STATISTICAL AND MACHINE LEARNING APPROCHES FOR MARKETING

Customer Churn Prediction - Orange

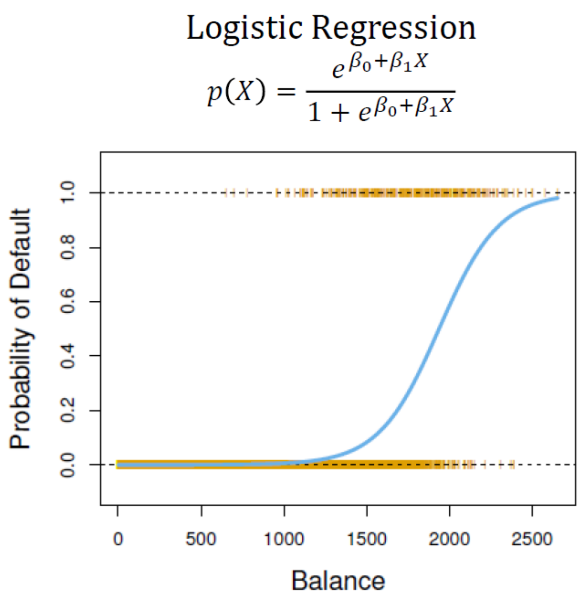
Alexandra-Maria Ionascu
April 2021

Table of contents

1. ML models

Logistic regression

This method is used in problems with binary output, where the model returns the probability that a variable Y belongs to a category (1 or 0). Logistic regression fits a logistic function of shape "S", which can be seen in the graphic bellow:



Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). An Introduction to StatisticalLearning: WithApplications in R. Springer PublishingCompany, Incorporated.

Objective function:

The objective function of the logistic regression is called "Maximum likelihood". The function estimates the function parameters by calculating the product of the individual likelihoods of each observation in the data.

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

In addition, statisticians infer that calculating the log likelihood is more stable. Thus, this method implies calculating the log value for each of the individual likelihoods and calculating their product. To maximize this log likelihood, the Gradient Descent optimization algorithm is used, which minimizes the loss function.

$$\ell(\beta) = \log \mathrm{L}(\beta) = \sum_{i=1}^{N} \{y_i \log p(x_i;\beta) + (1 - y_i) \log(1 - p(x_i;\beta))\}$$

$$= \sum_{i=1}^{N} \{y_i \beta^T x_i - \log(1 + e^{\beta^T x_i})\}$$

Pros:

- easy to interpret
- is time efficient to train
- it can be extended to multiple classes
- performs well for simple datasets
- model coefficients can be interpreted as feature importance

Cons:

- it can lead to overfitting if the number of features is bigger than the number of records
- if the assumption of linearity is wrong, logistic regression cannot be a solution
- it can work only on a discrete number set
- it requires no multicollinearity between independent variables

KNN (k-nearest neighbors)

This classification algorithm tries to find the most similar points in proximity, assuming that similar points are close to one another. For this, the algorithm calculates the distance from the observed point to the k number of neighbors (records) using multiple methods (a popular method would be Euclidean distance).

The Euclidean distance between two points p and q can be computed using the Cartesian coordinates of each point (p_1, p_2; q_1, q_2) and applying the following formula:

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

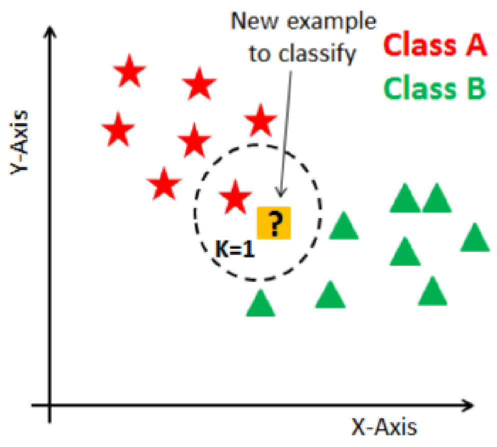
Source : https://ai.plainenglish.io/the-math-behind-knn-7883aa8e314c

