Alma Mater Studiorum - Università di Bologna

BDA Project - Presentation
Sparkify

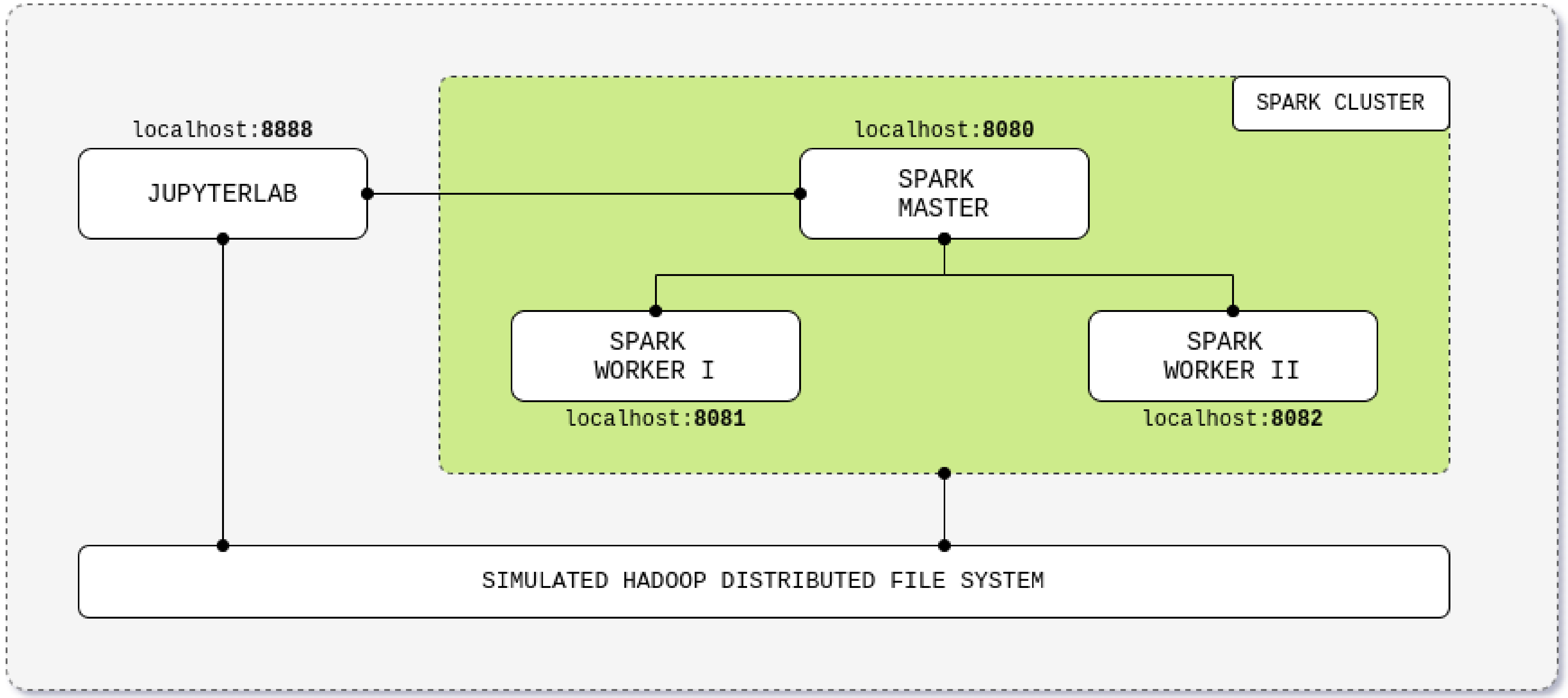
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Artificial Intelligence

STUDENT
Alessio Cocchieri
matr. 1046067
alessio.cocchieri2@studio.unibo.it

Academic year 2022-2023
FEBRUARY

Introduction

Cluster architecture



- Apache Spark Cluster on **Docker**
- Cluster mode with a JupyterLab interface built on top of Docker.
- Spark Worker node with 2 cores and 5GB of memory (default)

Sparkify

- Sparkify is a **music streaming service**
- Users can choose free tier subscription with ads or paid tier without ads.
- Users can upgrade, downgrade, or cancel the service.

Binary classfication task

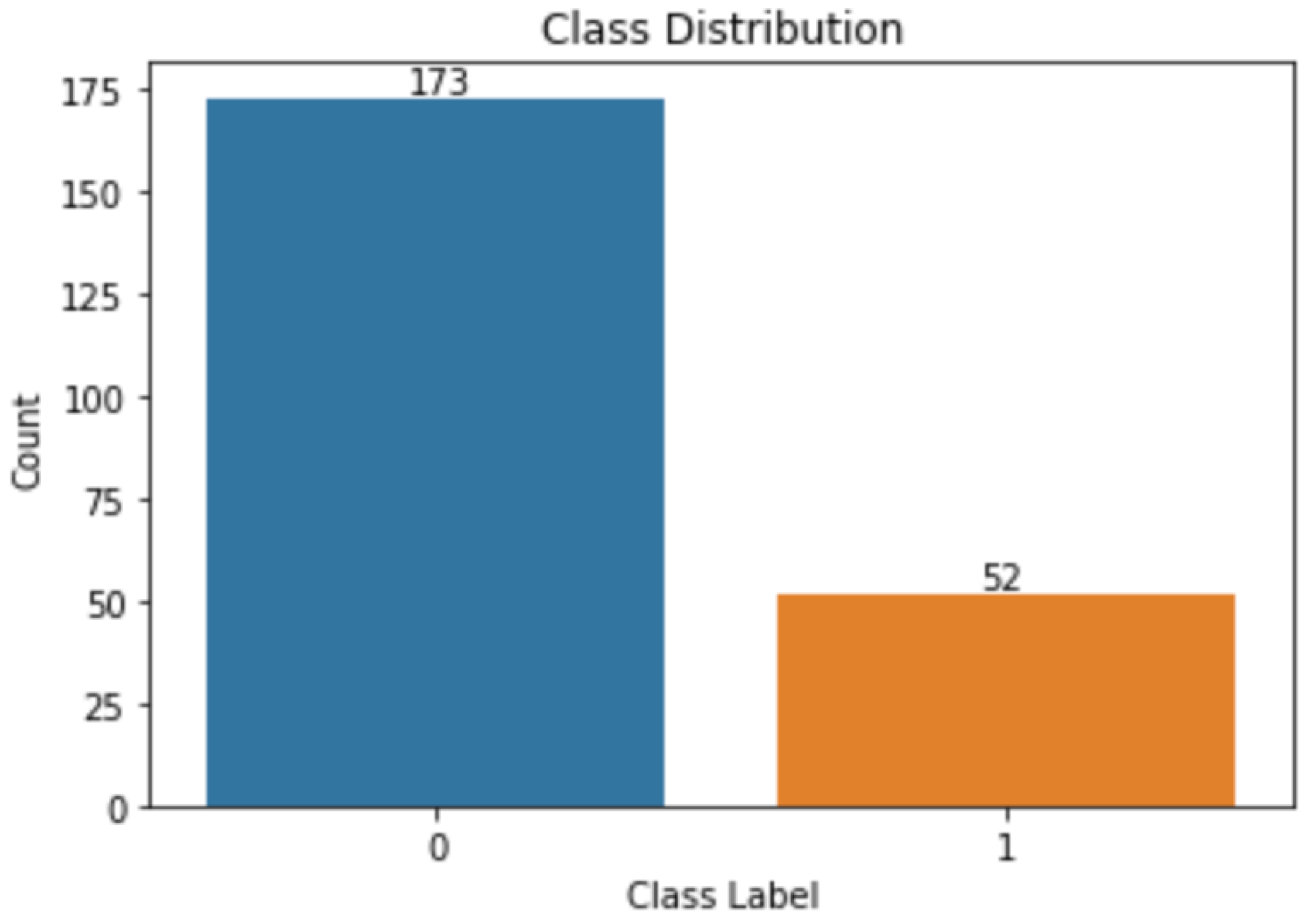
- The objective is to identify clients more likely to churn.
- If we can identify which users are at-risk to churn, then the business can take action and potentially make them stay.

