

# Customer Churn Prediction

Francesca Gaeta

*Machine Learning Project  
Alma Mater Studiorum  
Digital Transformation Management*

**Abstract** - This study proposes a Machine Learning approach to predict the possible churning customers in a Telecom Company. It embodies a classification problem, since we will try to assign a specific label based on the "Churn" binary feature, addressing the common problem of imbalance in the dataset. The analysis begins with extensive data profiling and exploratory data analysis to uncover behavioral patterns, correlations, and class imbalance, followed by a preprocessing phase involving feature engineering, dimensionality reduction, outlier analysis, and distributional transformations. Several predictive models are implemented and compared, including Logistic Regression, Random Forest, XGBoost, and Artificial Neural Networks, with additional insights gained from unsupervised techniques such as PCA and t-SNE to assess data complexity and separability. Model performance is evaluated using the accuracy score. The results demonstrate that ensemble tree-based models, particularly Random Forest and XGBoost, significantly outperform linear approaches. The study concludes with recommended solutions on how customers might be retained based on the described observations.

## 1 Introduction

With the rapid development of the telecommunication industry, service providers are more inclined towards expansion of the subscriber base. It is stated that the cost of acquiring a new customer is higher than for retaining the existing one. Therefore, it is essential for the Telecom industries to use advanced analytics to understand consumer behavior and, in turn, predict the association of the customers as whether or not they will leave the company. Based on customer level information, the project focuses on analyzing the dependencies of several data points with one another applying statistics and various machine learning algorithms.

The implementation is organized in four main parts, with each addressing a different aspect of the functionality of the project:

1. **Data Import.** The initial stage involves importing an archive called *telecom-churn* from Kaggle. The archive includes only one format csv file with all the data related to the case study. We can import it thanks to Kaggle's API, creating a connection between the notebook and the website via a Kaggle's TOKEN that can be generated in the account's settings and should automatically download locally a kaggle.json file which contains the username and secret token information (it must not be exposed since it is personal for authentication). The common procedure is verifying where this file is saved and eventually move it into the .kaggle folder into the users folder of the local machine (or specifying in the code the different path to find the token). It might happen, as in my case, that the token is not automatically saved, so the secret key can be copied and pasted in a file saved with the .json extension. The file structure is the following:

```
{  
  "username": "username_on_kaggle",  
  "key": "token"  
}
```

To complete this operation, we need to import *os* and *KaggleApi* libraries. All practical applications can be found in the code.

2. **Data Profiling.** This section delves into exploring and analyzing the dataset. In this section, some common simple commands are applied to learn the basics about the dataset, like the number of rows and columns, the semantics of attributes, the data types.
3. **Data Preprocessing.** This stage includes a standard data cleaning step (removing unnecessary/redundant attributes, detecting and replacing outliers).
4. **Machine Learning Modeling.** In this stage of the project, various machine learning models are trained and tested to predict customer churn. This section provides a detailed description of the implementation of the construction of forecasting algorithms with the aid of AutoML techniques and experimentation with ANN (Artificial Neural Networks).

A final section is dedicated to the **evaluation of the predictions** of the developed models on the test set using standard error metrics: *Accuracy*, *Precision*, *Recall* and *F1-score* typical in classification cases, *ROC AUC*, and the *Confusion Matrix*. The model yielding the lowest error values across these indicators is considered the best-performing approach.

## 2 Work with Data

### 2.1 Data Profiling

Data Profiling or Understanding is the initial step where the csv file gets read. By printing information, the number of entries and columns are shown, and it scans all rows to display the number of non-null values associated to their Dtype. The non-null count is the first hint to quickly find out whether there are missing values to deal with. For instance, if values are missing in just a few spots, rows with voids can be dropped if the different groups in the dataset are still represented, or an entire column if the attribute is irrelevant for the final predictive analysis, or just infer the answers in any of the traditional ways (avg, random...). The Dtype is an indicator for understanding changes to apply to data representation (e.g., encoding for nominal values). With the descriptive statistics in hand, some Python libraries are useful for Exploratory Data Analysis (EDA) as a bridge between descriptive statistics and ML workflow, such as *seaborn* and *matplotlib*.

The **visualization of class imbalance** matters because, if the churn group is too small, the ML model might struggle to learn their behavior, 1. In this context, around 500 items is considered enough if hyper-parameters are carefully chosen (e.g., weight via scikit-learn). The risk, otherwise, is that the model always guesses by saying that no one will churn. Some techniques ensure that the model correctly learns, like not using accuracy as primary metric for this dataset, but *Recall* (how many of the actual churners are caught?) or *F1-Score*.

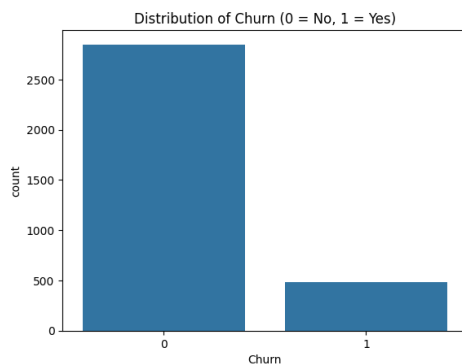


Figure 1: Histogram of Churn Class Imbalance

With a **feature correlation heatmap**, which is a visual representation of a correlation matrix, the most connected attributes to the churn are discovered without manual checks refheat. Both the horizontal (x) and vertical (y) axes list every attribute in the dataset. Each square (cell) shows a number between -1.0 and 1.0 called the Correlation Coefficient (by setting *annot = False* the coefficients are not shown because they are not clearly readable due to the reduced dimension of the cells). The diagonal (value 1.00) running in the middle is trivial because every variable is perfectly correlated with itself (e.g., DayMins vs DayMins).

