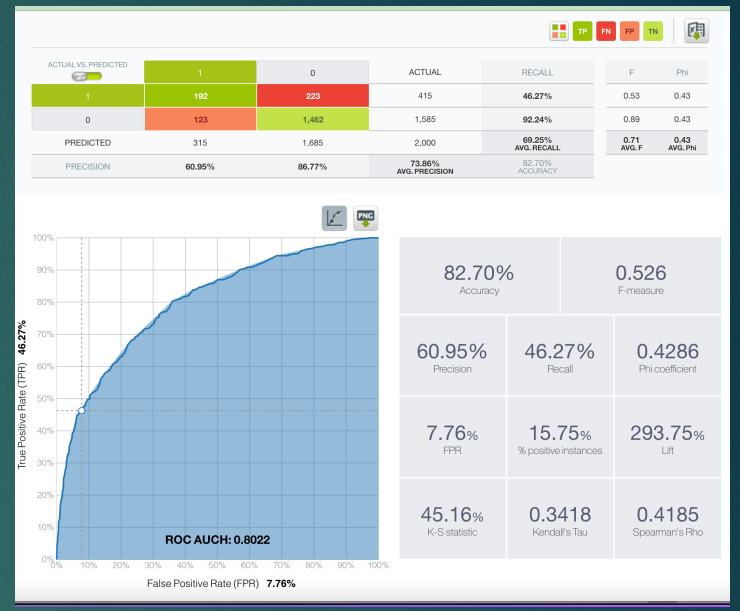


Figure 21

Figure 22 & 23

Feature removal:

- This model was created following the same creation parameters (see figure 10) and on the same training set as the first simple model we made. The only difference is that we deselected HasCrCard and Customer service calls
- In term of performance, this model has a higher accuracy and recall than the first model we made, but a slightly lower Precision. The ROC AUCH is also higher than our first trained model (see figure 12)
- In view of this increase in overall performance, we have decided to permanently remove the two features mentioned above

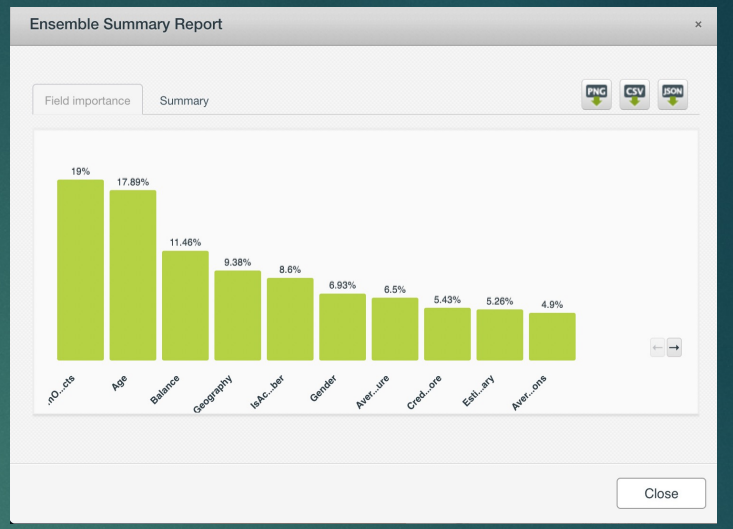


Figure 22

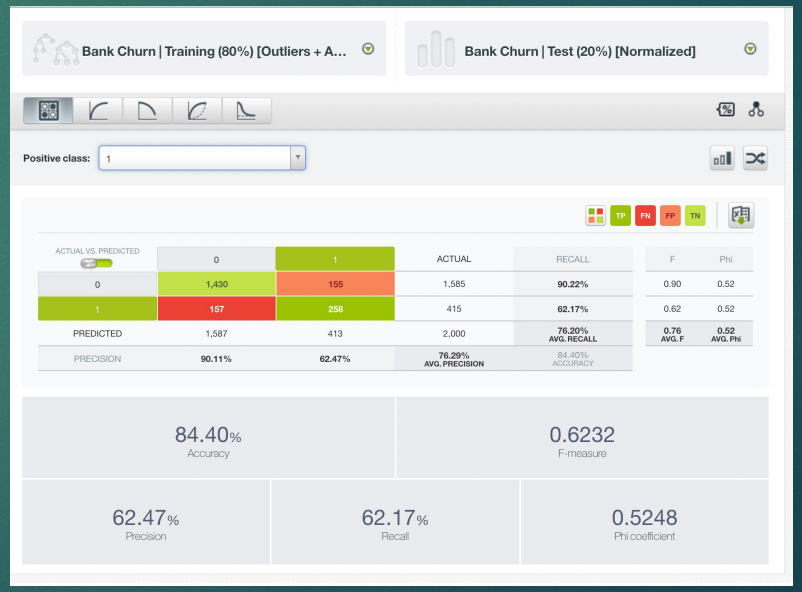


Figure 23

Figure 24

Deepnet model creation

- To create this model, we used the same dataset as the dataset used to create the model shown in figure 22 & 23

- When it comes to the configuration of the deepnet model, we opted to use the "Automatic optimization" feature offered by BigML