After having the distances computed, the model chooses the class of the points with the most frequent class. For this, the following formula is used, where x represents the input and the highest probability to belonging to a specific class is assigned:

$$P(y=j|X=x)=\frac{1}{K}\sum_{i\in\mathcal{A}} I(y^{(i)}=j)$$

Source : https://ai.plainenglish.io/the-math-behind-knn-7883aa8e314c

Here, K is the number of neighbors passed as an input to the model. This number can be chosen through Grid search in order to optimize it, as the number of neighbors can differ depending on the dataset.



Source: https://www.datacamp.com

Pros:

- Easy to implement
- No need for tunning multiple parameters
- Can be used also for regression
- Can be used for recommendation systems

Cons:

- As the dataset gets larger, the algorithm gets slower

Decision trees

The algorithm can be used for both classification and regression problems, but in the case of the churn dataset, I used the classification decision tree. The decision tree algorithm starts by looking at the root node (contains the entire population) and it splits the data based on multiple conditions (branches/edges), creating decision nodes (contain sub splits of the data). When the algorithm reaches the final node that cannot split, the node is called a terminal note or a leaf. For classification, the end leaves of the tree represent the target distinct values (ex: churn or not churn). The decision tree algorithm is also called the greedy algorithm, as the split is chosen through a cost function, where intention is to choose the lowest cost.

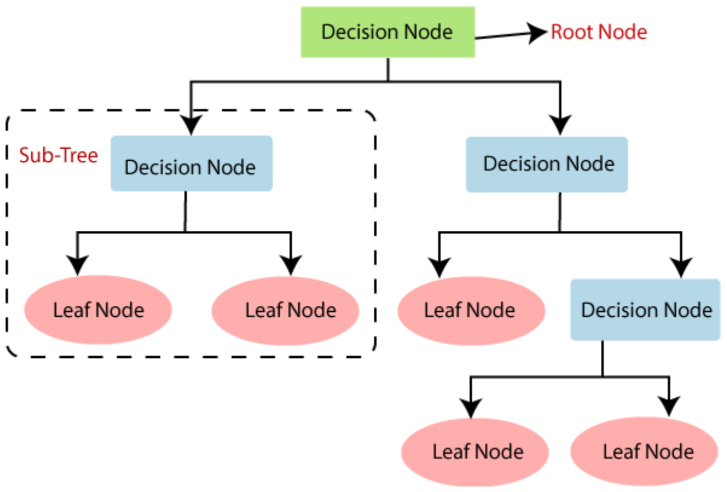
The cost function in a decision tree classifier is called “Gini Index Function”.

$$E = \sum (p_k * (1 - p_k))$$

,

such as, pk = “the proportion of training instances of class k in a particular prediction node”.

The result of the function for a particular node should be 0 in order for each split output to belong to a single class. This concept is called “information gain” and as the split has a high percentage for each of the distinct target value for each output, we’ll have a better performing model as we “gained information”. This method would lead though to overfitting the data, which is why among the parameters that need to be set when building a decision tree is the number of samples for each terminal node. (if there are less samples than the specified value, the note is not considered as a final node). The training stops when the algorithm reaches the final leaves.



Source : https://www.javatpoint.com/

Pros:

- Easy to implement, understand and interpret
- Works for both classification and regression

- Works with both continuous and categorical variables
- The algorithm does not need feature scaling
- Cons:
- It can easily lead to overfitting the data which results in high variance
- Not suitable for large datasets
- It’s unstable and can be easily affected by small changes in the data

Gradient Boosting Classification

The algorithm uses “ensamble learning”, which combines individual and simple models into a new complex and more powerful one. The GBC fits an initial model to the data (ex. decision tree). The next model is going to predict the cases in which the initial model performs poorly, and thus it gets better by learning from the errors the previous model made.