Dataset

- **250600 records** and **18 features**
- The 18 features can be divided into 3 different levels:
  - User-level information (name, gender, location...)
  - Log-specific information (timestamp, status, type of interactions..)
  - Song-level information (song artist and duration)
- **225** distinct users
- **Smaller version** (128mb) of the real dataset (12gb)

artist	auth	firstName	gender	itemInSession	lastName	length	level	location	method	page	registration	sessionId	
Martha Tilston	Logged In	Colin	M	50	Freeman	277.89016	paid	Bakersfield, CA	PUT	NextSong	1538173362000	29	
Five Iron Frenzy	Logged In	Micah	M	79	Long	236.09424	free	Boston-Cambridge-...	PUT	NextSong	1538331630000	8	
Adam Lambert	Logged In	Colin	M	51	Freeman	282.8273	paid	Bakersfield, CA	PUT	NextSong	1538173362000	29	Time
Enigma	Logged In	Micah	M	80	Long	262.71302	free	Boston-Cambridge-...	PUT	NextSong	1538331630000	8	Knocking
Daft Punk	Logged In	Colin	M	52	Freeman	223.60771	paid	Bakersfield, CA	PUT	NextSong	1538173362000	29	Harder B

Data overview



High imbalanced class ratio

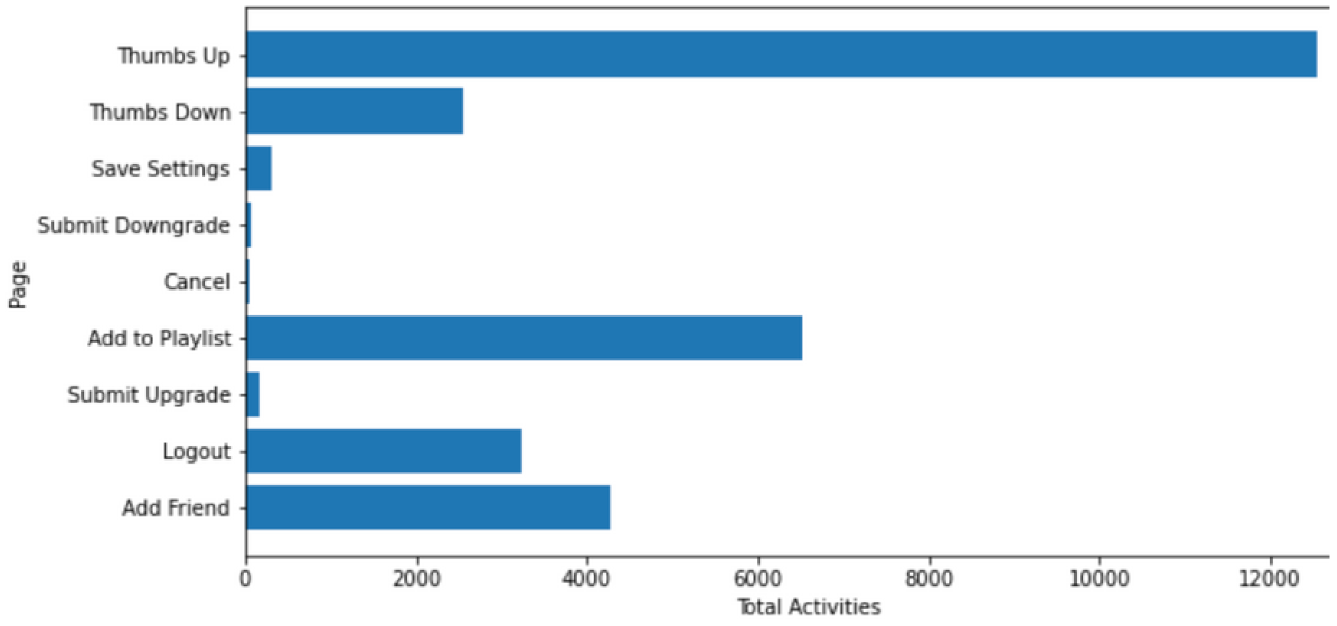
The number of instances for each label is not uniform, resulting in positive class (churn) to be only the 23.11% of the distribution

Evaluation metric

- **Accuracy** is inappropriate as evaluation metric because of the high imbalanced class ratio in the data, .
- **Precision** quantifies the number of positive class predictions that actually belong to the positive class.
- **Recall** quantifies the number of positive class predictions made out of all positive examples in the dataset.
- **F1-score** provides a single score that balances both the concerns of precision and recall in one number.
- Since **our goal is to predict the users more likely to churn** (postive class), **F1-score of positive class** has been considerend as main reference to select the best model for the current problem.