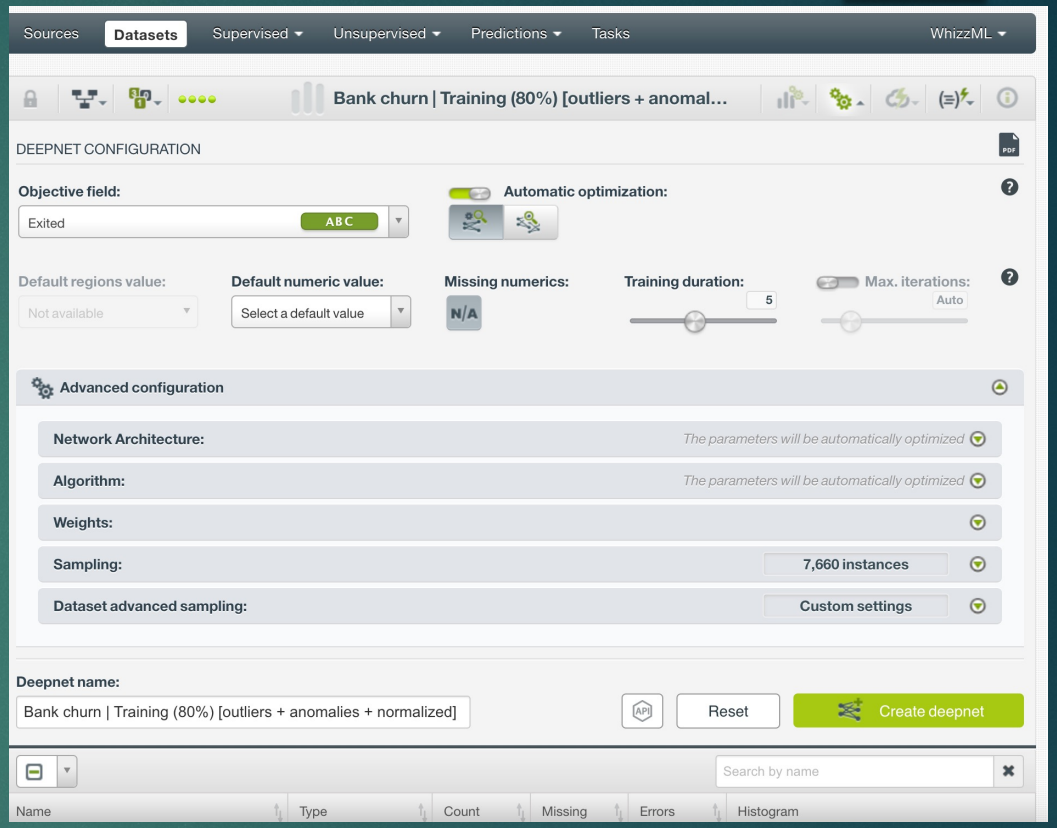


Figure 24

Figure 25

Deepnet model performance

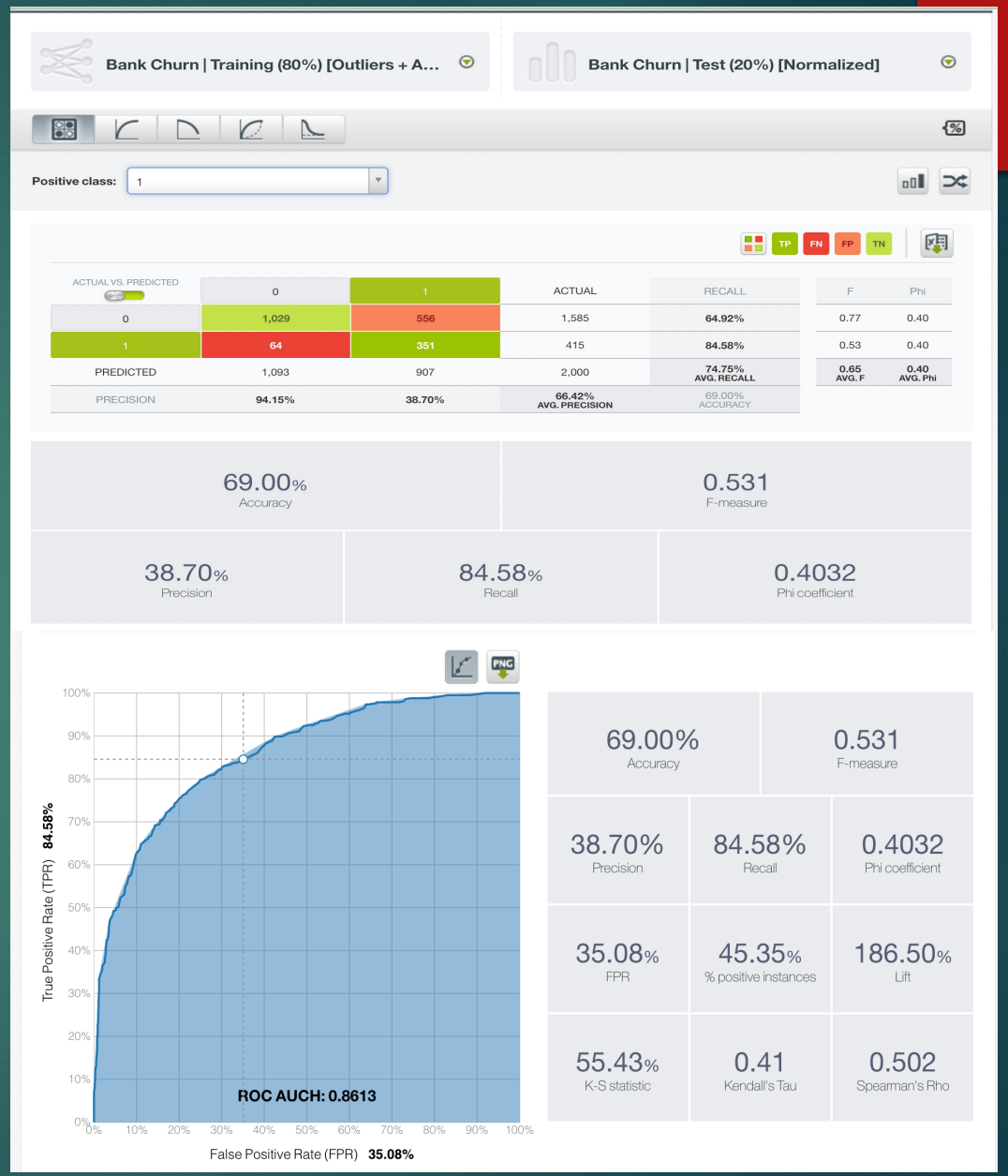


Figure 25

Figure 26

Deepnet k-fold cross-validation

- In order to carry out the cross validation (using the SCRIPT feature on BigML), we used the total dataset (cleaned, unnormalized) to which we removed the variables HasCrCard and have elected to go with 5 k-folds as it is standard practice in machine learning
- We did not make any further modifications to the dataset as this would contribute towards target leakage (as each fold of the dataset will serve as training and test set)