The initial prediction of the data is made using the log value of the odds (the probability) and it subtracts the observed value for each observation. In the next step, a new decision tree is built and the model tries to predict the new values for the residuals, using the ones calculated in the previous step. The process finished when the algorithm reached the number of trees chosen or when additional trees fail to improve the model.

As the model is based on the Logistic regression, the objective function is the same, maximization of the log(likelihood). The log(likelihood) represents the loss function, so we compute the negative value of the equation.

In order to use the model to make predictions, the algorithm uses the predicted values of the residuals multiplied by a learning rate and it sums the value with the base log of the odds.

The learning rate is a hyperparameter with values between 0 and 1 and it is used in order to scale the contribution of each tree and thus avoid overfitting by lowering the variance.

Pros:

- Because it uses simple models, it is easy to interpret
- Reduces the chances of overfitting

Cons:

- The algorithm is sensitive to outliers/noise

Random Forest Classifier

This method adds more flexibility to the decision tree algorithm, and it comes also with improved accuracy on the models. The random forest is built from multiple individual decision trees, where each tree uses different data and features to be trained, each tree predicting only one class.

The process of randomly selecting a sample from the dataset is called bagging. For this, each tree will have the same sample size, but at the same time it will be a random sample each time, with the mention that the observations can be repeated in the sample.

Besides using random samples for each tree, the algorithm also implies that each tree can choose features from a random subset of them, concept called feature randomness. This will imply low correlation and more diversification compared to the simple decision trees.

As in the decision tree algorithm, the random forest classifier also uses the Gini Index classifier for choosing the optimum split of the data when building the trees. This measure is calculated using the sum of squared probabilities (p_i) of each class from the target variable, as per the formula bellow:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Source : https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb

In addition to the Gini Index, Entropy is also an important measure used in building the trees. The measure computes the randomness of the dataset by making use of probabilities computation. The entropy is calculated by summing the probabilities of the target feature of having a specific value multiplied by the probability of getting the specific value on a random choice. This can be translated in the following formula:

$$Entropy(x) = - \sum (P(x=k) * \log_2(P(x=k)))$$

Source : https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb

Now, by using this basic formula, the information gain can be computed for each feature and thus the feature that hold the largest value for the resulted information gain is used as the root node when building the.

$$InformationGain(feature) = Entropy(Dataset) - Entropy(feature)$$

Source : https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb

Pros:

- Better accuracy compared to decision trees
- Lower risk of overfitting

Cons:

- Time consuming

2. Experiment description and results

In order to find the best classification model for the churn model for Orange I used the 6 machine learning algorithms: Logistic regression, Decision Tree, KNN, Gradient boosting classifier, Random Forest Classifier, Naïve Bayer Classifier.
For each method I applied hyper-parameter tuning using Grid Search to ensure the best combination of parameters and cross validation to verify the stability of the model. In the end, the models were evaluated based on accuracy and AUC.

The grid search applied for each model implies 5 folds and different parameters for each model, as follows:

- Decision tree classifier: max depth
- KNN – n neighbors
- Gradient boosting classifier: max depth, number of estimators
- Random forest classifier: number of estimators, maximum features

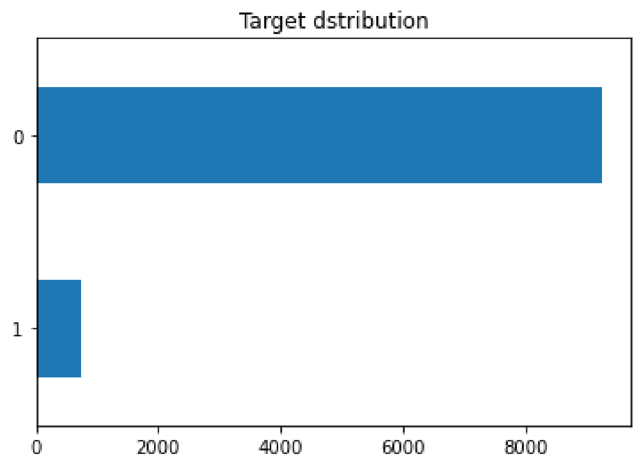
In addition to modeling, I experimented on different base tables (having a train – test - validation split of 60% – 20% – 20%):

