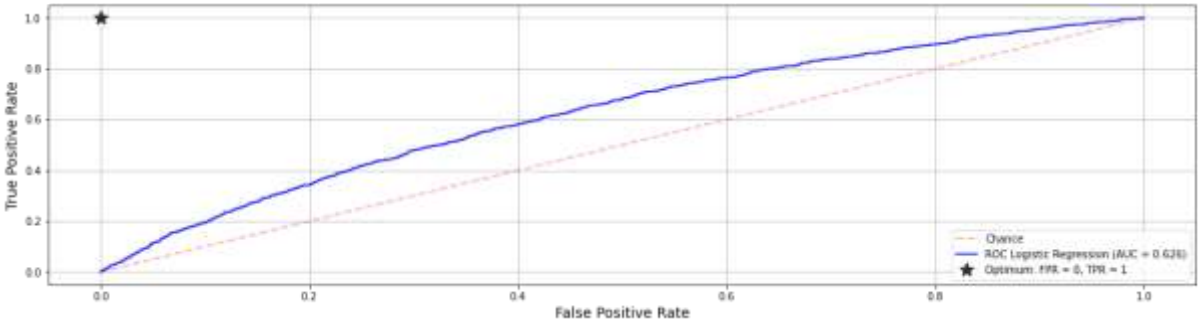
QoE estimation-MRN Project

GIACOMO SGUOTTI - 10667547

Data Analysis

- Classes are imbalanced in the train set (67.3% vs 32.7%)
- 12 features with no nan values
- From conditionals CDF plots and correlations, features relative to YouTube Session seems to have no relations with the QoE class even if the QoE valued is the one relative to YouTube.

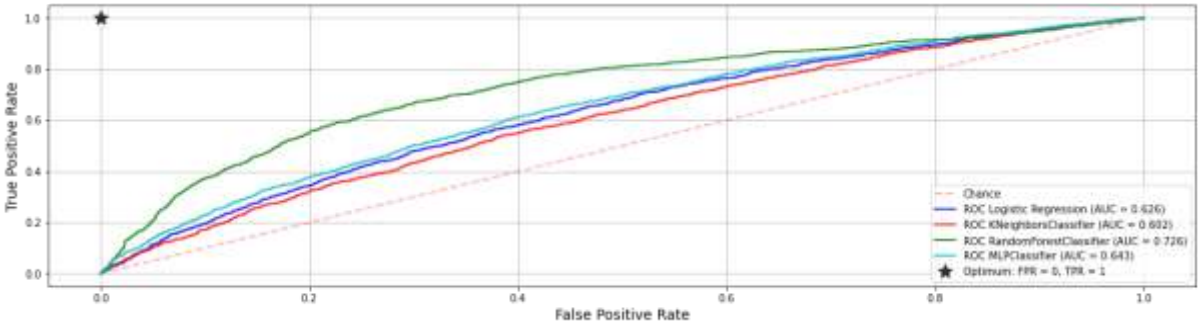


Firstly, I decided to leave the notebook as it was without dropping or adding any feature to have an initial point of reference on performances.

Very bad performances.

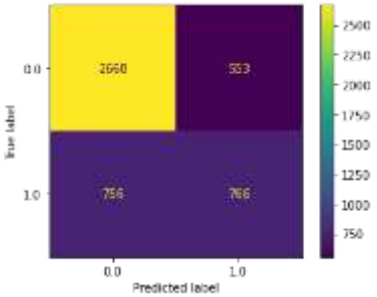
What about, before taking hands on the dataset, trying with different ML algorithms?

Let's try with KNN, Random Forest Classifier and a Multilayer perceptron.



Random Forest Classifier seems to perform much better than the others, as stated in provided papers.

From the AUC curve I noticed that the two classes performed differently, so I plotted the confusion matrix to dig more into the details.



From the Confusion Matrix, it turned out that when the RFC guess the class 1, it is right less than 60% of the times.

I supposed it might be caused by class imbalance, I thought about two different methods to tackle with this problem:

- **Feature Selection, Combination and Transformation**
- **Synthetic Minority Oversampling Technique (SMOTE)**

First Approach

Feature Combination

I decided to consider three additional features created from the combination of the already existing ones:

Percentage of Service Time spent on YouTube

As Sum of YouTube Session Times / Sum of Service Times
Correlation: 0.038

Why I introduced this feature?
I thought that the more enjoyable is the service, the more likely you are to spend time with it.
I discarded it immediately because the conditioned CDFs where nearly overlapped and by looking at the distribution, it had a very low variance.