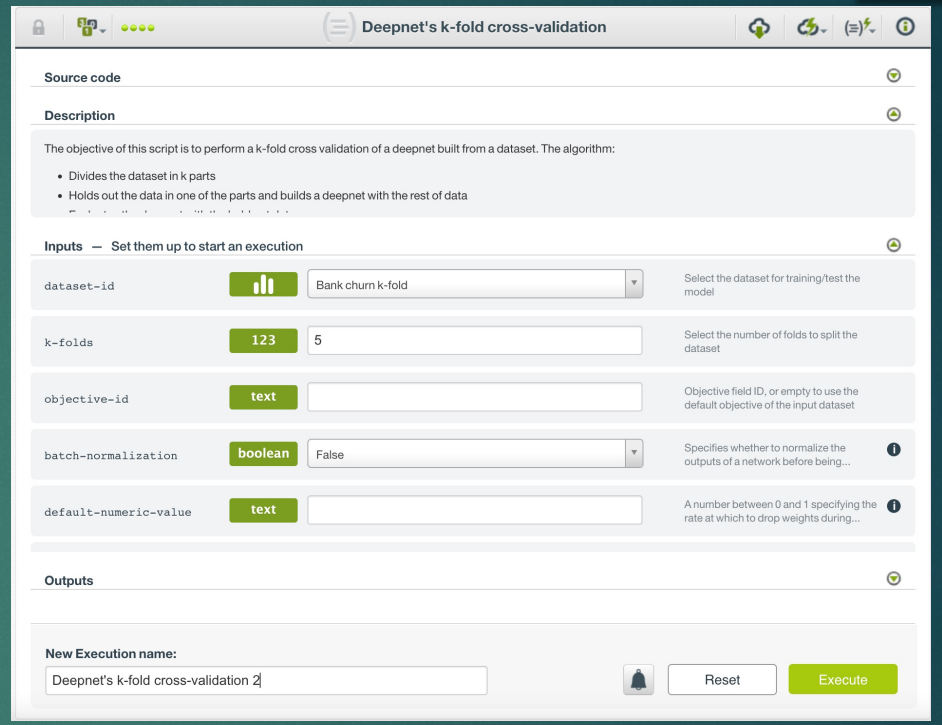


Figure 27

Figure 27

Deepnet k-fold model performance

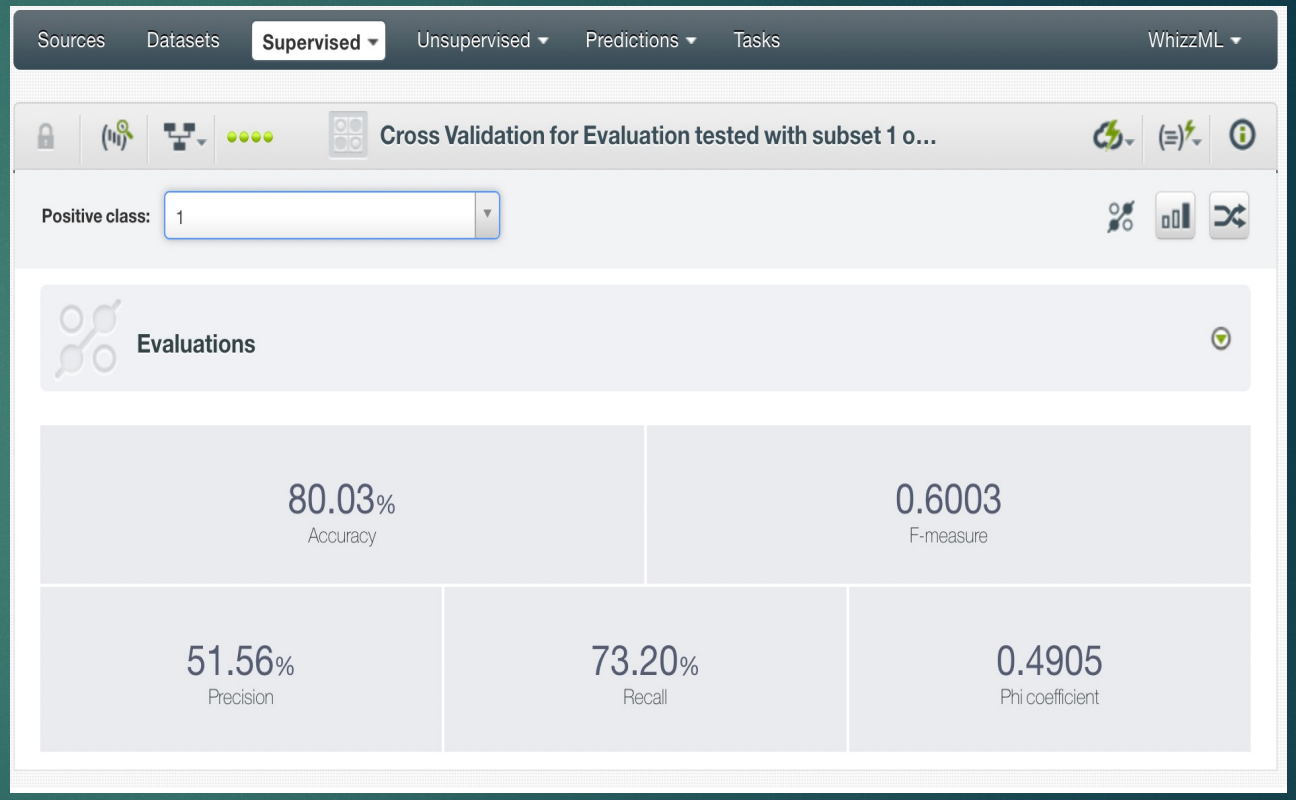


Figure 28

Randomized Decision Forest k-fold cross-validation

- In order to carry out the cross validation (using the SCRIPT feature on BigML), we used the total dataset (cleaned, unnormalized) to which we removed the variables HasCrCard and have elected to go with 5 k-folds as it is standard practice in machine learning

- We did not make any further modifications to the dataset as this would contribute towards target leakage (as each fold of the dataset will serve as training and test set)