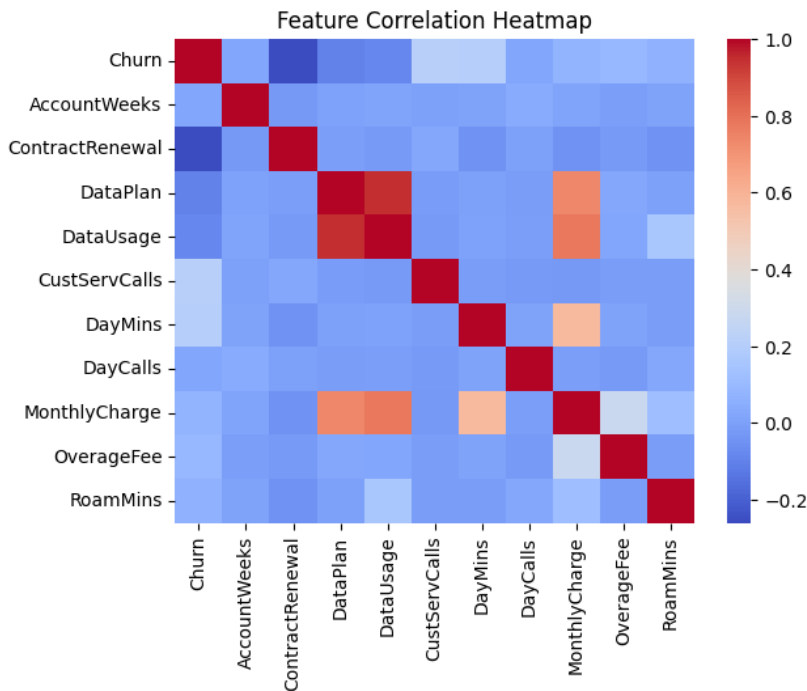


Figure 2: Feature Correlation Heatmap

Of course, there are some banal insights, such as a positive and strong correlation between DataUsage and DataPlan or DataUsage and MonthlyCharge, but a combination of data filtering and correlation analysis can find some deeper insights to verify important correlations with the churn tendency. Being near +1.0 (Dark Red) highlights a strong positive correlation (when one goes up, the other goes up, ex. DayMins and MonthlyCharge usually have a high positive number because more minutes used correspond to higher bills). The contrary happens with a strong negative correlation close to -1.0 (Dark Blue). Light or White color is used when variables seem not to affect each other (value 0).

The existence of a linkage is not related only to the red color, but also the inverse relationship can be interesting: if **ContractRenewal** has a correlation of -0.26 with Churn, it means the *absence* of a renewal (0) is strongly linked to the presence of churn (1).

**CustServCalls** was believed to be one of the main drivers for churning, but the map reveals there are some more important considerations. Looking at the numbers, anyways, a correlation around 0.20 might seem a small number in absolute terms, but it is an interesting finding for this type of data, also because it is the highest we can find in relation to churn in the matrix together with DayMins. Correlation measures the linear strength of the relationship across the entire dataset, but the averages tell the story of the groups. The value 1.45 is the mean of the CustServCalls column, but only for the rows where the customer did not churn; 2.23, instead, underlines a 54% increase in the call volume for the churn group:  $x = ((2.23 - 1.45) / 1.45) * 100$ .

**The Heavy User Paradox:** In many industries, unhappy customers are expected to be those who don't use the service much, but this dataset shows the opposite. By looking at the Avg DayMins, there is a clear difference between loyal and churned clients. Churners talk about 18% more during the day than loyal customers, so they are also being billed more (Using the percentage formula  $(207 - 175) / 175$ ). The Usage Intensity insight reveals a specific business problem: the most active users that therefore generate the most revenue are the ones leaving. If someone uses a lot of minutes and sees high monthly charges plus overage fees, is more prone to look for better alternatives by competitors. Usually, high usage is seen as engagement, but in this dataset it is actually a warning sign that the customer might find a better deal elsewhere.

## 2.2 Data Preprocessing

### 2.2.1 Data Cleaning

Things are simpler in this dataset thanks to the lack of missing values and duplicates. Nominal data are not to convert into numbers, yet are directly displayed with binary numerical values 0 and 1. Anyways, high correlation guides feature engineering retaining the same level of information with less columns (dimensionality reduction).

**The Redundancy in DataPlan and DataUsage.** These two attributes have a correlation of 0.946. It means they are effectively providing the same information to the model. In almost every case, if a customer has a DataPlan (1), they have recorded DataUsage. We can perform **Dimensionality Reduction** by keeping only DataUsage, a continuous variable (0.0) which gives the model more "granular" information than the simple binary 0/1 of the DataPlan. The adopted algorithms (tree-based) are able to autonomously split it into the best intervals, but their job can be simplified by performing logarithmic transformation.