- base table with basic preprocessing
- base table with dimensionality reduction
- base table with variable selection - Lasso regression
- base table with oversampling and variable selection - Fisher + SMOTE
- base table with oversampling and dimensionality reduction

Base table with basic preprocessing

- the table is filtered to have only records with more than 2 unique values
- the table is filtered to have only records that have the total number of null vales less than 70% of the data
- the null values are imputed with the mean value for that specific column
- the categorical values are encoded using ordinal encoder
- the Booleans variables are converted to Integer type

The initial distribution of the target variable (0-not churn, 1 – churn) is illustrated in the image bellow:



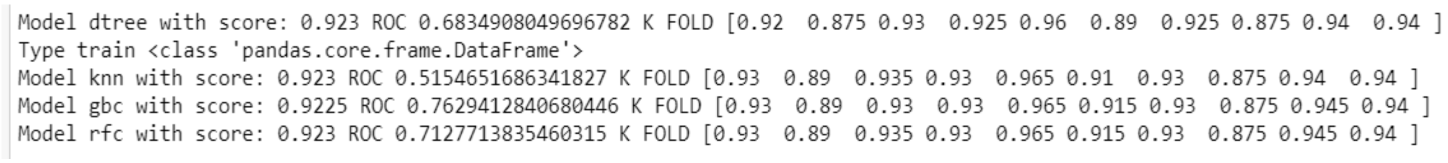
The **benchmark** experiment I considered to be the **Logistic Regression,** as it's one of less complex algorithm and it was applied to the table with only basic preprocessing steps. The results of the benchmark are the following:

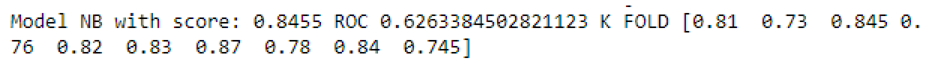
- Accuracy: 0.923
- AUC: 0.52
- KFOLD: [0.93 0.89 0.935 0.93 0.965 0.915 0.93 0.875 0.94 0.94]

The best performing model using fitted on the basic preprocessed table is the **gradient boosting classifier**, scoring:

- Accuracy: 0.923
- AUC: 0.76
- KFOLD: [0.93 0.89 0.93 0.93 0.965 0.915 0.93 0.875 0.945 0.94].

The results of the other models on the basic preprocessed table can be seen in the illustration bellow.


```
Model dtree with score: 0.923 ROC 0.6834908049696782 K FOLD [0.92  0.875 0.93  0.925 0.96  0.89  0.925 0.875 0.94  0.94 ]
Type train <class 'pandas.core.frame.DataFrame'>
Model knn with score: 0.923 ROC 0.5154651686341827 K FOLD [0.93  0.89  0.935 0.93  0.965 0.91  0.93  0.875 0.94  0.94 ]
Model gbc with score: 0.9225 ROC 0.7629412840680446 K FOLD [0.93  0.89  0.93  0.93  0.965 0.915 0.93  0.875 0.945 0.94 ]
Model rfc with score: 0.923 ROC 0.7127713835460315 K FOLD [0.93  0.89  0.935 0.93  0.965 0.915 0.93  0.875 0.945 0.94 ]
```


```
Model NB with score: 0.8455 ROC 0.6263384502821123 K FOLD [0.81  0.73  0.845 0.
76  0.82  0.83  0.87  0.78  0.84  0.745]
```