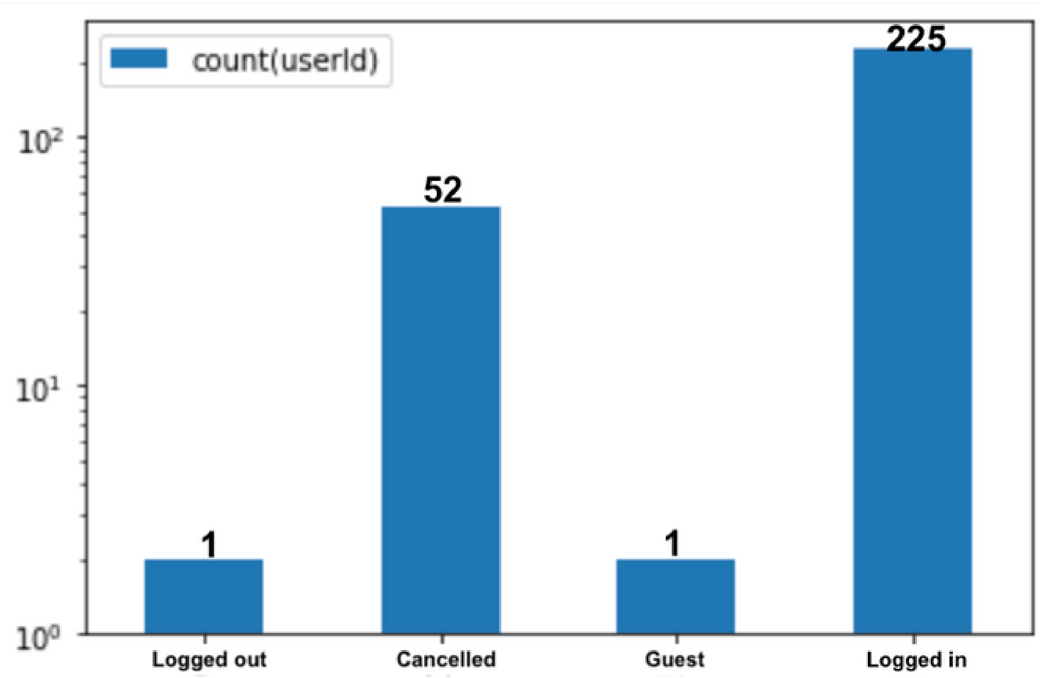
Exploratory Data Analysis

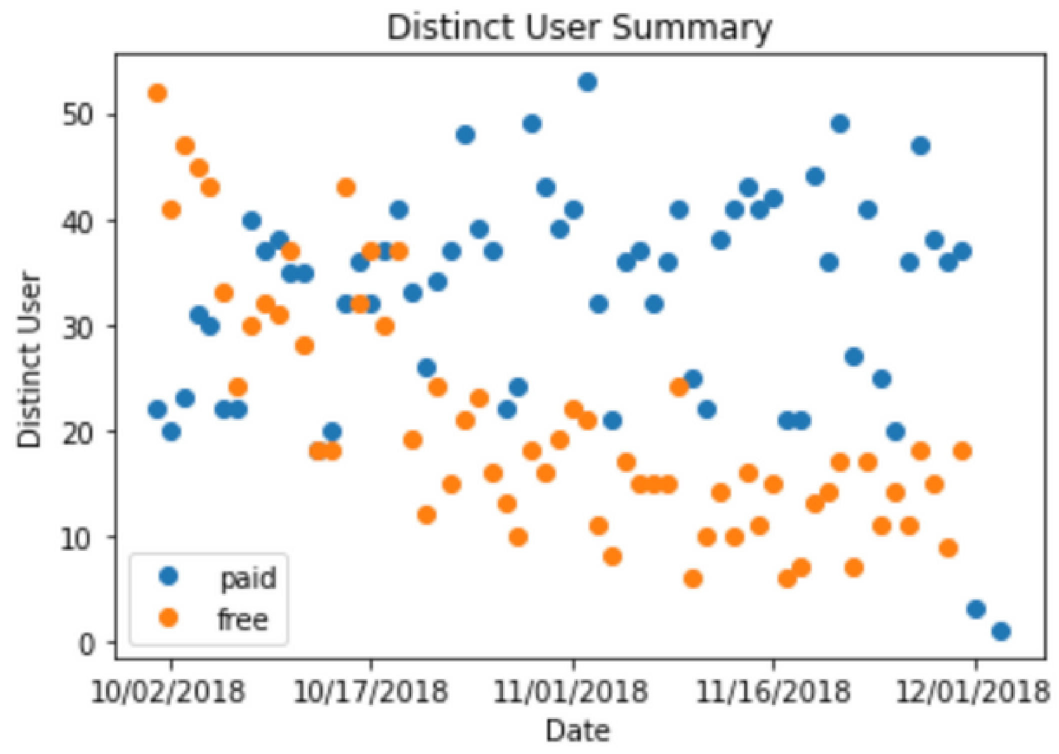
User activity

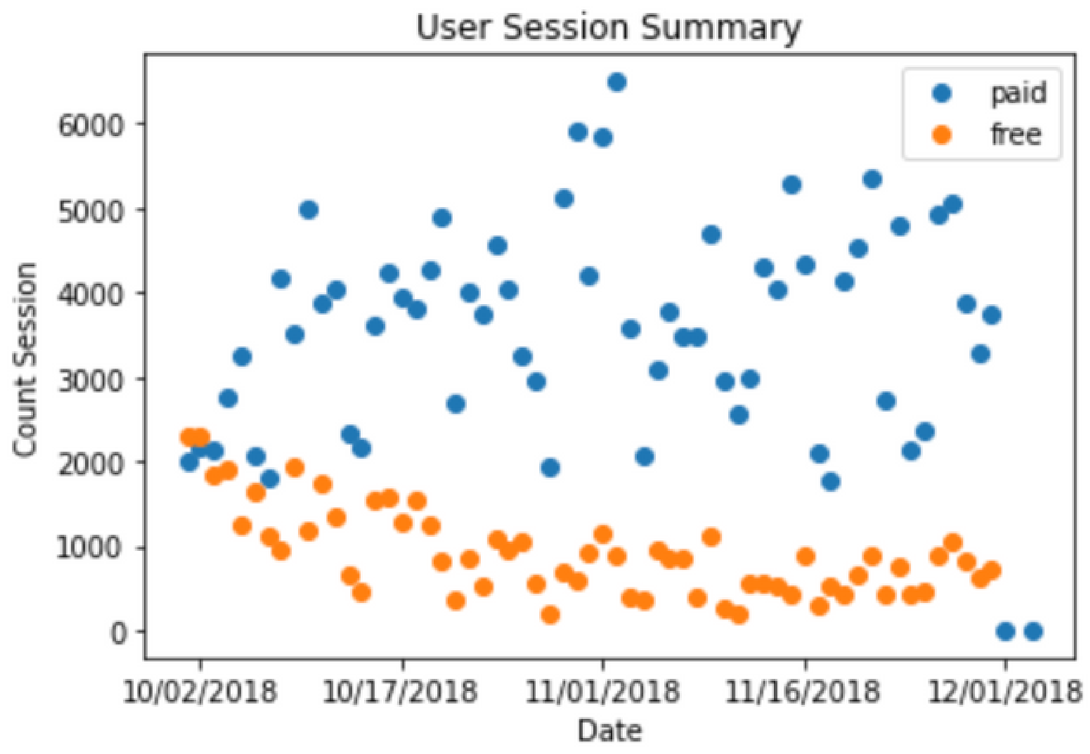


Paid/Free customers churn

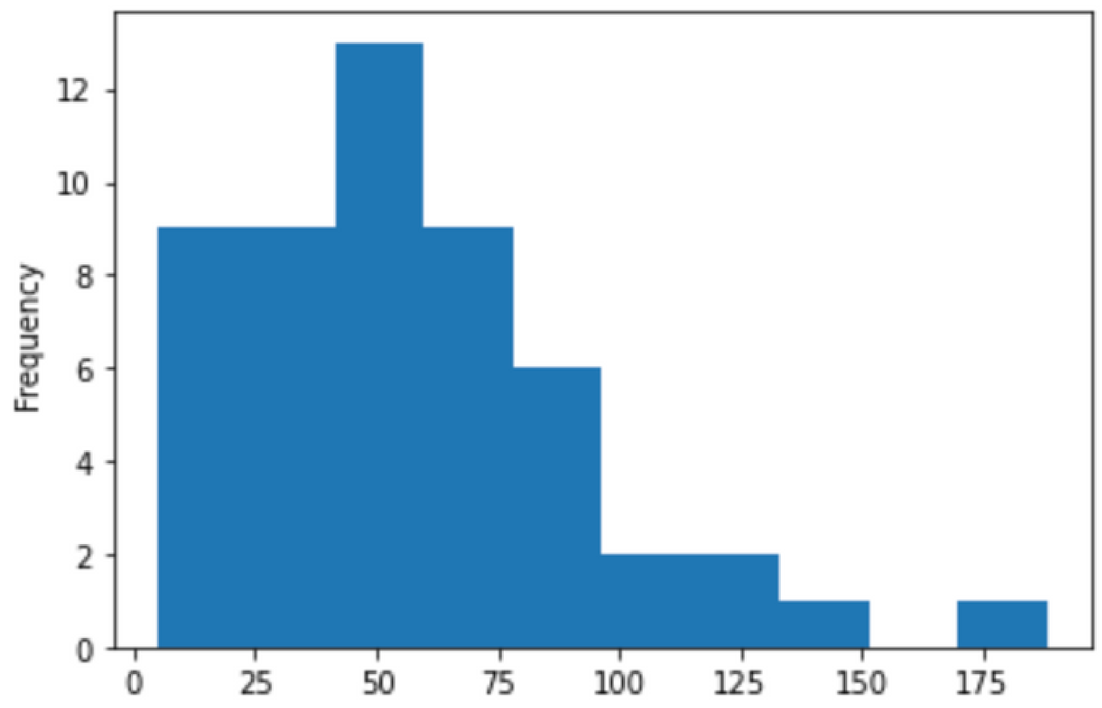
Churn definition







Most churns happens within 100 days after registration.

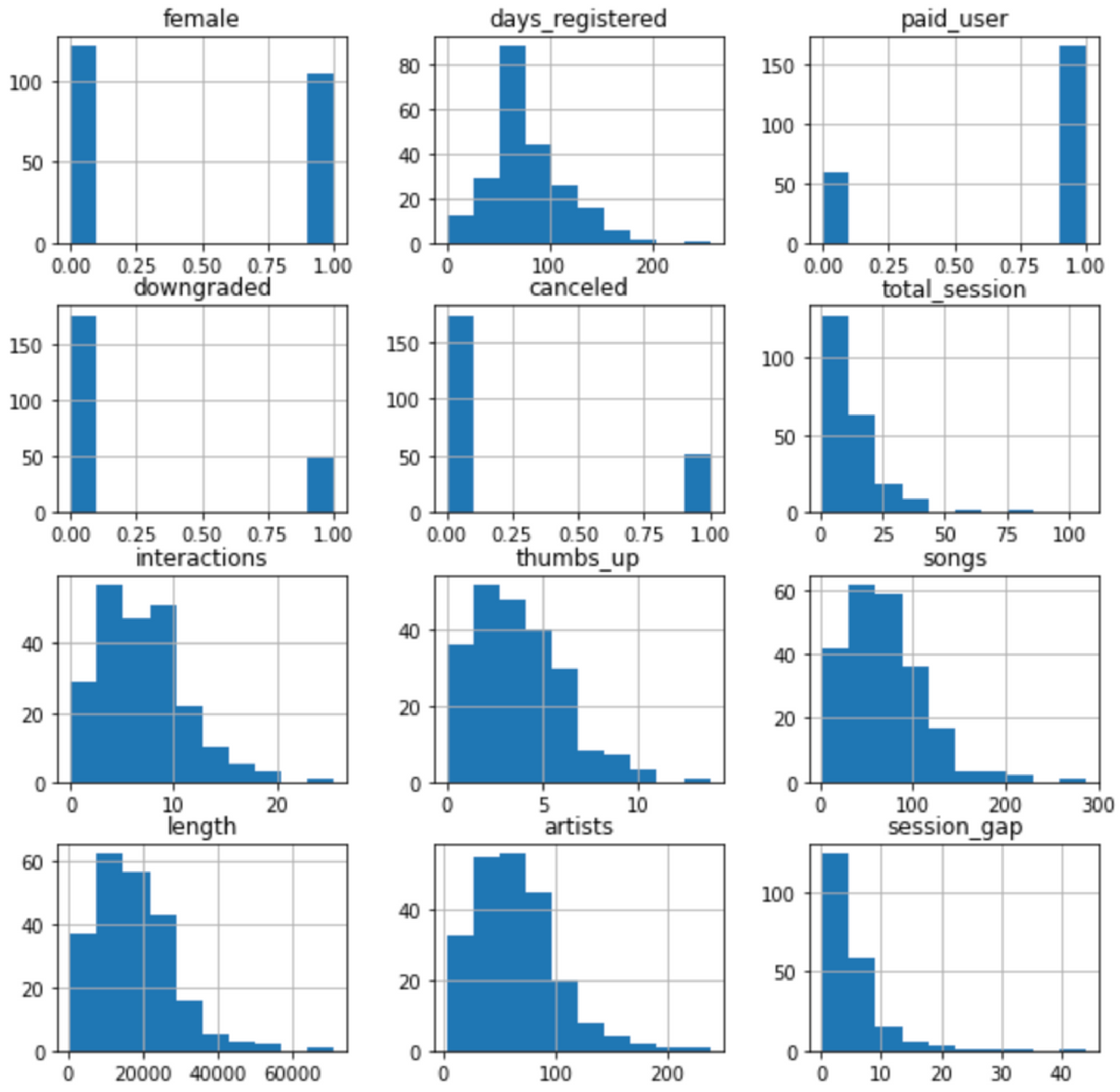


At the beginning, paid tier and free tier have similar numbers of user sessions and distinct user, but then, both metrics increase over time for the paid tier as the metrics decrease for free users.