**Case of High Correlation.** MonthlyCharge has a strong correlation with DayMins (0.57) and DataUsage (0.78). The more minutes and data a customer uses, the higher their bill. While 0.78 is high, it is usually below the "drop" threshold (typically 0.80 or 0.85). Instead of dropping them, we could create a New Feature (**Feature Engineering**) called **PricePerMinute** by dividing MonthlyCharge by DayMins.<sup>1</sup> This *merges* the two into a single attribute that represents "Value for Money," which is often a better predictor of churn than the raw numbers alone. We then dropped MonthlyCharge since its information is embedded in other features and it is less relevant.

**Random Forest in Preprocessing.** A quick RandomForest can be run during preprocessing to see the Feature Importance chart to guide attribute selection. Some parameters need to be set first to make it effective, like `class_weight='balance'` because only 14.5% of the customers churned and

---

<sup>1</sup>In my first calculation, we added a small constant  $+(1e-9)$  to the denominator to handle two observations with zero data usage, which would otherwise have resulted in divisions by zero, but the distortion survived and it became evident in the exploratory analysis with skews. Rather than adding an arbitrary small constant to the denominator, we now use the `np.where` method to explicitly handle these cases by removing (or excluding) the outliers in PricePerMinute that arise from divisions by zero or near-zero values.

the model should not ignore this behavior; the `n_estimators=100` as standard tradeoff between stable importance score and time spent; the `random_state=42` to ensure that every time we run the code we get the same importance score not to randomly change conclusions.

In machine learning, a common rule of thumb for feature selection is to keep attributes that contribute at least 5% (0.05) to the model's decision-making. Even though `RoamMins`, `DayCalls`, `AccountWeeks`, and `DataUsage` aren't "stars", they help the model differentiate between similar customers. For example, two people with high `DayMins` might be distinguished by their `AccountWeeks` (loyalty) or `DayCalls` (frequency). Even though `DataUsage` is the lowest, it represents the only "data-related" behavior left in the model after removing `DataPlan`. Dropping it would mean the model loses all visibility into whether the customer uses mobile data or not.

Running statistics on `DataUsage` checks whether people who don't use data are more prone to churn. This is confirmed by a rate of almost 18% versus 10%, which is still a significant number.

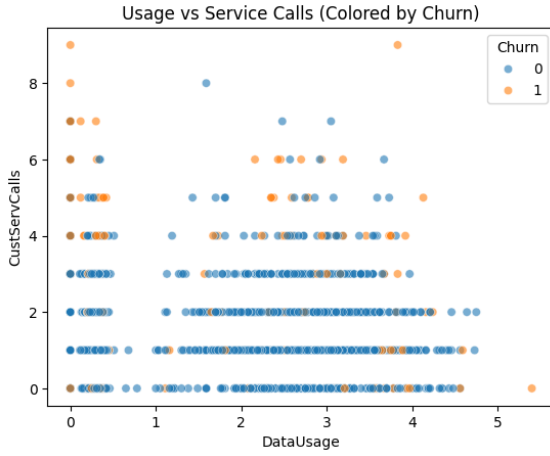


Figure 3: Scatterplot `DataUsage` vs `CustServCalls`

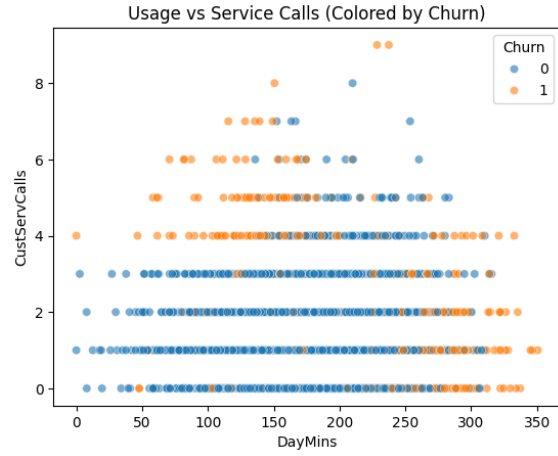


Figure 4: Scatterplot `DayMins` vs `CustServCalls`

The scatterplots show if churners gather in specific "zones" 3, 4. Each dot represents a customer. The location tells two things about minutes of call and number of calls to the customer service. If the two colors are mixed, the features together can't predict the churn very well. The "Danger Zone" is where one color dominates. Data usage is less impacting. The real discriminating factor in this comparison is still the calls to the customer service, since the bottom side of the graph is predominantly blue (no churn independently from the amount of data utilized). This was also somehow confirmed by the importance matrix.

We look for the same statistics in `DayMins`, where there are two clients not using the service. Since there are only 2 users with exactly zero minutes, a "percentage" can be misleading (50%), so it's better to show the raw data for them to prove the point. Out of the 2 users, one churned (50% churn rate), so it's highly volatile. While the sample size is tiny, it's still significantly higher than the average population churn rate of 14.47%. The comparison statistically does not make sense since we have a small sample size, but logically it can make sense as a diagnostic flag to show that the extreme data points are unstable and should be kept under control since not utilizing minutes can push towards the churn.