Average rate on YouTube

As Sum of YouTube DL Volumes / Sum of YouTube DL Times
Correlation: -0.050

Why I introduced this feature?
Higher rates allow better video quality and less buffer underflows occurs.

Percentage of Time spent in Full/Limited/No Services

As Cumulative Full/Lim/No Service Times / Sum of Cumulative Service Times.
Correlation: -0.094, 0.098, 0.068

Why I introduced these features?
Suggested in papers.

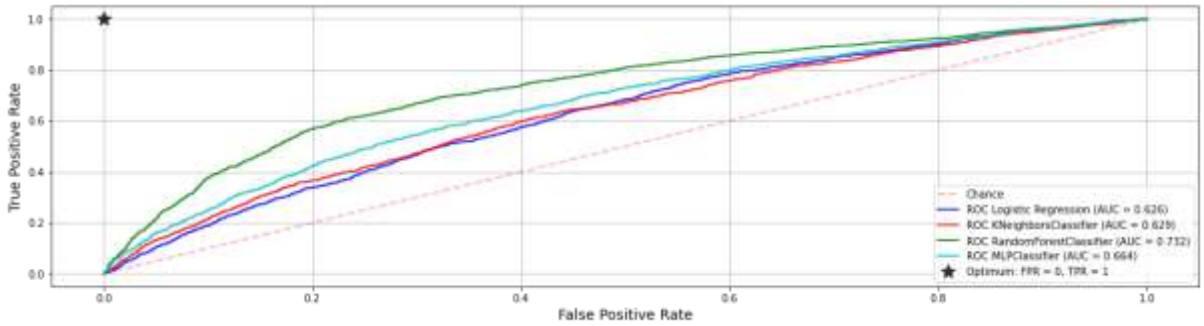
LTE Service Percentage

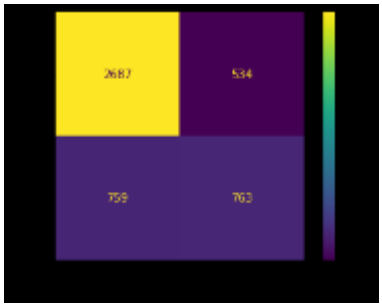
As Cumulative Full+Lim+No Service Times in LTE/ Sum of Cumulative Service Times.
Correlation: 0.022

Why I introduced this feature?
LTE brings higher rates and lower delays.

Performances

All of them had good correlation values with the ground truth, the AUC ROC metric slightly increased.



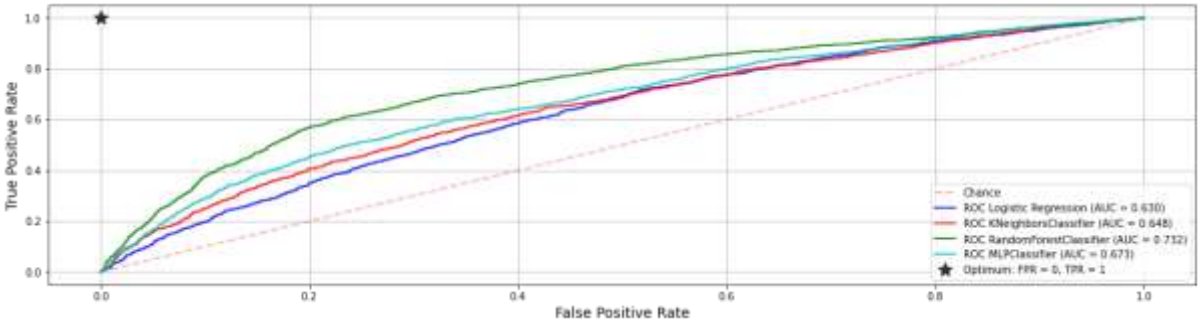


Feature Transformation

By plotting the feature distributions, it is possible to notice that some of them do not resemble in the least a Gaussian distribution (Cumulative_YoutubeSess_LTE_DL_Time, Cumulative_YoutubeSess_LTE_DL_Volume, Cumulative_YoutubeSess_UMTS_DL_Time, Cumulative_YoutubeSess_UMTS_DL_Volume, Max_SNR).

I applied log like transformations to some of them:

- log (x+1) to Cumulative_YoutubeSess_LTE_DL_Time/Volume
- -log(1-x) to Full/Lim/No_Service_Percentage as written in provided papers



The performance obtained are the same in RF but increased in KNN and MLP.