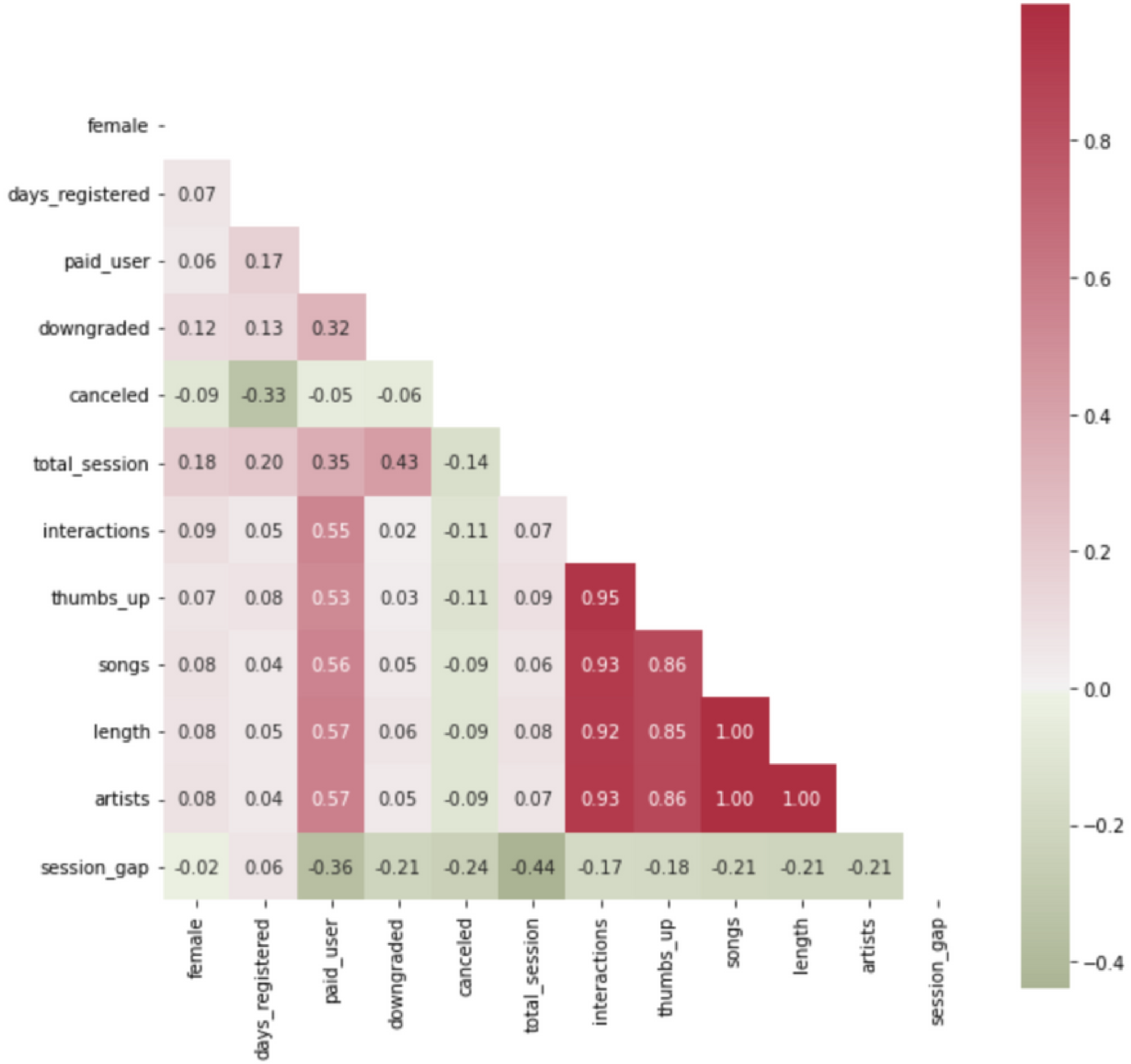
Feature Engineering

Extracted features

- **gender:** male or female
- **days_registered**: number of days the user is registered
- **paid_user**: free account or premium account
- **downgraded**: has the user ever downgraded from premium to free?
- **artists:** average number of artits listened per session by the user
- **songs:** average number of songs listened per session by the user
- **length**: average second listened of songs per session by the user
- **interactions:** average proactive operations performed by the user per session
- **thumbs_down:** average thumbs down released by the user per session
- **total_session:** total number of session
- **session_gap:** average time between each session and the pevious one