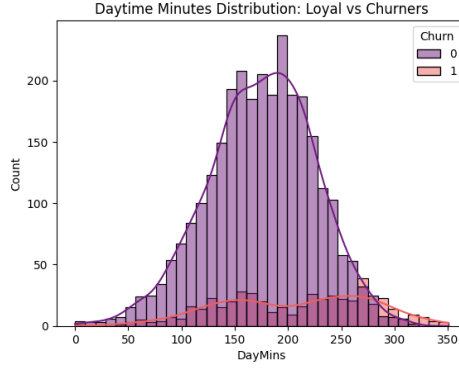


Figure 5: DayMins Distribution loyals vs churners

This Figure 5 shows two overlapping distributions, one for those who stayed and one for those who left (x-axis the value, y-axis how many customers fall into the range). Colors represent loyalty (y/n). Where colors overlap, is an uncertainty zone: people behave similarly whether they stay or leave. The right skew is the only part where the number of churning customers overcome non-churning. This proves that churners generally talk more than loyals.

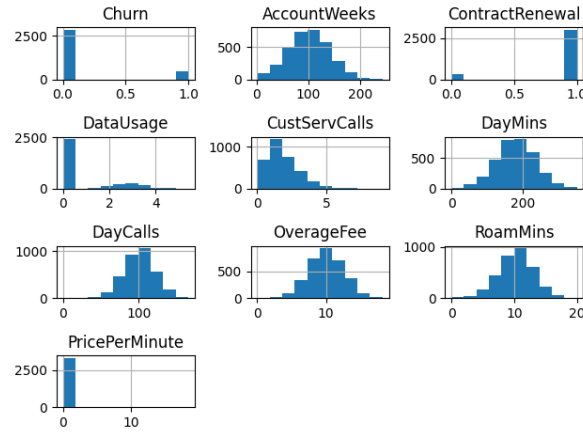


Figure 6: Histograms for data distribution

The histogram is one of the most powerful tools to understand density and distribution of the dataset quickly. Most of the features like *DayMins*, *DayCalls*, *OverageFee* and *RoamMins* look like Bell Curves (Normal Distribution). This tells that customers are consistent. Most people are in the middle, and extreme behavior is rare. Features like *DataUsage* and *CustServCalls* are right-skewed. The long "tail" to the right represents the problematic customers who are consuming a lot of support resources and are likely to churn. For binary columns, the histogram will look like two separate "towers" at 0 and 1. The result is that the imbalance is directly visible. To spot more particular behaviors, we can observe that *AccountWeeks* has a spike which might indicate a successful marketing campaign from about 100 weeks ago. In *DataUsage* we might see a huge bar at exactly 0. This visually proves that a large portion of customers simply does not use mobile data at all. In *PricePerMinute*, the X-axis from 0 to 10 proves that we successfully removed the extreme mathematical errors discussed before, bringing the data to a human scale 7. Even though the scale is better, the data is still highly concentrated. To see the actual distribution, we need to zoom in 8.

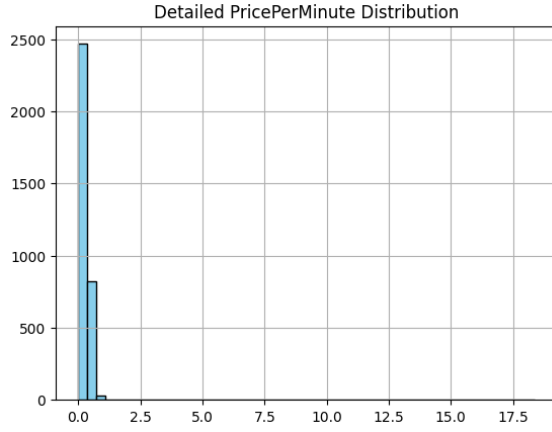


Figure 7: PricePerMinute Hidden Distribution

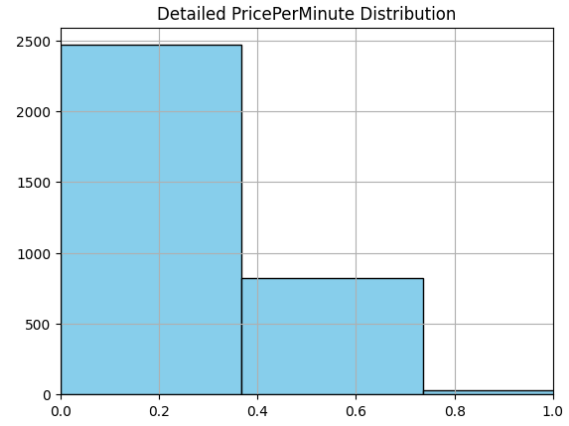


Figure 8: PricePerMinute Zoom In 0-1 range

75% of customers pay less than 0.37 euros per minute. We have the few heavy users that pay 18.38 euros per minute. By default, a histogram might split the 0-to-18 range into 10 "buckets." This means the first bucket covers everything from 0 to 1.80 euros. Since almost all 3333 customers pay less than 1.80, they all get piled into that very first bar, making it look like one giant block at the start.

## 2.2.2 Outlier Detection

Outlier Detection should not just find the numbers that are far away, but help decide which ones are errors and which are key behavioral signals. In telecom churn, outliers are often the most important data point (for instance, a customer calling support 9 times is an outlier, but they are also the most likely to churn). The graphical representation immediately spots which columns have extreme values.