Feature Selection

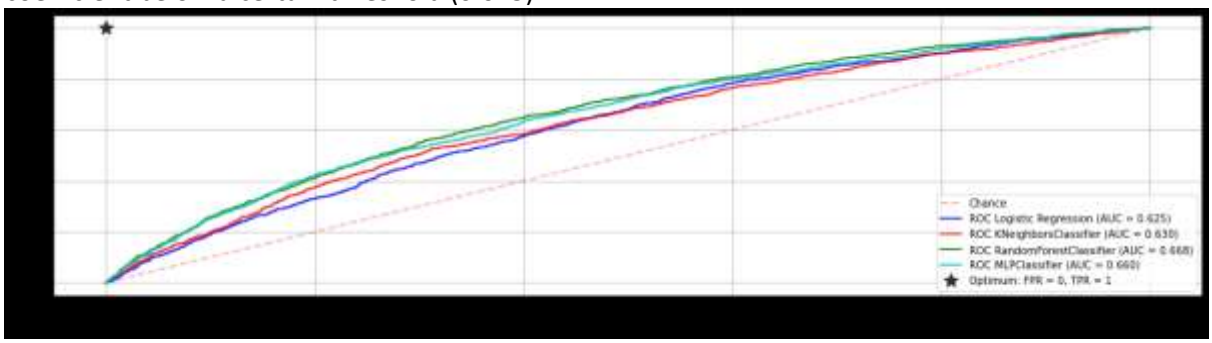
Keep only newly added features

Feature Selection method mainly used to see the goodness of the Feature Combination approach

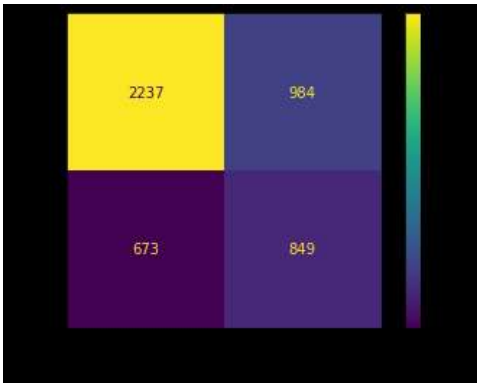
The performance metrics obtained worsened, but they are still acceptable meaning that the newly added features are able to describe the QoE of the user.

Pearson's Correlation Coefficient ≤ 0.015

At first, I thought about removing features with an absolute value of Pearson's Correlation coefficient below a certain threshold (0.015).



The features discarded, after adding features and applying transformations, are the following:
Cumulative_YoutubeSess_LTE_DL_Time
Cumulative_YoutubeSess_LTE_DL_Volume
Cumulative_YoutubeSess_UMTS_DL_Time
Cumulative_YoutubeSess_UMTS_DL_Volume



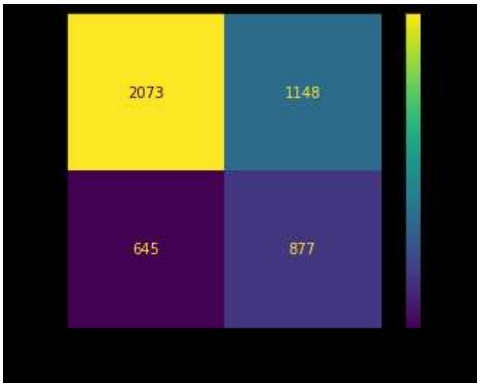
The AUC metric worsened but looking at the Confusion Matrix the number of correctly guessed users of class 1 increased.

Dropping features might be the right way to increase the performances of class 1 users, but I needed a better tool to select which one to select the right ones.