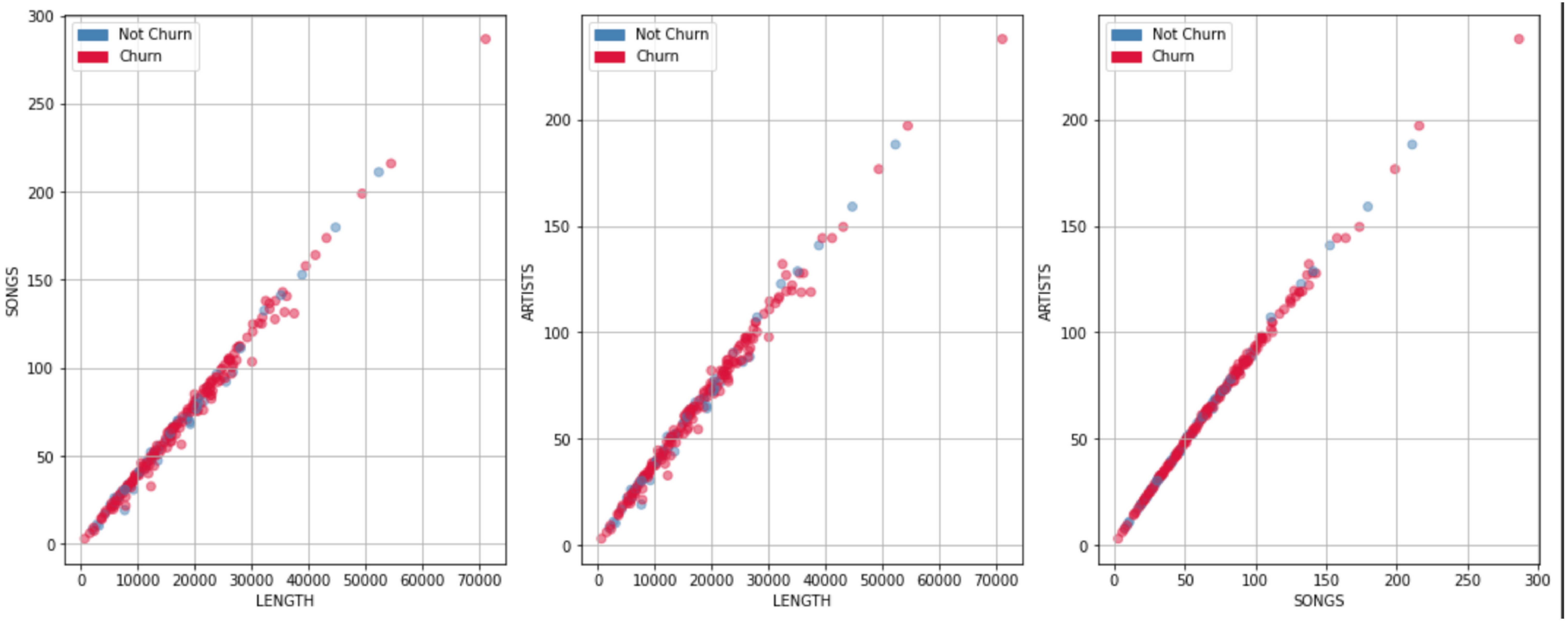


Correlation between features



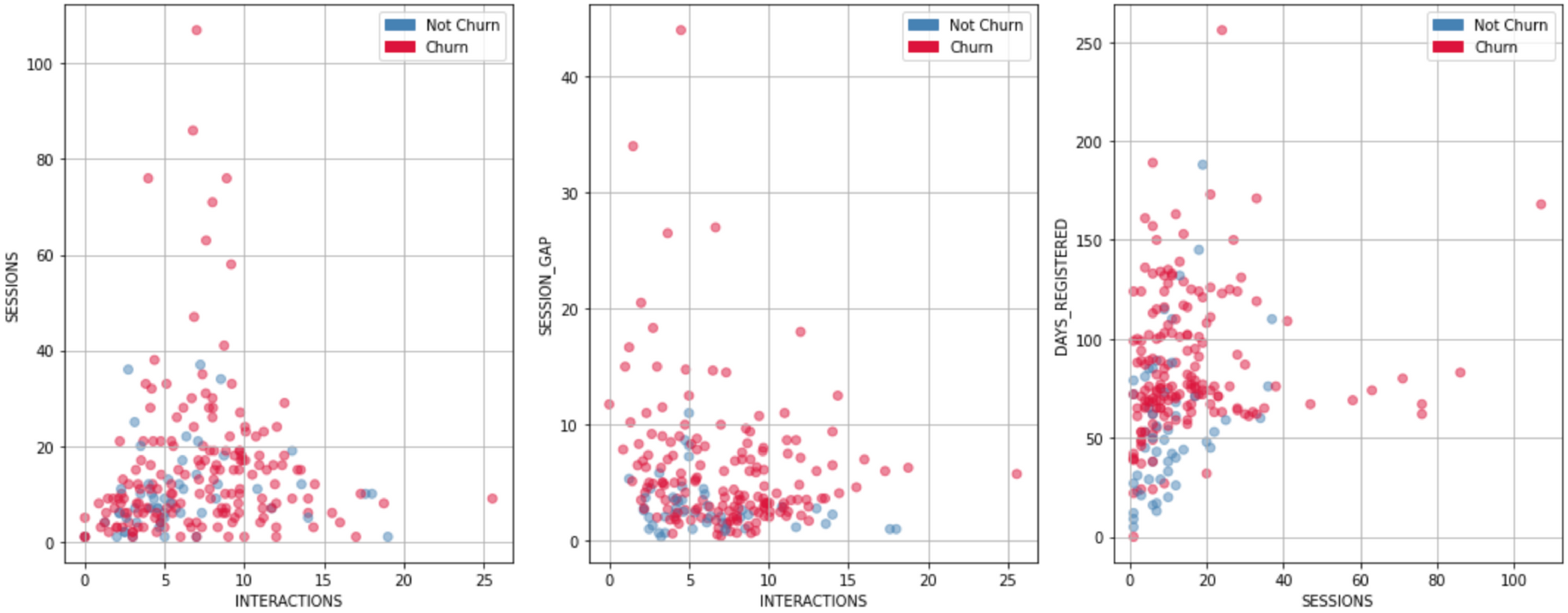
- There is no obvious strong predictor for **cancelled** except for **days_registered**
- The folllowing features are **very similar** according to the histograms:
  - songs
  - interactions
  - thumbs_up
  - length
  - artists
  - this is probably **caused by the small dataset** (225 users). If we have more data, we might see more variance in user behaviors
  - Therefore, we will **only exclude songs and artists** as they will always be similar to length.

Further proof...



... of linear dipendent features

Further proof...



... of non-correlated features

Modeling

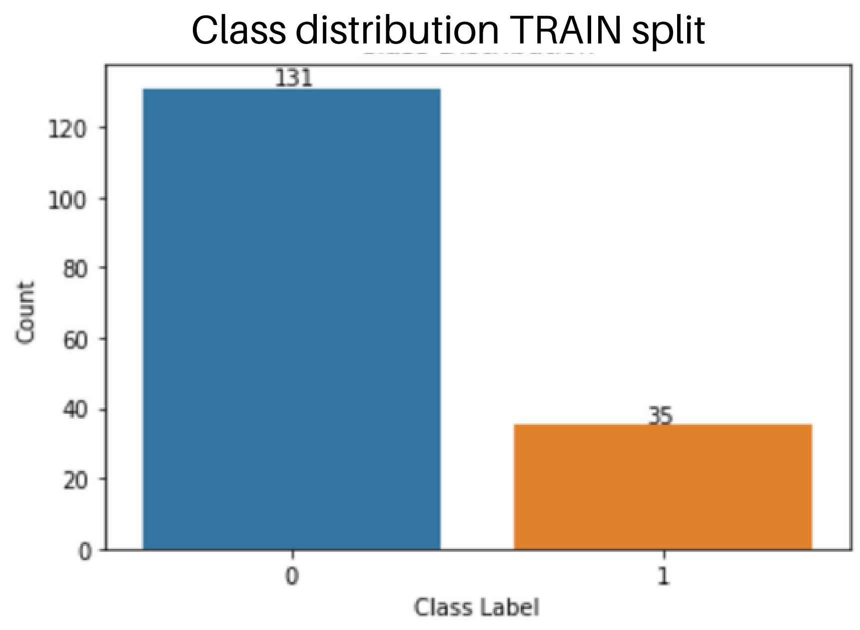
Features preparation

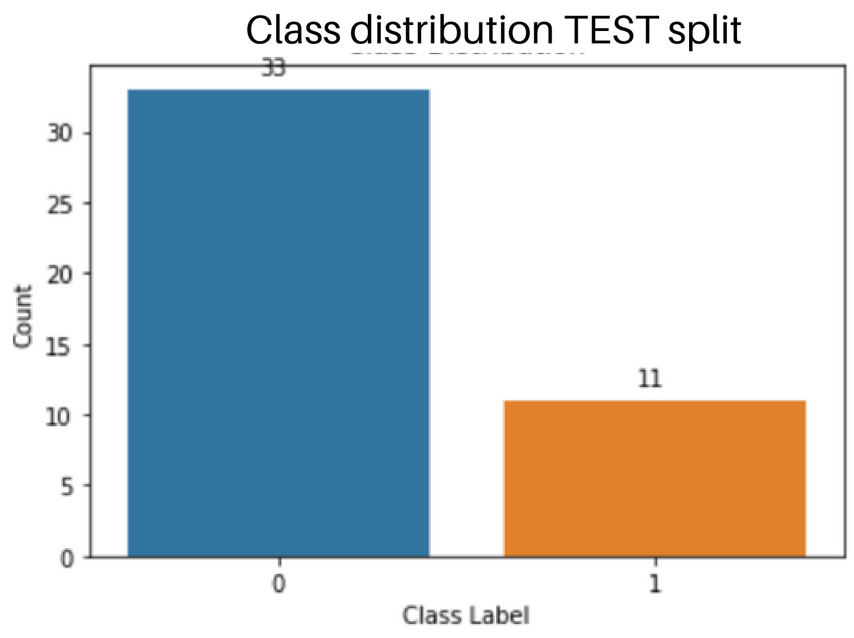
Features must be prepared in the right way in order to be fed to the model. In this regards, a couple of further steps are needed:

- **Assembling:** For each record, features must be assembled in a unique array by exploiting the function **VectorAssembler**()
- **Scaling:** After being assembled, the features must be scaled by exploiting the function **StandardScaler**().

Split train and test set

- The split has been r**andomly generated**
- However, the **seed was set** = **42** to keep the same splits for all the tested approaches, in order to:
    - make evaluations statistically meaningful
    - provide consistent results among the different classifiers.



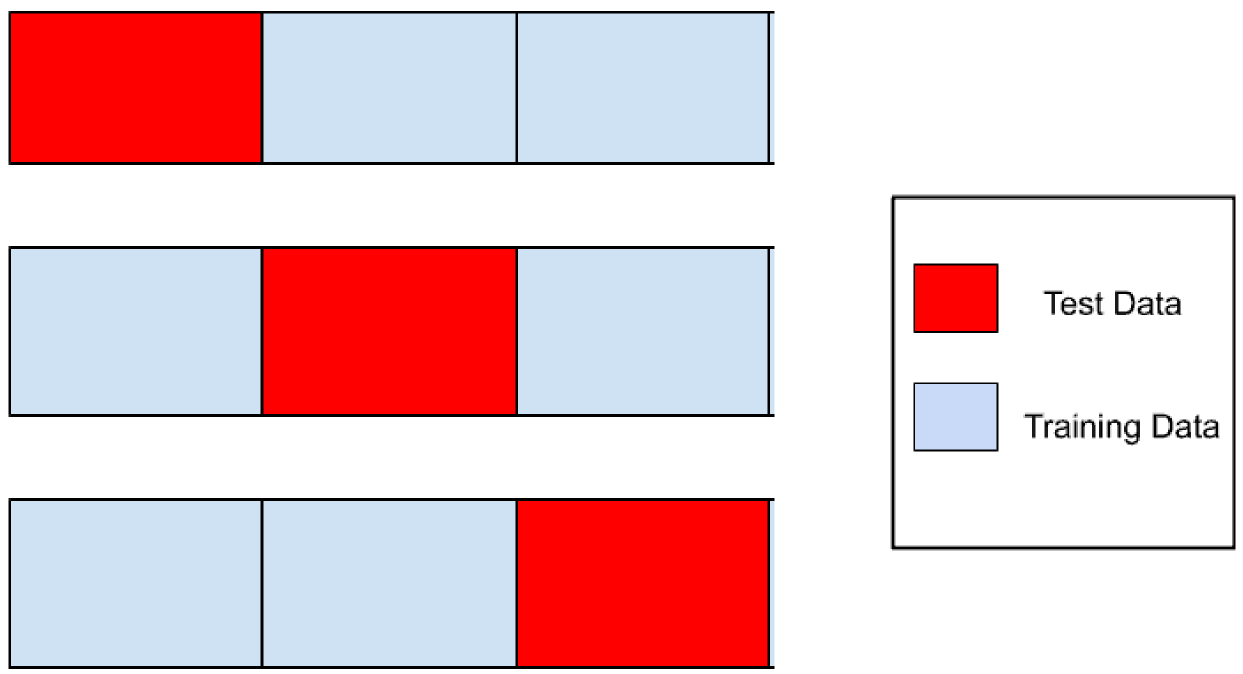


Model fine-tuning

Cross-validation

To find the best model and parameters, I use **CrossValidator** function to evaluate the model performance and validate the robustness of the models. The validator takes as input:

- the estimator (in our case a pipeline)
- the number of folds (=3),
- the paramGrid
- the evaluator



Definition of a custom evaluator

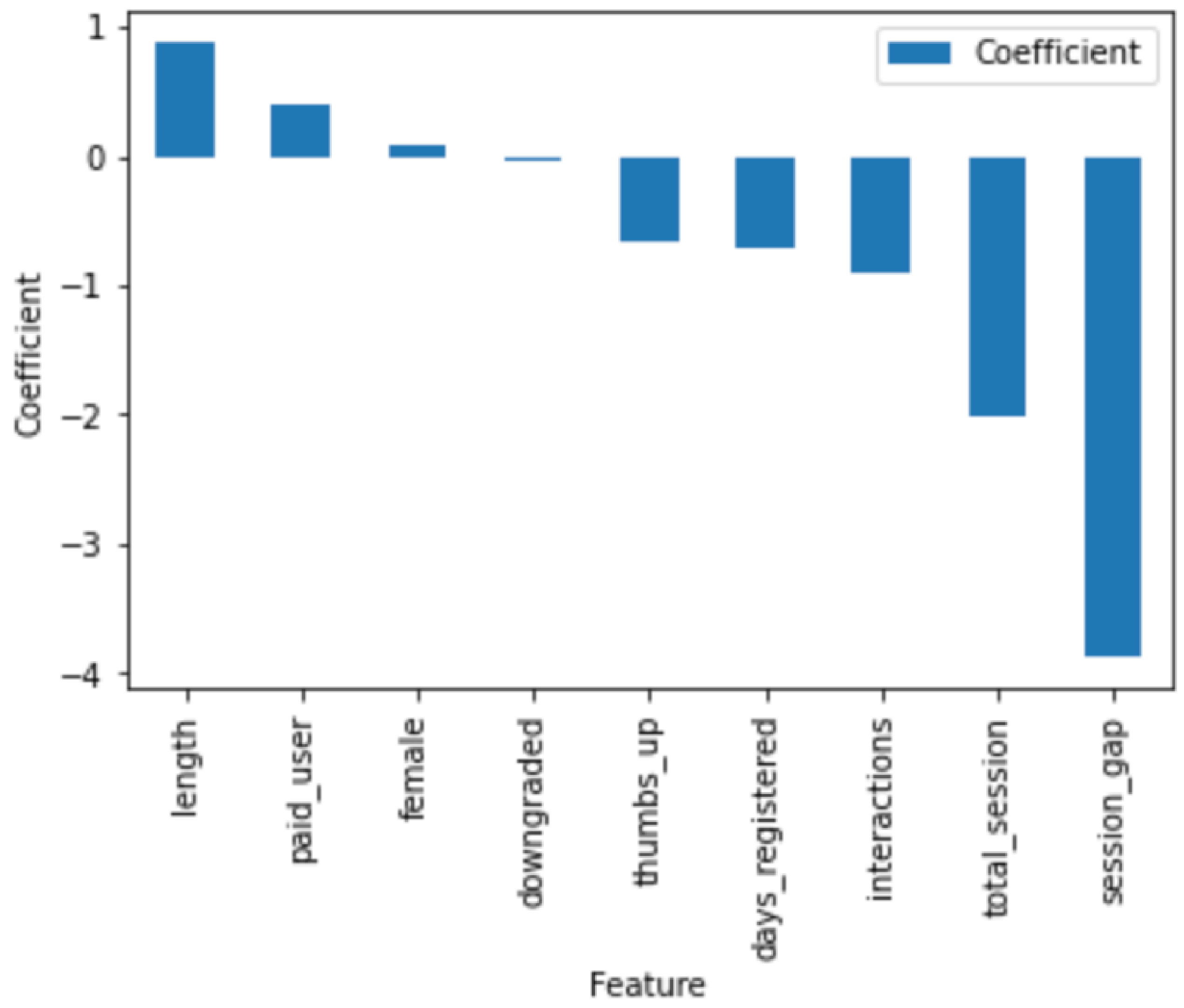
Since our objective is to maximize the ability of the model to predict users more likely to churn, I decided to define a custom evaluator which consider as reference metric only the F1 score of the positive class 1(churn).

Selected Models

- Logistic Regression
- Random Forest
- SVM (Linear SVC)
- GBT

Logistic regression

- It is a simple and easy to interpret model
- it outputs a probability between 0 and 1 that represents the likelihood of the positive class.
- The coefficients of the model represent the contribution of each feature to the predicted probability.

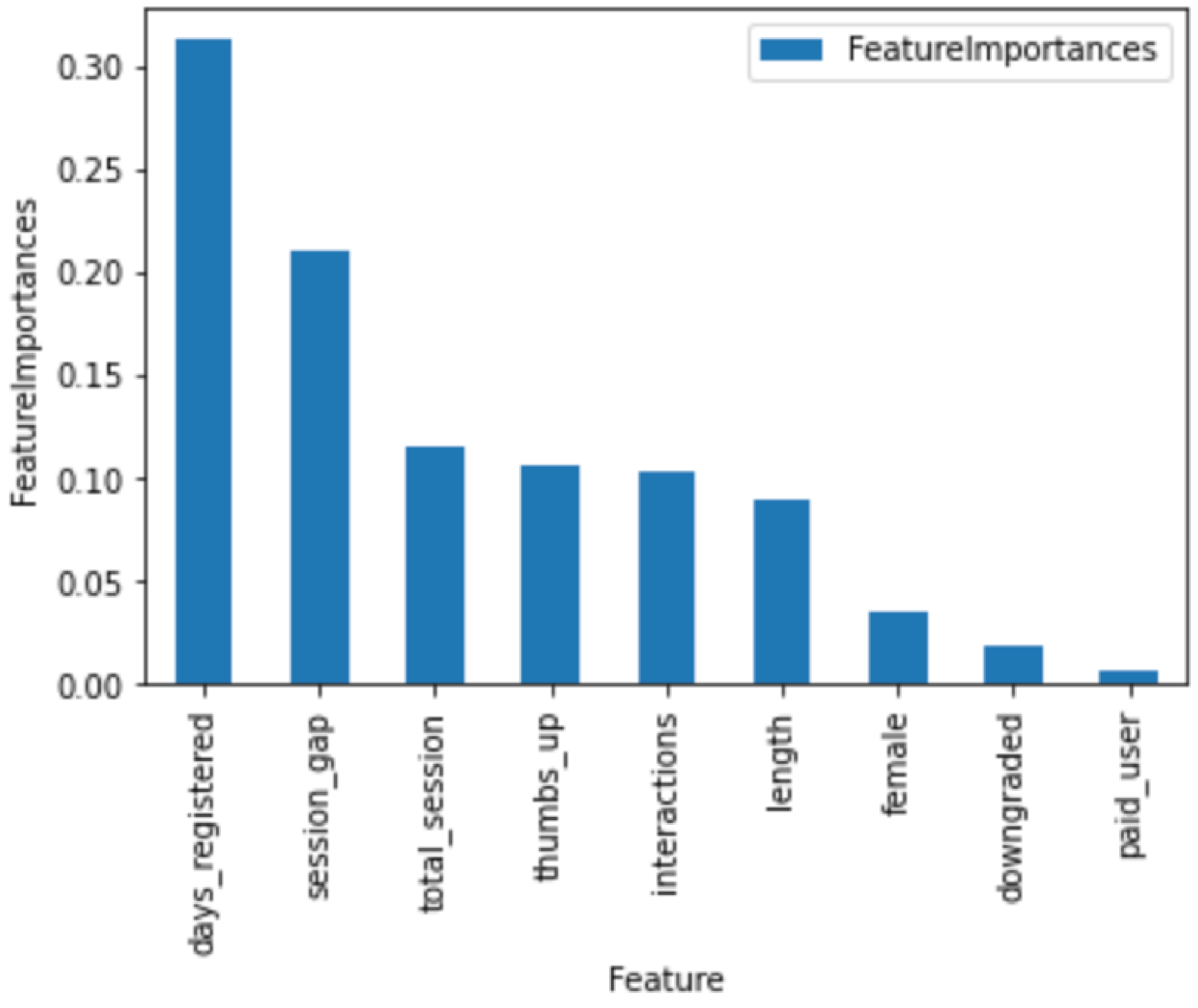


- **session_gap, total_session** and **interactions** are **inversely correlated with the probability of churn**, so the higher their value, the less the probability of the client to churn

- On the other hand, the **length** and the **typology of the user** turned out to be features **directly correlated** with the probability of being a churn.