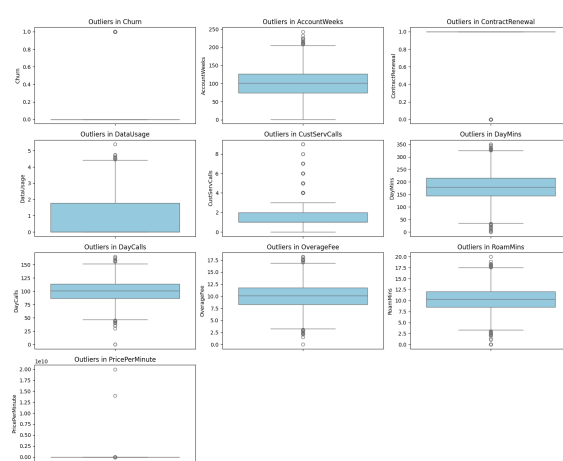


Figure 9: BoxPlot for outlier detection

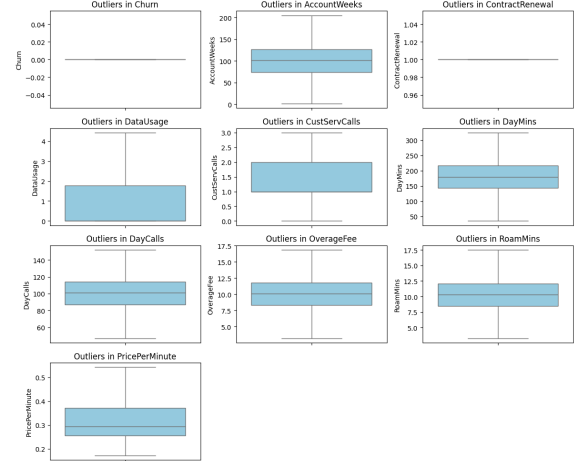


Figure 10: Box without fliers

It is normal not to see the box in certain graphs, as in the PricePerMinute feature for the mathematical reasons expressed above. The quick fix to visualize the box is setting the `showfliers` parameter to False (it will hide the outliers) so that we can acquire a physical idea of the distribution [10](#).

Afterwards, instead of checking outliers one by one, we create a summary table that shows exactly

how many of them exist in each category and what the "normal" boundaries are with the IQR method ((**Interquartile Range**), a statistical technique used to measure the spread of the middle 50% of the data, so the median. The IQR method is more robust than simple means, which can be ruined by single crazy numbers. To decide if a value is an outlier, the method creates "fences" or whiskers. 9 Anything outside these fences is officially an outlier. For the lower and upper bound we multiply times 1.5 because statisticians found it to be the perfect "sweet point" for normal distribution. For instance, from our data we understand that any customer who makes 4 or more calls is mathematically an outlier in the dataset. The only reason to remove this data is to let tree-based algorithms perform better, even though they do not suffer from anomalies since they are capable of finding the right thresholds to split and find the general rules, but keeping the values more compact help enhancing accuracy anyways.

Instead, we want to verify whether those high prices are really caused by Data Usage, not a mistake in the dataset, and that some people really use that many minutes. First of all, we reload the original dataset since we need again the MonthlyCharge for this calculation. We can't directly proceed with PricePerMinute because we need to go back up to the history of data to see the bills that created the high prices. We use the same 0.54 limit we found with the IQR method to find every customer paying an "outlier" price to create a smaller list of just the 173 "suspicious" rows and we can inspect them closely. If a customer has a high bill but 0 minutes, it seems an error. But if we notice they used 2.5 GB of data, we suddenly understand why the bill is high. This step proves the data is correct, not an error. In the same way, we can check whether who calls a lot has more churning probability considering the upper bound value. The rational is that if these people were errors, their churn status would be random; instead, it is stated that almost all of them have Churn = 1. This proves that these outliers are the most valuable data and by deleting them we would remove the target people the model will try to identify.

We should ignore the outlier report for Churn and ContractRenewal because they are binary (representing categorical attributes behind ordinal numbers) and this is not a problem per se, but the IQR works with continuous values, so it won't identify the real lower and upper bounds. The overall outcome of our outlier detection is that we don't need to get rid of anything for our analysis.

**Isolation Forest** detects "anomalies" by isolating them in a tree structure. The strength is that it finds combinations of features that are rare, even if individual values aren't extreme. This is the reason why we talk about *multivariate anomalies*. If an outlier has 9 service calls but doesn't churn, it might confuse the Random Forest model into thinking service calls aren't a strong predictor. This is not our case because the classifiers have already identified it as the strongest feature for our analysis, so these kinds of outliers don't impact that much on the final result. Thanks to the comparison among the means for outliers (167 customers), inliers (3166 customers), and the means taking into account the whole dataset, we can observe as expected that the second and the latter means don't diverge much, while for outliers we have some remarkable insights:

- The most significant finding is that 34% of these outliers churned. This is nearly 2.5 times higher than the normal group. This indicates that being "unusual" is a massive predictor of leaving.
- These outliers call customer service roughly 3 times more often than normal customers (2.21 vs 0.74). Since high service calls correlate with dissatisfaction, this group likely contains troubled accounts.
- The normal group pays 0.32 per minute, but outliers pay 0.67. This suggests they might be on the wrong plan for their usage patterns.
- They use significantly less data than the normal group, further suggesting they are under-utilizing the service relative to what they might be paying.

Since these representations place the accent on some highly skewed data, we can try to apply **Logarithmic Transformation** to DataUsage and CustServCalls to acquire a Gaussian-like distribution that would be helpful during the training with some algorithms 11. We can also calculate the mathematical proof of improvement, that enhances if the skewness is closer to zero.