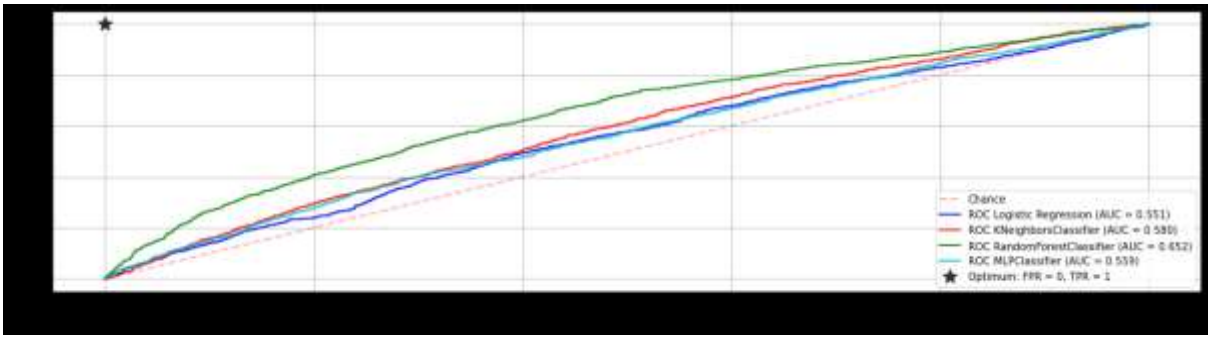
SelectFromModel

As described here, it consists in using an additional estimator to extract the most important features, used later to train the final classifier.

In my case, the model LinearSVC selected the following features:
Cumulative_YoutubeSess_LTE/UMTS_DL_Volume
Cumulative_Full/Lim/No_Service_Time_UMTS
Cumulative_Full/Lim/No_Service_Time_LTE
Average_Youtube_Rate
Full/Lim/No_Service_Percentage
Full_Service_Percentage
LTE_Service_Percentage



Features that I thought were interesting, in fact the obtained performances are worse.



The further I go the more I see a trade-off between class 1 accuracy and class 0 accuracy. The only problem was that increasing the accuracy of class lower the AUC ROC performances because the number of samples are lower.

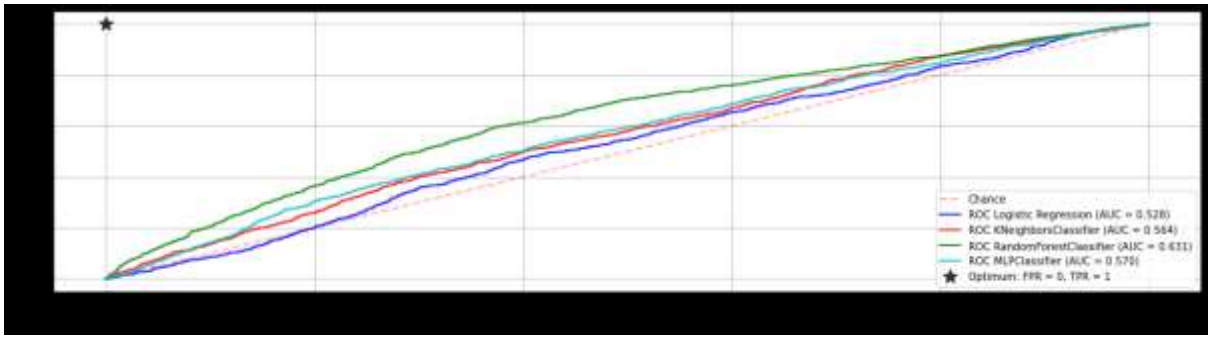
Therefore, I tried creating Synthetic samples of the minority class with SMOTE.

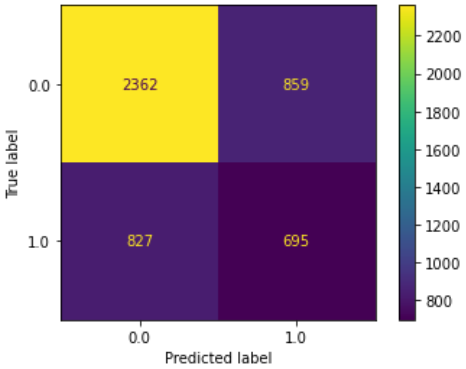
SMOTE

The Synthetic Minority Oversampling Technique (SMOTE) is a technique used in unbalanced classification problems and permits to create additional samples of the minority class to solve the unbalancing of classes.

In particular, I decided to use a variant of SMOTE oversampling technique, the Adaptive Synthetic Sampling (ADASYN) that generates synthetic samples inversely proportional to the density of the examples in the minority class.

The workflow with SMOTE was the following: oversample with SMOTE -> create new features -> transformations -> eventual feature selection (SelectFromModel below).





Performance decreased with respect to the initial configuration, in particular class 1 is particularly penalized probably by the untruthfulness of the introduced synthetic data.

Results

In the end, I also added a hyper tuned Keras model to try to see if it increases performances, but it didn't.

I tried many combinations of the described methods and the best AUC ROC metric obtained was 0.731784 with Random Forest Classifier adding the new features (no YT Time) and then transforming them.