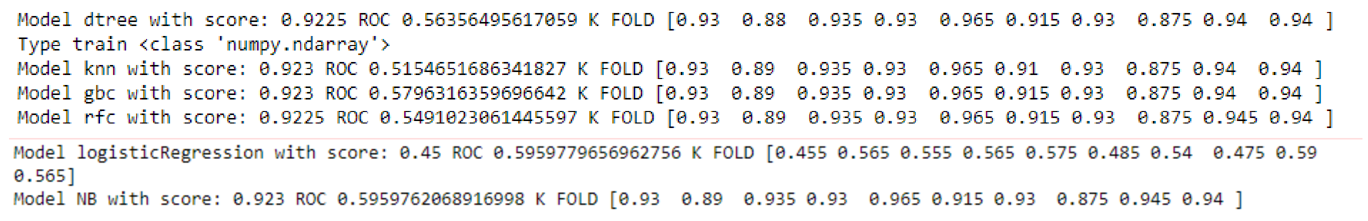
Base table with dimensionality reduction

The dimensionality reduction method used is PCA resulting in 20 principal components, out of the 115 original features.

The best performing model using this base table is still the **Gradient Boosting Classifier**, followed closely by the K- nearest neighbors. The GBC scored:

- Accuracy: 0.92
- AUC: 0.54
- KFOLD: [0.93 0.89 0.935 0.93 0.965 0.915 0.93 0.875 0.945 0.94].

Having this, by applying the PCA dimensionality reduction method, the model seems to perform approximatively the same as the benchmark model.
The results of the other models on the basic preprocessed table can be seen in the illustration bellow.


```
Model dtree with score: 0.9225 ROC 0.56356495617059 K FOLD [0.93  0.88  0.935 0.93  0.965 0.915 0.93  0.875 0.94  0.94 ]
Type train <class 'numpy.ndarray'>
Model knn with score: 0.923 ROC 0.5154651686341827 K FOLD [0.93  0.89  0.935 0.93  0.965 0.91  0.93  0.875 0.94  0.94 ]
Model gbc with score: 0.923 ROC 0.5796316359696642 K FOLD [0.93  0.89  0.935 0.93  0.965 0.915 0.93  0.875 0.94  0.94 ]
Model rfc with score: 0.9225 ROC 0.5491023061445597 K FOLD [0.93  0.89  0.935 0.93  0.965 0.915 0.93  0.875 0.945 0.94 ]
Model logisticRegression with score: 0.45 ROC 0.5959779656962756 K FOLD [0.455 0.565 0.555 0.565 0.575 0.485 0.54  0.475 0.59
0.565]
Model NB with score: 0.923 ROC 0.5959762068916998 K FOLD [0.93  0.89  0.935 0.93  0.965 0.915 0.93  0.875 0.945 0.94 ]
```

Base table with variable selection - Lasso Regression

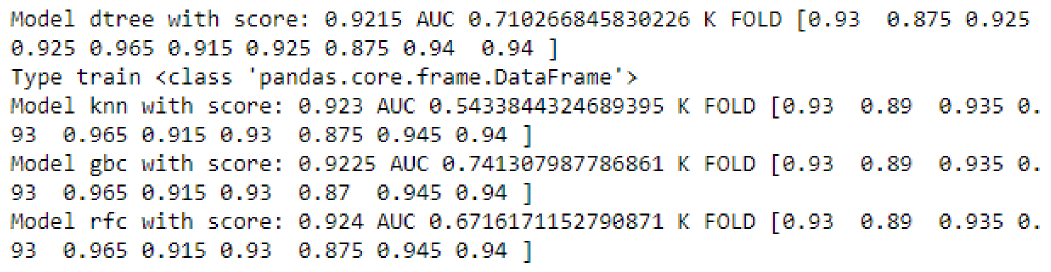
By using Lasso Regression, the number of features used for modeling decreased from 115 to 35.