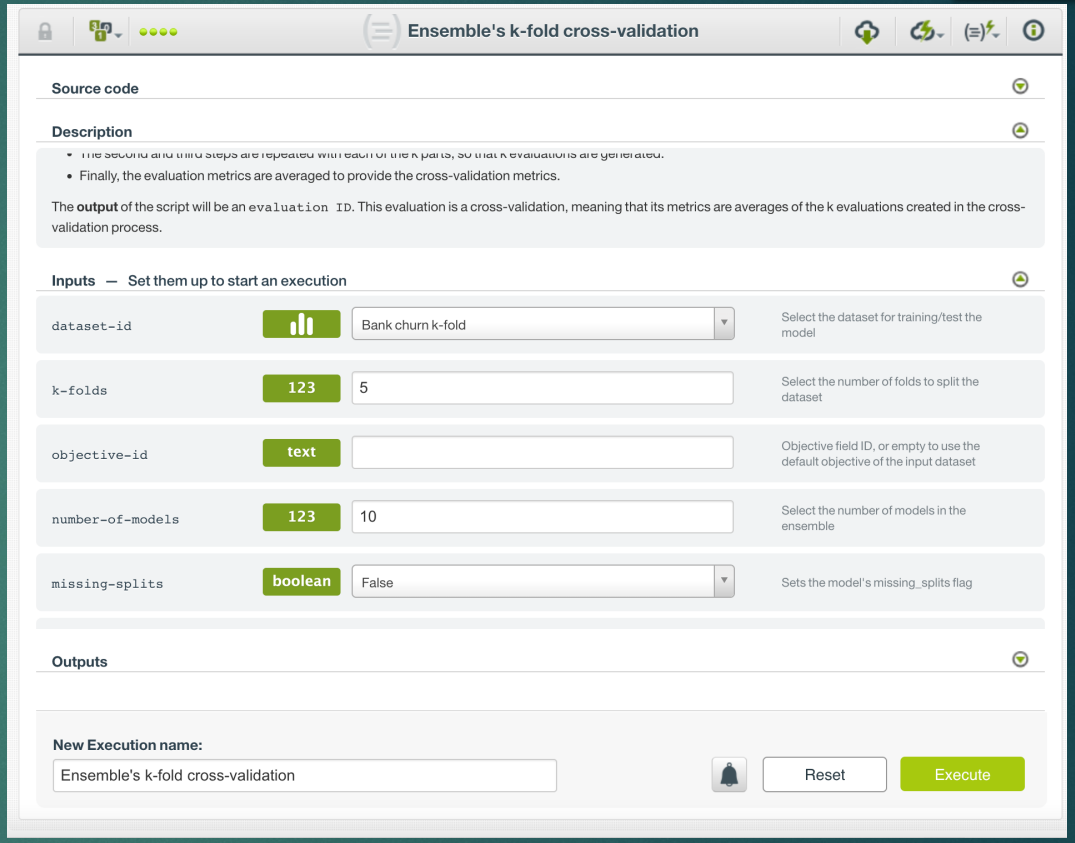


Figure 29

Figure 29

Ensemble k-fold model performance

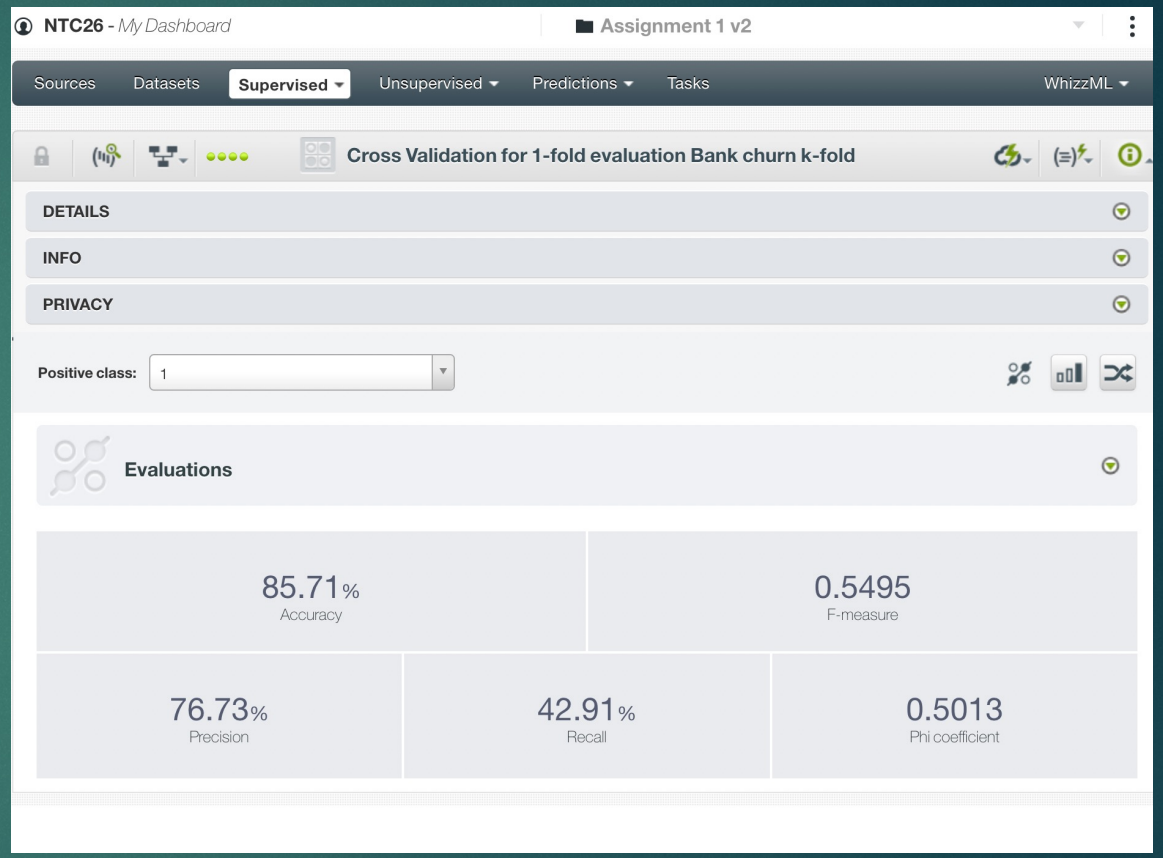


Figure 31

Fine-tuning the Ensemble model: :

- The dataset used to train this model is the same dataset as the one used to create our initial ensemble model (80% training set, to which we removed outliers and anomalies and normalized numerical values)
- The reason for which we did not use the full 100% of the dataset is because we will be keeping the test set to verify whether this model is overfitted, since this model will be trained an evaluated on the training set through Cross-validation