Random Forest

- high performances in binary classification problems by combining the predictions of many decision trees.
- It provides feature importance scores

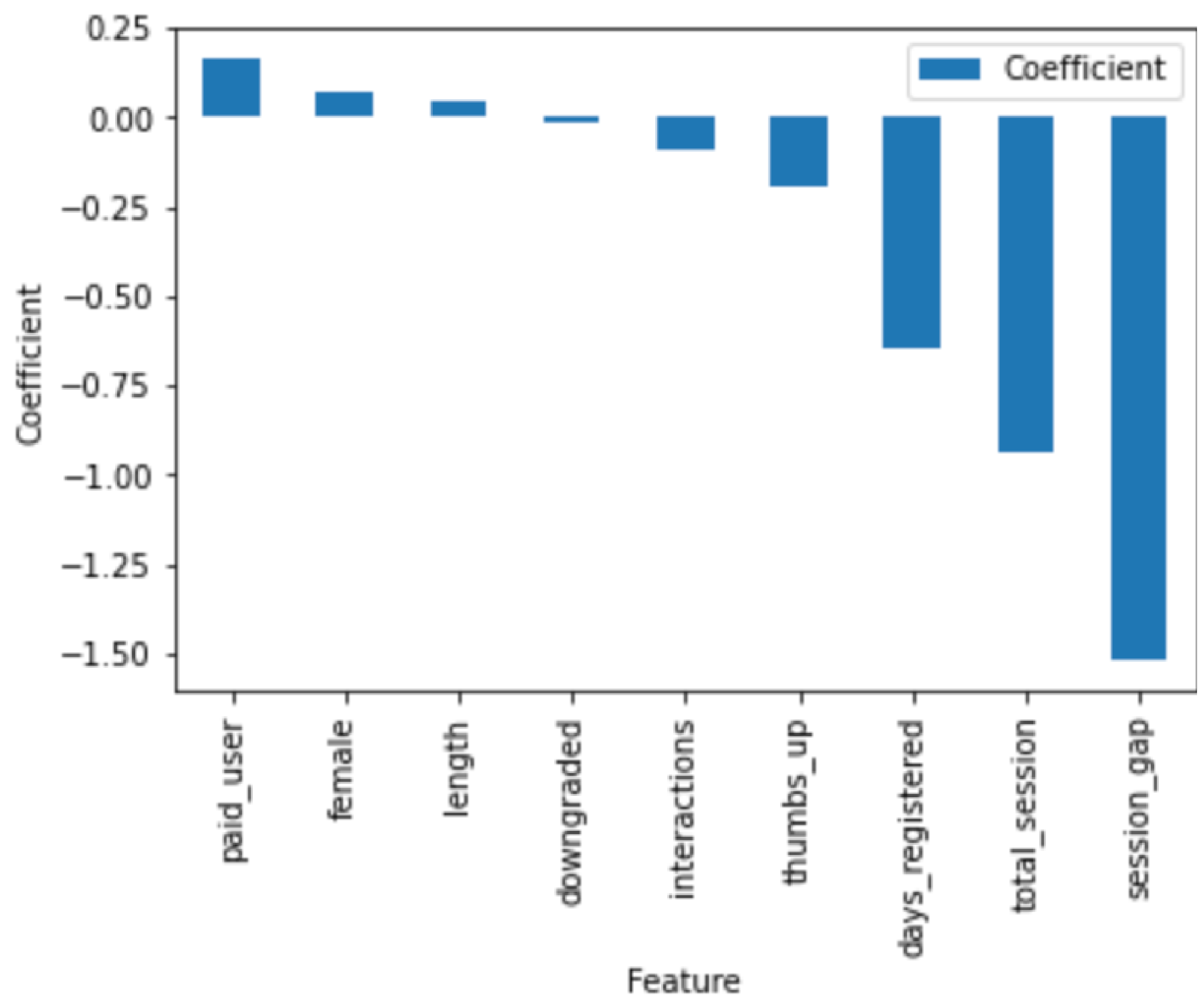


Feature importance is a measure of the contribution of each feature to the prediction accuracy of the model. From the plot we can see as :

- **paid_user** and **downgraded** are the feature with t**he least impact** on churn prediction
- **days_registered** and **session_gap** turned out to be **the most important.**

Linear SVC

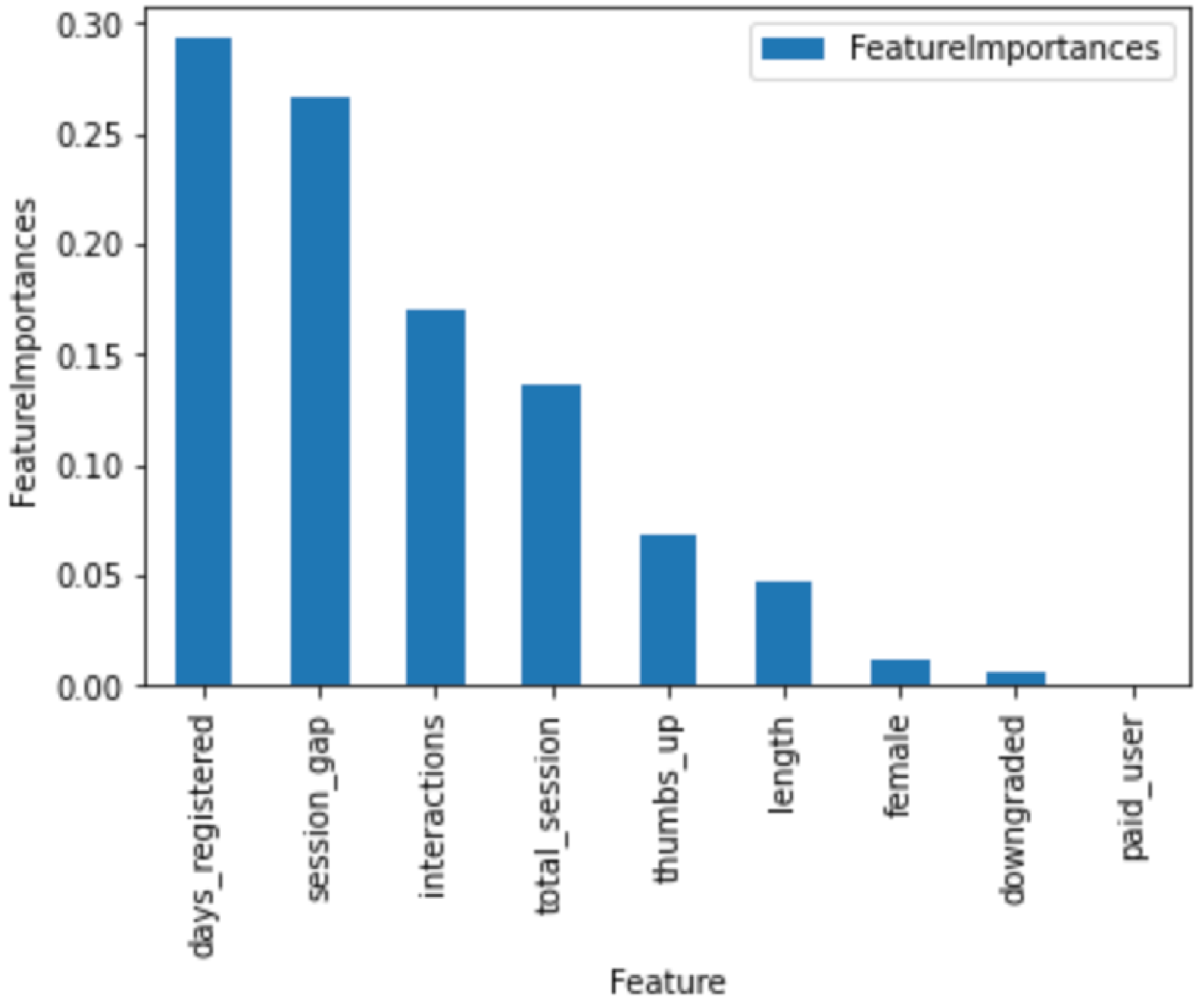
- models the relationship between the features and the target variable as a **linear boundary**, which makes it easy to visualize and interpret the relationship.
- The goal is to **find the best boundary (or hyperplane)** that separates the data into two classes.
- The coefficients in an SVC model represent the weights assigned to each feature in determining the position of this boundary.



session_gap and **total_session** have the highest impact are **the most important features** in determining the boundary.

GBT Classifier

- As it happens for Random Forest, it models **non-linear relationships**, and provides the feature importance scores.
- However, GBT classifier is generally considered **more complex** and **slower** to train with respect to Random Forest



- GBT does **not make use** of **paid_user** to predict the target.
- **Female** and **downgraded** **very low impact** in determining wether the client is a churn or not.

Results

MODEL	F1 (train)	F1 (test)
LR	0.74	0.71
RF	0.81	0.50
SVM	0.43	0.42
GBT	0.77	0.50

- **Logistic regression** resulted **the best model**
- **Random Forest** and **GBT** showed **high overfitting**, probably caused by the small amount of train data at disposal
- **Linear SVC** turned out to be **the worst model** with only 42% of F1-score for the positive class. It is the model that most suffers the high imbalanced class ratio.