The best performing model using this base table is still the **gradient boosting classifier**, followed closely by the decision tree. The GBC scored:

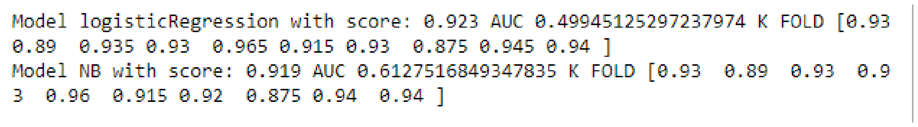
- Accuracy: 0.92 for the accuracy
- AUC: 0.74

- KFOLD: [0.93 0.89 0.935 0.93 0.965 0.915 0.93 0.87 0.945 0.94]

This model did not make a significant difference compared to the benchmark in terms of accuracy, but the AUC is significantly higher here.
The results of the other models on the basic preprocessed table can be seen in the illustration bellow.



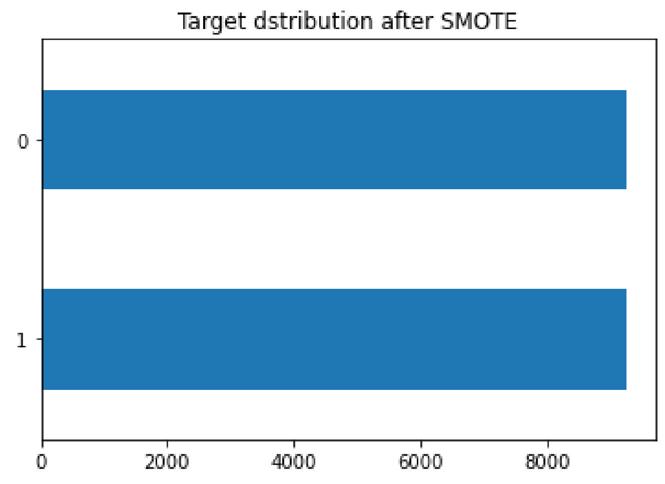
```
Model dtree with score: 0.9215 AUC 0.710266845830226 K FOLD [0.93  0.875 0.925
0.925 0.965 0.915 0.925 0.875 0.94  0.94 ]
Type train <class 'pandas.core.frame.DataFrame'>
Model knn with score: 0.923 AUC 0.5433844324689395 K FOLD [0.93  0.89  0.935 0.
93  0.965 0.915 0.93  0.875 0.945 0.94 ]
Model gbc with score: 0.9225 AUC 0.741307987786861 K FOLD [0.93  0.89  0.935 0.
93  0.965 0.915 0.93  0.87  0.945 0.94 ]
Model rfc with score: 0.924 AUC 0.671617152790871 K FOLD [0.93  0.89  0.935 0.
93  0.965 0.915 0.93  0.875 0.945 0.94 ]
```



```
Model logisticRegression with score: 0.923 AUC 0.49945125297237974 K FOLD [0.93
0.89  0.935 0.93  0.965 0.915 0.93  0.875 0.945 0.94 ]
Model NB with score: 0.919 AUC 0.6127516849347835 K FOLD [0.93  0.89  0.93  0.9
3  0.96  0.915 0.92  0.875 0.94  0.94 ]
```

Base table with oversampling and variable selection

By using Fisher Score, the number of features used for modeling decreased from 115 to 50. The oversampling method applied is SMOTE, which lead to adding artificial observations in the dataset, so as the number of records having clients that churned is equal to the number of clients that did not churn. This can be observed in the graphic bellow:

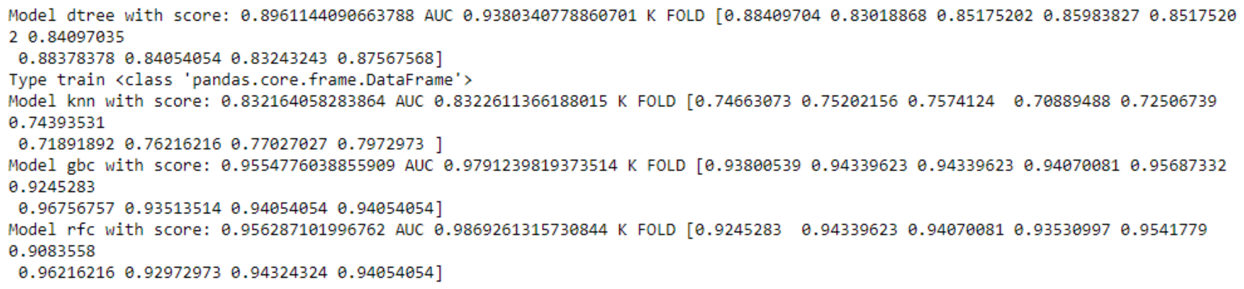


The best performing model using this base table is the **Random Forest Classifier**, followed closely by the gradient boosting classifier. The RFC scored:

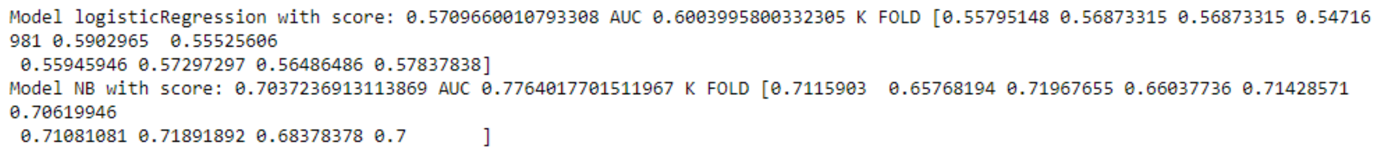
- Accuracy: 0.95
- AUC: 0.98

- KFOLD: [0.9245283 0.94339623 0.94070081 0.93530997 0.9541779 0.9083558 0.96216216 0.92972973 0.94324324 0.94054054].

Having this, by applying the Fisher selection and SMOTE, the model significantly improved compared to the ones trained before, including the benchmark model.
The results of the other models on the basic preprocessed table can be seen in the illustration bellow.



```
Model dtree with score: 0.8961144090663788 AUC 0.9380340778860701 K FOLD [0.88409704 0.83018868 0.85175202 0.85983827 0.8517520
2 0.84097035
 0.88378378 0.84054054 0.83243243 0.87567568]
Type train <class 'pandas.core.frame.DataFrame'>
Model knn with score: 0.832164058283864 AUC 0.8322611366188015 K FOLD [0.74663073 0.75202156 0.7574124  0.70889488 0.72506739
0.74393531
 0.71891892 0.76216216 0.77027027 0.7972973 ]
Model gbc with score: 0.9554776038855909 AUC 0.9791239819373514 K FOLD [0.93800539 0.94339623 0.94339623 0.94070081 0.95687332
0.9245283
 0.96756757 0.93513514 0.94054054 0.94054054]
Model rfc with score: 0.956287101996762 AUC 0.9869261315730844 K FOLD [0.9245283  0.94339623 0.94070081 0.93530997 0.9541779
0.9083558
 0.96216216 0.92972973 0.94324324 0.94054054]
```



```
Model logisticRegression with score: 0.5709660010793308 AUC 0.6003995800332305 K FOLD [0.55795148 0.56873315 0.56873315 0.54716
981 0.5902965  0.55525606
 0.55945946 0.57297297 0.56486486 0.57837838]
Model NB with score: 0.7037236913113869 AUC 0.7764017701511967 K FOLD [0.7115903  0.65768194 0.71967655 0.66037736 0.71428571
0.70619946
 0.71081081 0.71891892 0.68378378 0.7        ]
```

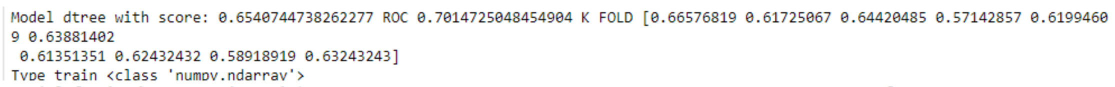
Base table with oversampling and PCA

By using PCA, the number of features used for modeling decreased from 115 to 20.
SMOTE was applied to treat the imbalance dataset