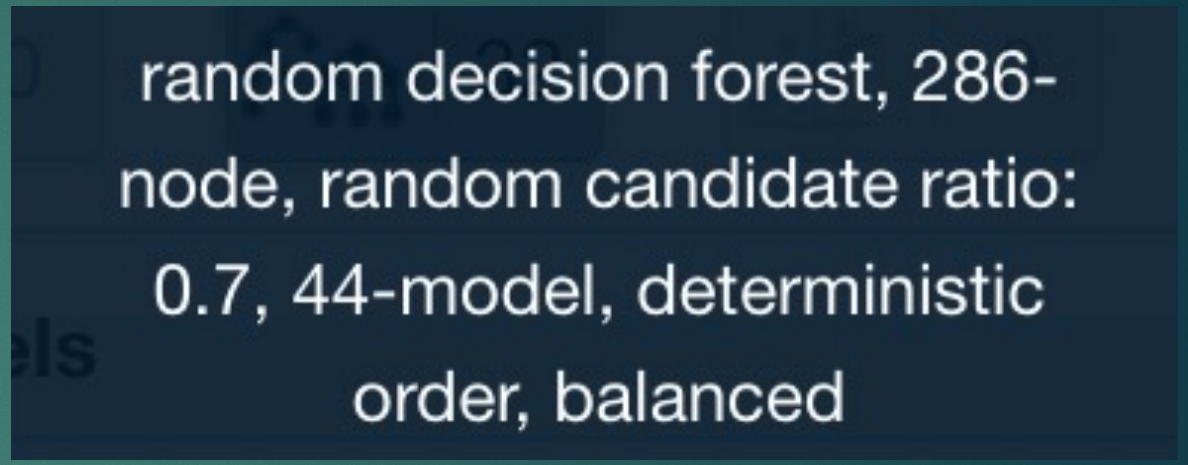


Figure 31

Figure 32 + 33

Fine-tuning the Ensemble model: performance metrics + feature analysis

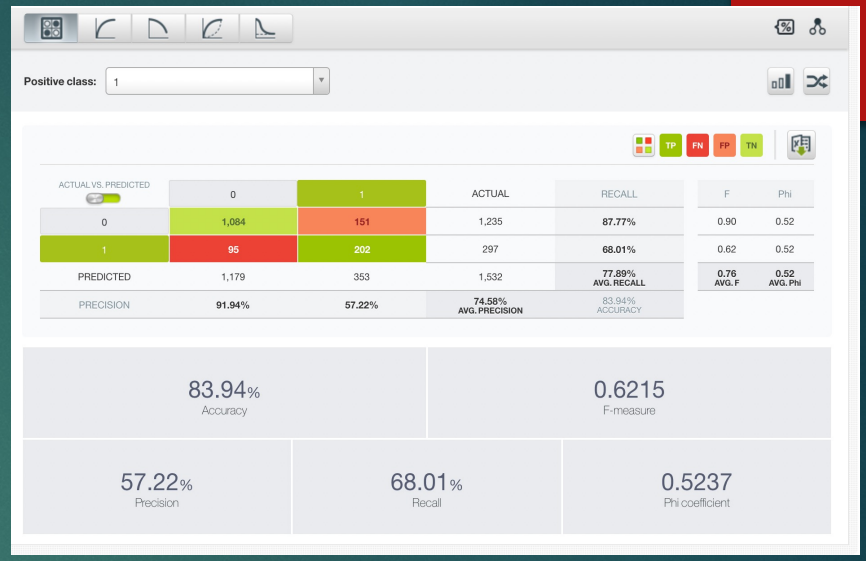


Figure 32

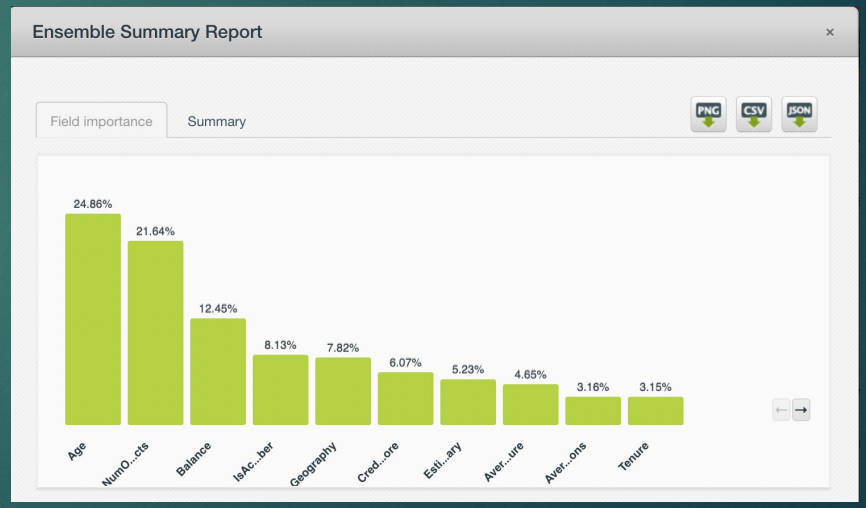


Figure 33

Figure 34

Fine-tuning the Ensemble model: overfitting check

- In order to check whether our model was overfitting, we decided to evaluate it on unseen data

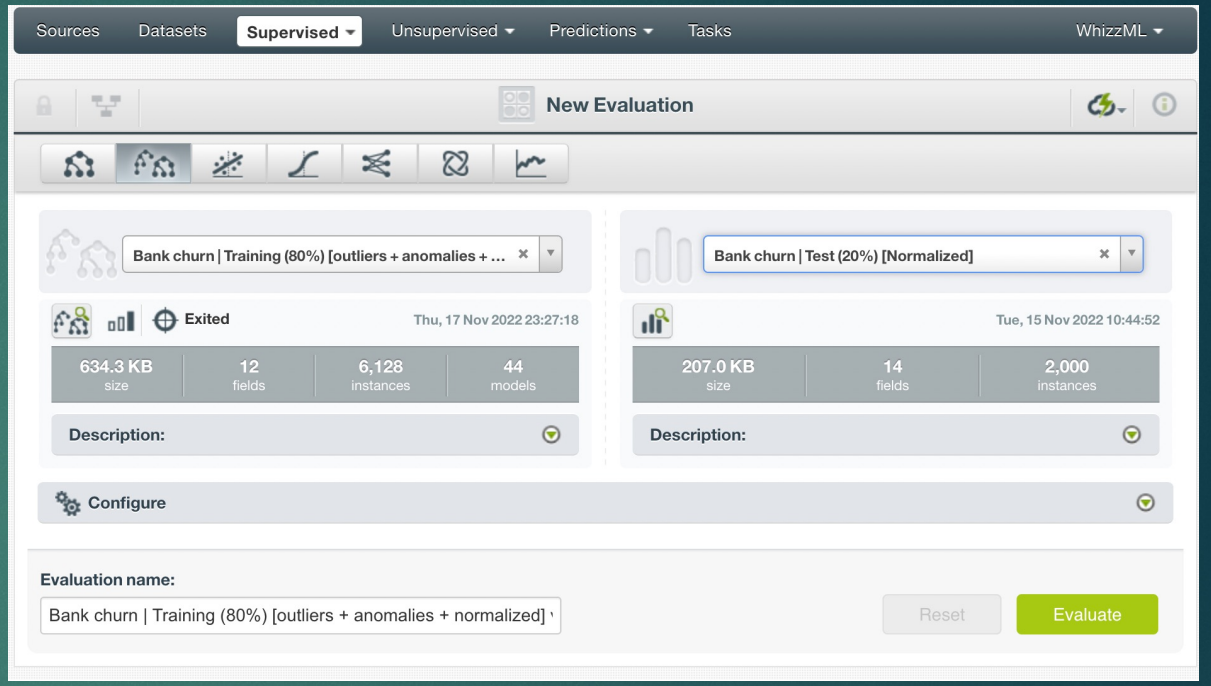


Figure 34

Figure 35

Fine-tuning the Ensemble model: overfitting check

- In order to check whether our model was overfitting, we decided to evaluate it on unseen data. In order to do this, we evaluated the model a second time using the test set
- As we can see, the performance of the model has not dipped significantly, further confirming that our model does not suffer from overfitting
- We decided to carry out our calculations using this version of the model, as this is a more accurate predictor of the model's performance on a new dataset