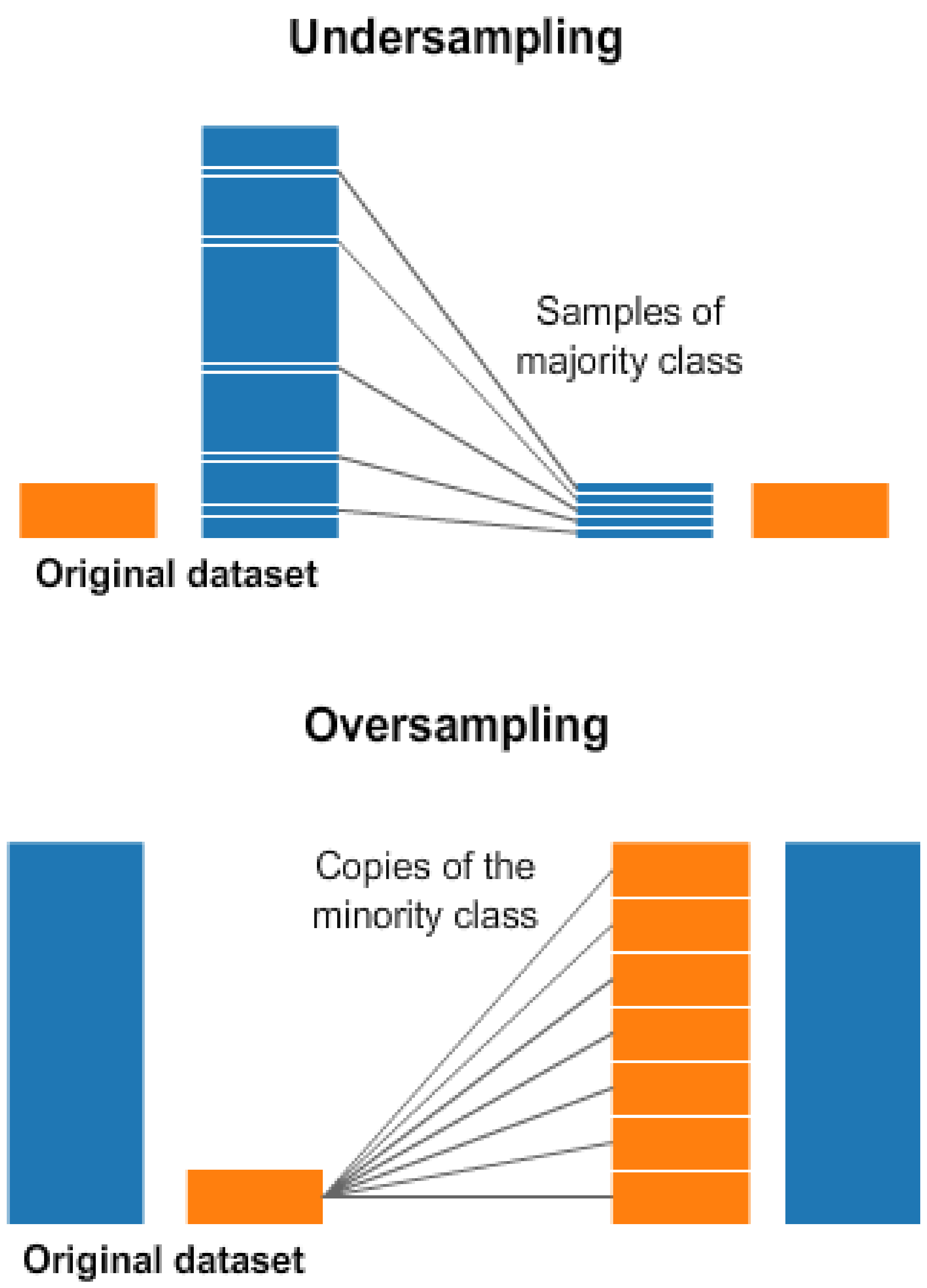
Results improvement

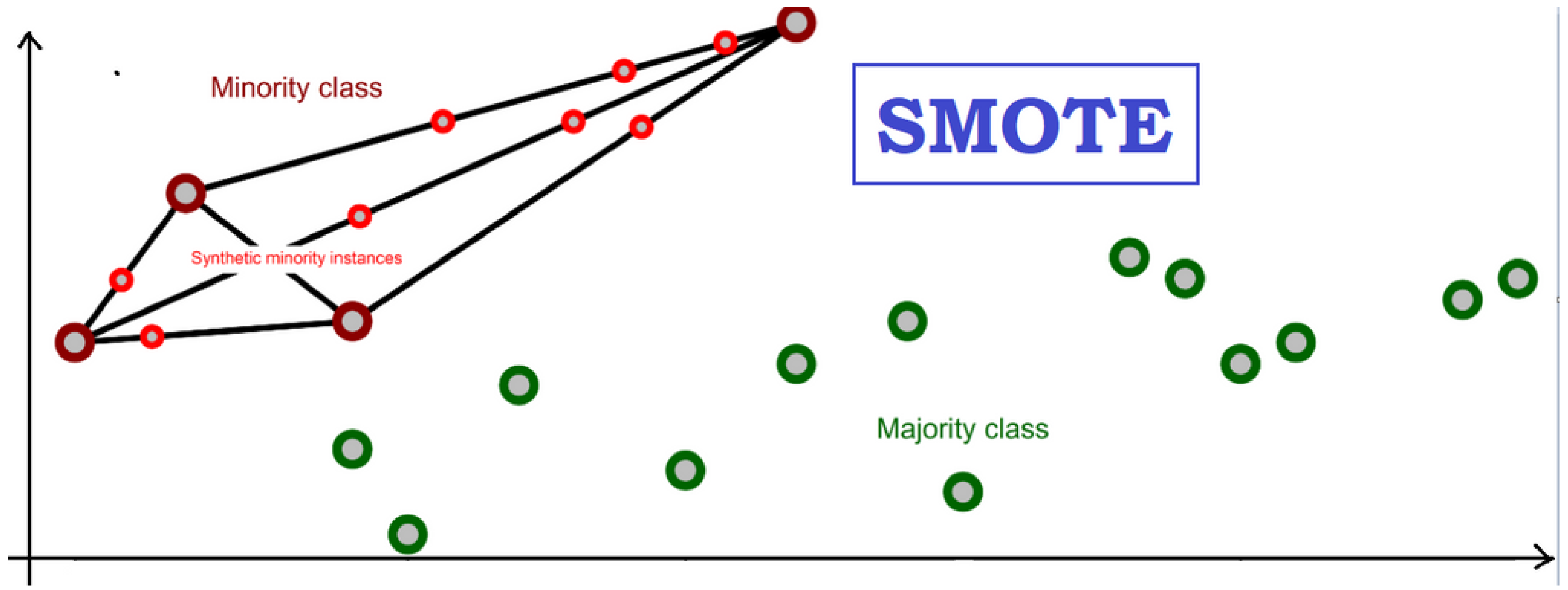
Handling imbalanced class ratio

- When dealing with significantly unbalanced dataset in the target label, it becomes harder for most machine learning algorithms to efficiently learn all classes.
- In this specific case, only **about 23.11%** of the data are **labelled as churn** (label=1).

- There are several ways to address this issue with PySpark
  - Adjust threshold
  - Resampling
  - Weighting

Resampling


Undersampling
Samples of majority class
Original dataset
Oversampling
Copies of the minority class
Original dataset


Minority class
Synthetic minority instances
SMOTE
Majority class

Weighting

- The samples from the **minority class can be weighted more heavily** to balance the contribution of each class in the training process.
- In some cases, reweighting the dataset may provide better results than resampling techniques because it does **not change the distribution of the samples** in the original dataset
- In PySpark, you can reweight the datasets for imbalanced binary classification by adjusting the class weights in the loss function used in the machine learning algorithm
- The **setClassWeight** method is used to set the class weights in each model.

Final results (F1-score positive class)

MODEL	Baseline	Downsampling	Upsampling	SMOTE	Weighting
Logistic Regression	0.71	0.70	0.74	0.76	0.80
Random Forest	0.50	0.56	0.59	0.63	0.53
Linear SVC	0.42	0.69	0.70	0.73	0.74
GBT	0.50	0.48	0.63	0.63	0.59

Main issues

- **Computing capacity**
- **Small amount of train samples**
- **High imbalanced class ratio**
- **Features engineering** and f**eatures selection** were the most crucial phases of the project
- **Extracting features in scalable way.** We aggregated 250k event-level records to 225 user-level records, which is 0.1% size of the raw data
- **Seasonality**: The data we are using only contains two months of data, which means the analysis could be biased by seasonality.

Future works

- Test the models with the **greater version of the dataset (12GB).** Probably RF and GBT would show interesting improvements in their performance due to the higher amount of data at disposal.

- **Better hyperparameters tuning**

- **Better feature extraction and selection**: with higher amount of data at disposal, a greater number of features would be necessary. There are some techniques (for example the *ChiSqSelector* provided by Spark ML) that might help in choose the best features for the model

Thanks for the attention!