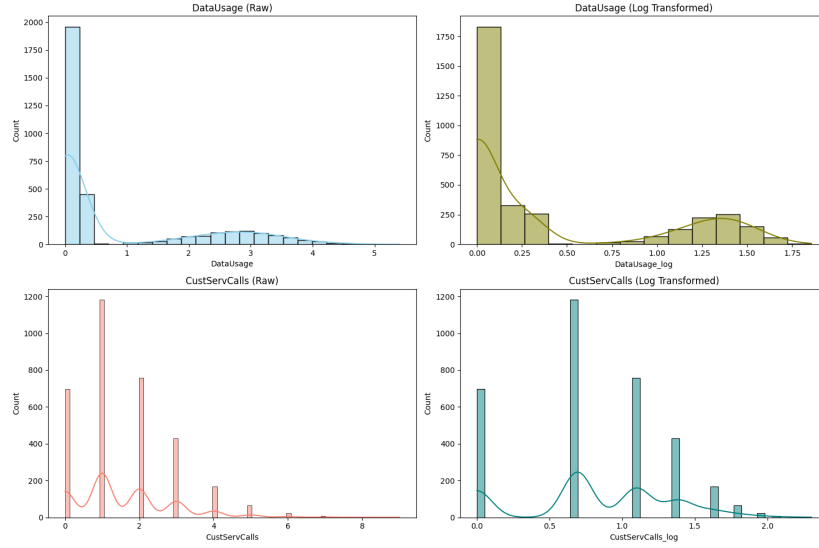


Figure 11: Comparison of skewness before and after log transformation

From the compared graphs, we can state that log transformation works with CustServCalls, which almost reaches symmetry, but is not worth it for DataUsage, and there is a clear statistical reason why: DataUsage is zero-inflated because more than 50% of the dataset does not have a plan. The math of the transformation is  $\log(1 + x)$ , hence when  $x = 0$ , the result is  $\log(1) = 0$ . Regardless of the transformation applied, we can't move massive data of the same value, so it is basically impossible to get to a bell in this case. As a matter of fact, this analysis was useful to evaluate the creation of binary feature for DataUsage (0 if no usage, 1 if there is usage).

### 3 Machine Learning Modeling

Before trying out various algorithms, we run a PCA (linear, 14) and t-SNE (non-linear, 15) to understand the complexity of making predictions in the dataset. In the **cumulative variance**, with only PC1 (related to Overall Volume) and PC2 (Plan Type) we overcome the 90% of information retention. Conceptually, cumulative variance gives indications of how many features we can "throw away" without losing the essence of data. The **elbow rule** says that when the line starts to flatten out in the Scree Plot, the sweet point is reached.

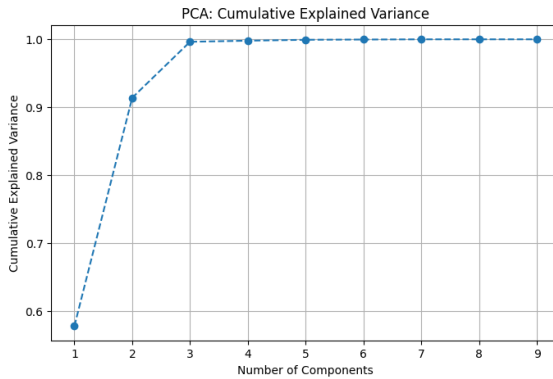


Figure 12: PCA cumulative variance

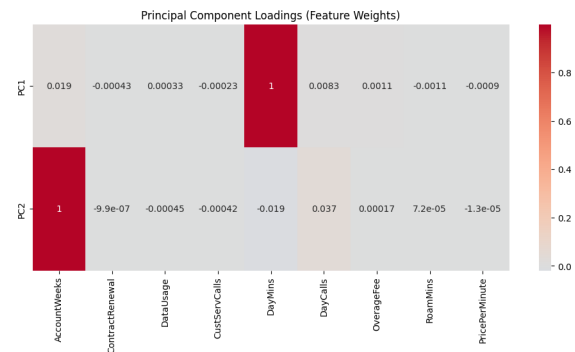


Figure 13: PCA loadings - feature contribution

Oppositely, t-SNE is non-linear and often better at revealing clusters in complex datasets. The t-SNE is harder to interpret: if it shows distinct clusters that PCA did not, it underlines that the relationship between features is non-linear. We can calculate a **silhouette score** for t-SNE which only corresponds to 0.071. From our analysis it emerges from both metrics that our data will be difficult to classify, and non-linear models like XGBoost or Random Forest are suitable to find the "hidden" boundaries.