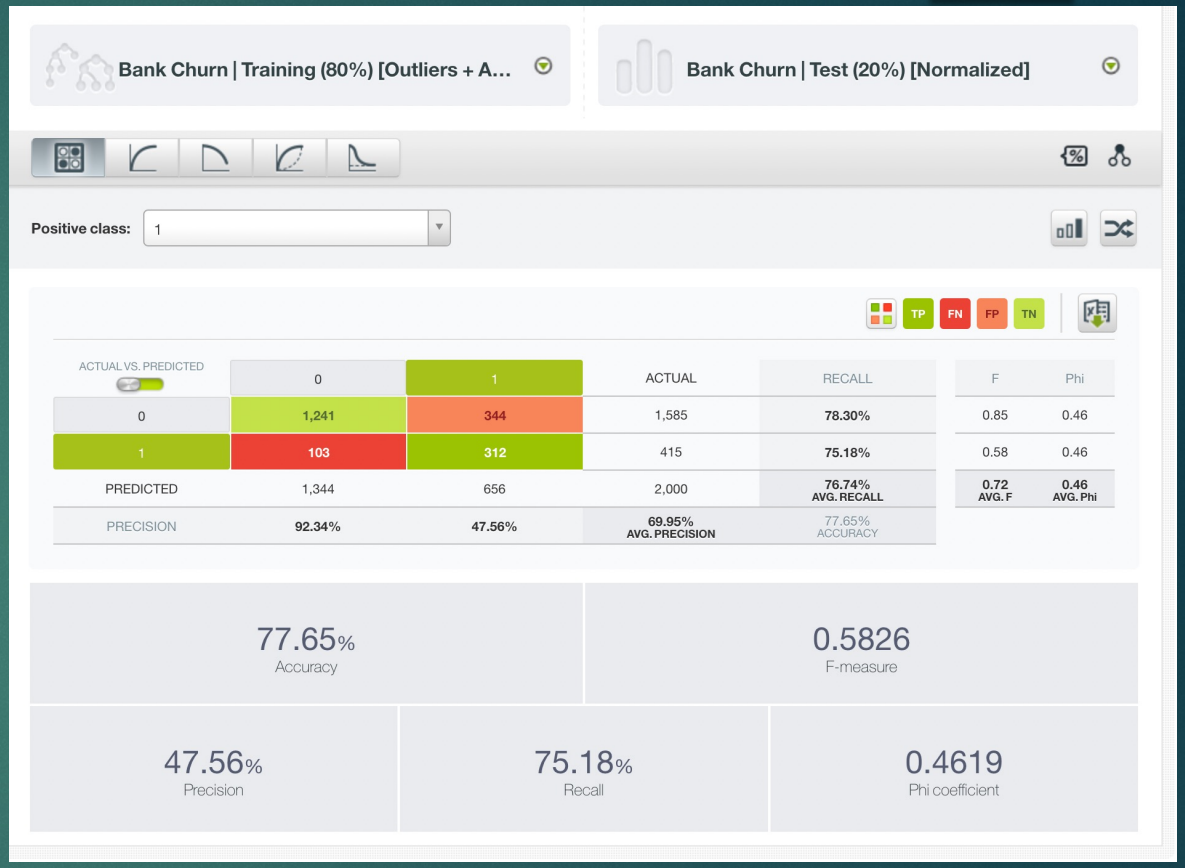


Figure 35

Figure 36

Fine-tuning the Deepnet model: hyperparameters selection

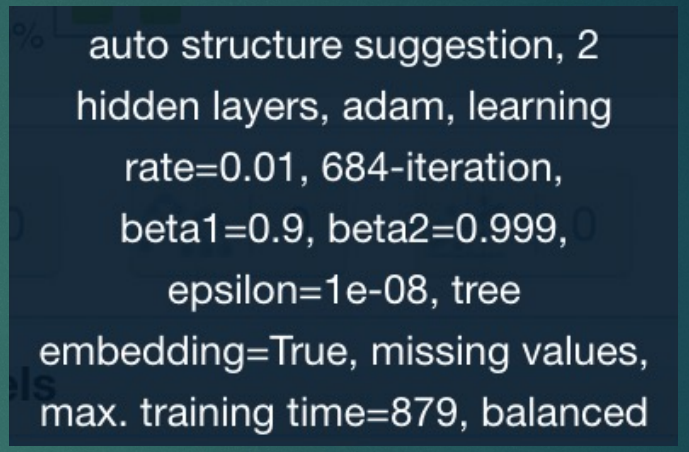


Figure 36

Figure 37 + 38

Fine-tuning the Deepnet model: model performance metrics + feature analysis

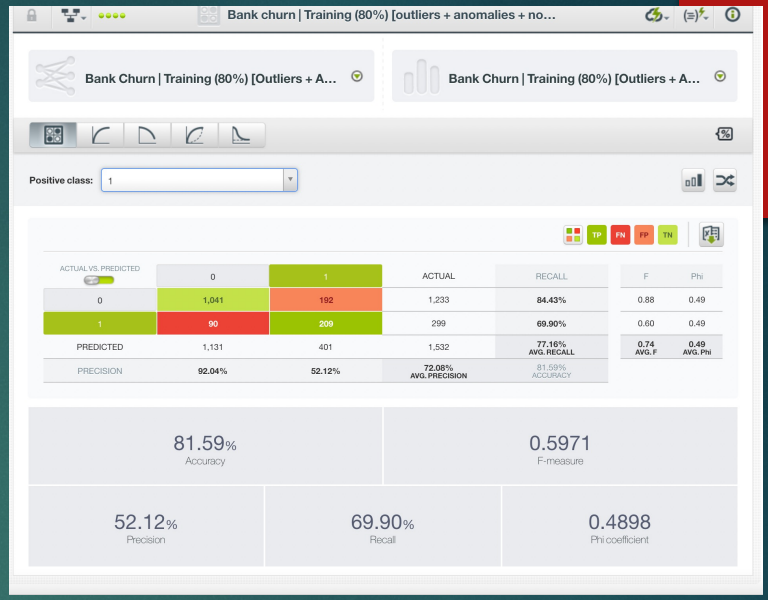


Figure 37

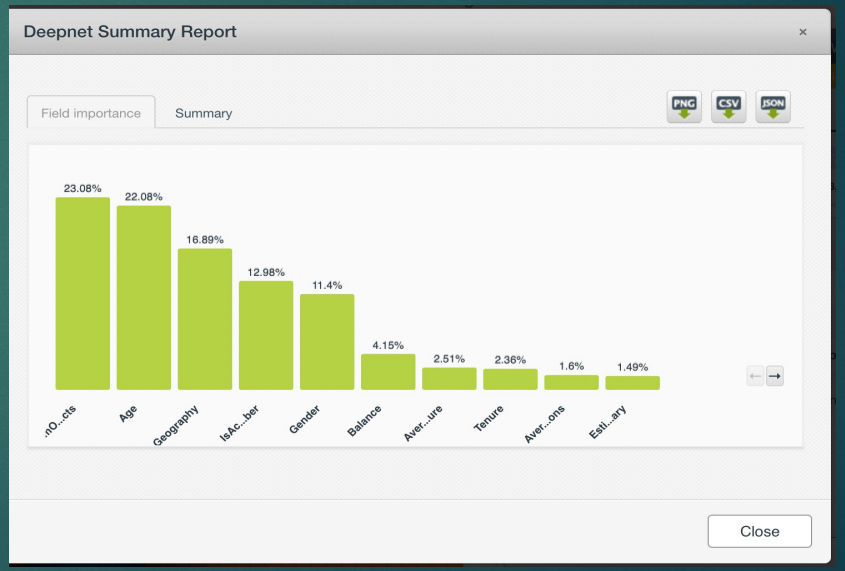


Figure 38

Figure 39

Fine-tuning the Deepnet model : overfitting check