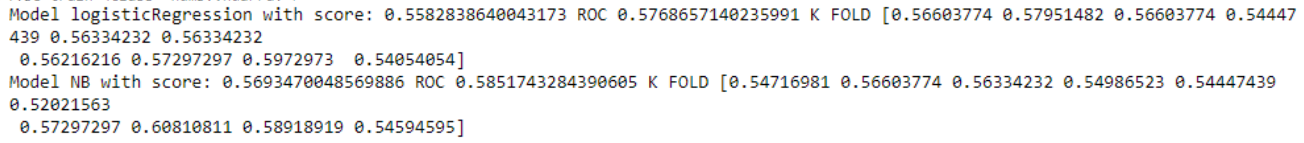
The best performing model using this base table is the **Random Forest Classifier**, followed by the gradient boosting classifier. The RFC scored:

- Accuracy: 0.91 for the accuracy
- AUC: 0.97
- KFOLD: [0.85983827 0.81401617 0.83557951 0.8032345 0.8032345 0.81401617
- 0.84594595 0.83783784 0.83513514 0.85405405].

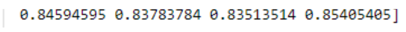
Having this, by applying the PCA and SMOTE, the model significantly improved compared to the benchmark in terms of AUC.
The results of the other models on the basic preprocessed table can be seen in the illustration bellow.



```
Model dtree with score: 0.6540744738262277 ROC 0.7014725048454904 K FOLD [0.66576819 0.61725067 0.64420485 0.57142857 0.6199460
9 0.63881402
 0.61351351 0.62432432 0.58918919 0.63243243]
Type train <class 'numpy.ndarray'>
```



```
Model logisticRegression with score: 0.5582838640043173 ROC 0.5768657140235991 K FOLD [0.56603774 0.57951482 0.56603774 0.54447
439 0.56334232 0.56334232
 0.56216216 0.57297297 0.5972973  0.54054054]
Model NB with score: 0.5693470048569886 ROC 0.5851743284390605 K FOLD [0.54716981 0.56603774 0.56334232 0.54986523 0.54447439
0.52021563
 0.57297297 0.60810811 0.58918919 0.54594595]
```



```
 0.84594595 0.83783784 0.83513514 0.85405405]
```

3. Conclusion

The way the data is transform has a high impact on the quality of the model. As observed, treating the imbalanced problem in the dataset, and applying variable selection methods like Fisher Score leads to a better performing model in terms of the AUC. Also, applying dimensionality reduction methods like PCA helps models to perform better. In addition, the data transformations seemed to have little to no impact to the model stability, as the cross validation resulted in close values as the result on the test set throughout all tested scenarios.

To conclude with the data analysis and data transformations are very important in the machine learning pipeline and each dataset should be treated uniquely and rigorously before fitting models. Next, when applying ML models, they need to be chosen for the specific situation (classification, regression, or unsupervised learning scenario) and also optimized. For this, grid search is a really helpful method for tuning hyperparameters, but it can be very time consuming and computationally expensive. In the end, the models can be tested and compared using specific metrics for each type of modeling and they need to be also tested for stability and potential overfitting using cross validation methods.

<u>References:</u>

Statistics and ML Course materials
https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
https://towardsai.net/p/machine-learning/why-choose-random-forest-and-not-decision-trees
https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d
https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80
https://corporatefinanceinstitute.com/resources/knowledge/other/boosting/
https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/
https://blog.paperspace.com/gradient-boosting-for-classification/
https://thatascience.com/learn-machine-learning/gini-entropy/
https://ai.plainenglish.io/the-math-behind-knn-7883aa8e314c