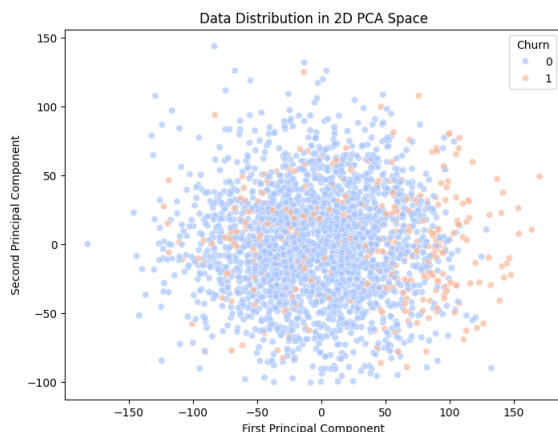


Figure 14: PCA distribution

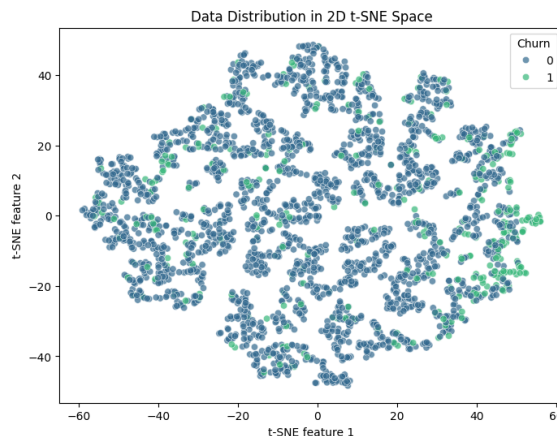


Figure 15: t-SNE distribution

Once the dataset was prepared, it was split into training and test sets. We took as reference both the original df, which is the one containing outliers, and the cleaned dataset after outlier detection and deletion. Even though anomalies don't make Decision Trees or Random Forest struggle, they are still "greedy", meaning they will keep splitting until they can perfectly classify every point in the training set if allowed, creating specific leaf nodes just for one extreme value. It means that, with outliers, the trees might create a specific leaf node just for one or two extreme outliers, causing *overfitting*. By removing those 5%, we remove the most confusing rows, allowing accuracy to slightly increase.

**Logistic Regression** tries to draw a straight line to separate churners. **Random Forest** draws complex (but orthogonal) boundaries. Since the Forest is much more accurate, we can conclude that churn is caused by combinations of factors, which Logistic Regression struggles to capture. This latter achieved 77% accuracy, which is below the 85.5% baseline accuracy. This indicates that the drivers of churn in this dataset are non-linear and complex. Because the dataset is highly imbalanced, achieving approximately 93% accuracy represents a meaningful improvement over simply predicting the majority (non-churning) class. Both the Cleaned Random Forest and XGBoost models reached this level of accuracy, delivering a 7.5% improvement over the baseline, which indicates that they are successfully identifying actual churners rather than assuming all customers remain loyal. By modeling complex, non-linear interactions and feature intersections, these tree-based methods effectively navigate the class imbalance and isolate the behavioral patterns associated with churn without being dominated by the majority class.

The implementation of the **AutoML** technique with FLAML at the beginning exploited 60s time-budget to try out the list of suggested algorithms and it confirmed that the most promising is **XG-Boost**.

I also experimented with **Artificial Neural Networks** using the open-source *tensorflow* library. It might be caused by the fact that the dataset is not huge and other algorithms already carry out effective forecasting, but the difference with all the other models is not striking.

## 4 Error Metrics

Machine learning models were trained on the training portion and subsequently used to generate forecasts on the test set. This approach enabled the evaluation of predictive accuracy and the identification of the best-performing model. We used the **classification\_report** method to evaluate the model accounting for accuracy, precision, recall, and F1-score, essential for imbalanced data. The accuracy alone underlines the percentage of total correct guesses. In this dataset, where around 85% of people do not churn, a model that simply guesses "No Churn" for everyone would obtain 85% accuracy, but would be completely useless at finding actual churners. **Precision** is to understand how many times the model is right when it predicts that someone will churn, **recall** to understand out of all the people that actually churned how many could the model catch, and the **F1-score** is an harmonic mean of precision and recall (if we look at the F1 for Class 1 - churners -, we can determine the best model). The **support** is the number of actual occurrences available in the test set.