- In order to check whether our model was overfitting, we decided to evaluate it on unseen data. In order to do this, we evaluated the model a second time using the test set
- As we can see, the performance of the model has not dipped significantly (see figure 37 for comparison), further confirming that our model does not suffer from overfitting
- We decided to carry out our calculations using this version of the model, as this is a more accurate predictor of the model's performance on a new dataset

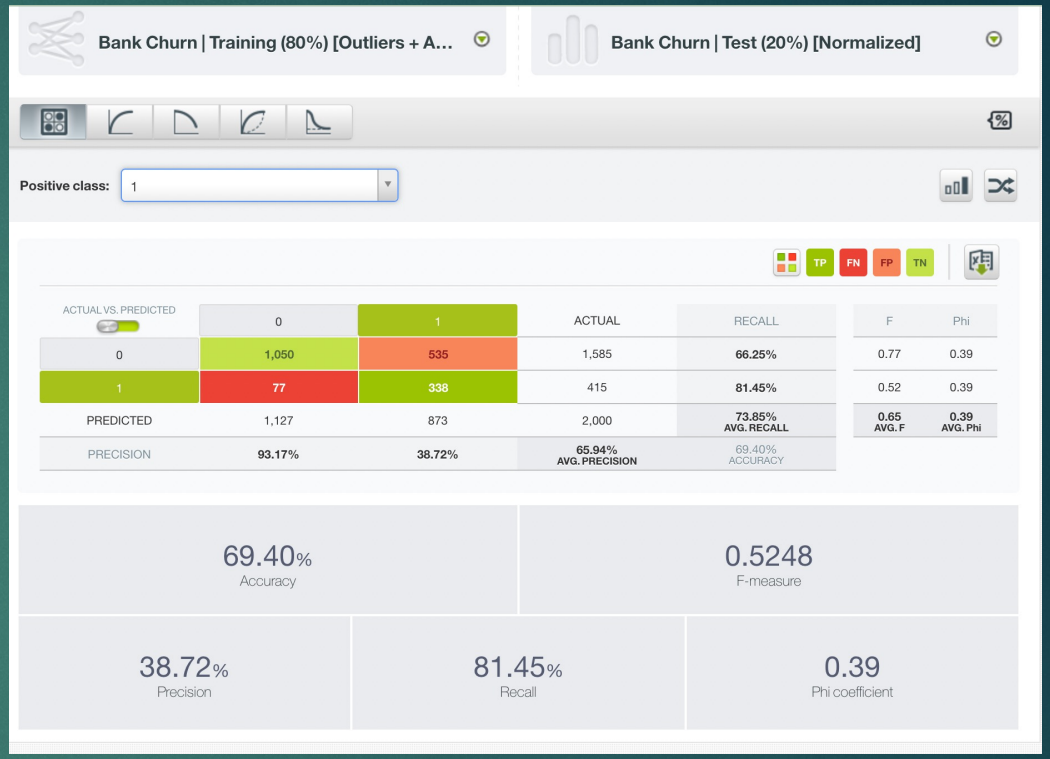


Figure 39