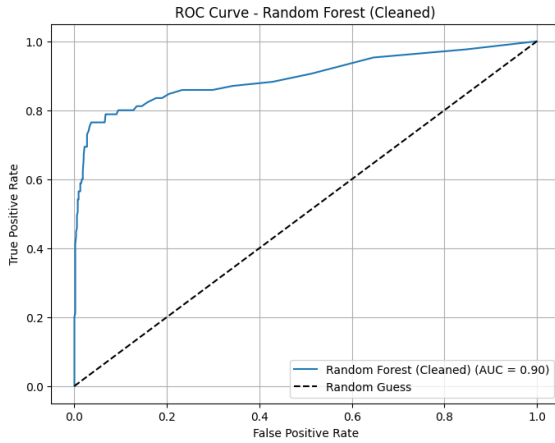


Figure 16: ROC AUC score - Random Forest

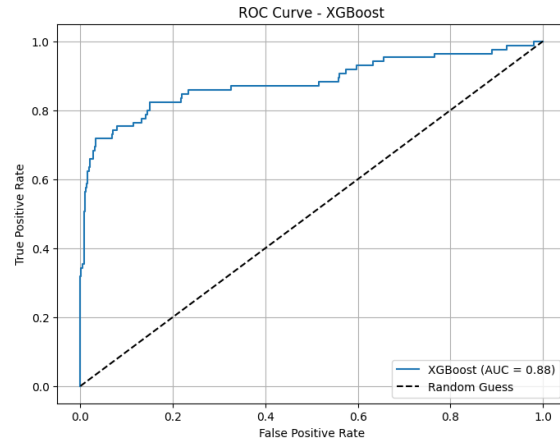


Figure 17: ROC AUC score - XGBoost

Comparing the best models - Random Forest and XGBoost - through the **ROC AUC score** (16, 17) is actually quite interesting because they show a classic trade-off in machine learning. While the Random Forest has a higher "overall" quality score (AUC), the XGBoost model is actually more useful for the specific business goal of catching churners (value [1] which TP in the ROC is associated to). The Random Forest has a higher ROC-AUC. This means that, mathematically, it is slightly better at distinguishing between a "churner" and a "non-churner" across all possible confidence levels. If we just looked at this number, we would pick Random Forest. However, in churn prediction, Recall is critical because it tells of all the people who actually churned, how many the model caught. Even though its AUC is slightly lower, XGBoost caught 10% more churners than Random Forest. In a business context, missing a customer who is about to leave is usually much more expensive than accidentally sending a discount email to someone who was going to stay.

Table 1: Classification report Logistic Regression

0	0.96	0.77	0.86	549
1	0.35	0.78	0.48	85
Accuracy			0.7744	634
Macro avg	0.65	0.78	0.67	634
Weighted avg	0.88	0.77	0.81	634

Table 2: Classification report XGBoost

0	0.96	0.96	0.96	549
1	0.74	0.72	0.73	85
Accuracy			0.9290	634
Macro avg	0.85	0.84	0.84	634
Weighted avg	0.93	0.93	0.93	634

<sup>2</sup>Tables chosen for representation purposes (the most vs less accurate algorithms).

Class	Precision	Recall	F1-score	Support
0	0.93	0.98	0.95	570
1	0.83	0.54	0.65	97
Accuracy			0.9160	667
Macro avg	0.88	0.76	0.80	667
Weighted avg	0.91	0.92	0.91	667

Table 3: Classification report - Random Forest with outliers

The **confusion matrix** for each algorithm can help us visualize faster what happened with our predictions. The matrix breaks down exactly where the model is failing. The goal is choosing the model that best satisfies our needs: for instance, we know that generally precision decreases when recall goes up and vice versa, but if in our business analysis we prefer to classify loyal customers as quitting because we need to be extremely conservative and consider even the worst of the cases, we may prefer the decision tree even though the random forest overall performs slightly better. It is a matter of personal considerations.

## 5 Conclusions

From a segmentation perspective adopting **K-Means**, both the Elbow Method and Silhouette Analysis initially suggested an optimal clustering solution at  $K=3$ . However, while the Silhouette Score peaked at this value, a deeper qualitative evaluation revealed that increasing granularity to  $K=4$  produced substantially higher business value. The drop in the Silhouette Score at  $K=4$  reflects a mathematical penalty for splitting a dense cluster, but this trade-off proved strategically justified. Moving from three to four clusters revealed that one subgroup, labeled *Quiet Loyalists (Persona 1)*, exhibits long tenure, balanced usage, and a low churn rate despite relatively higher prices per minute, indicating strong perceived value and brand loyalty. The other subgroup, *Price-Sensitive Light Users (Persona 3)*, shows shorter tenure, very low prices, and significantly higher churn risk, suggesting a trial-phase relationship with the provider. This distinction is critical: while these two groups are mathematically close in feature space, they require fundamentally different retention strategies. Another cluster of *Chronic Churners (Persona 0)* is characterized by heavy daytime voice usage, frequent overage fees, and extremely high churn rates, but they are among the most active users. This confirms the “heavy-user paradox”. Conversely, *Premium Data Users (Persona 2)* emerge as the most stable and profitable segment, exhibiting high data consumption, tolerance for higher prices, and minimal churn.

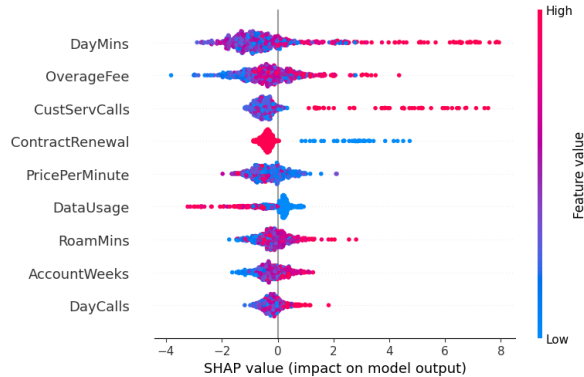


Figure 18: SHAP values for test set

The integration of SHAP (18) with the XGBoost model grasps why individual customers are predicted to churn. For example, while low price per minute appears correlated with churn at the segment level, SHAP analysis shows that churn risk in price-sensitive clusters is often triggered by excessive customer service interactions, indicating frustration rather than pure cost concerns. Similarly,

high daytime minutes consistently push predictions toward churn, whereas high data usage exerts a stabilizing effect. Our business strategy, in priority order, needs to focus on:

- **Persona 0:** we should stop bill shock before the customer leaves, and it can be practically done by showing personalized offers before the next bill cycle, or outbound calls from a retention specialist and not a generic support. Since these customers have high DayMins, we can offer voice add-ons instead of full plan changes (e.g., unlimited voice booster for just +5 euros/month). Customers already behave as unlimited users, and formalizing it removes friction. Overage Forgiveness in such cases is highly effective via models that flag user before overage hits sending an SMS (e.g., *"You're approaching your limit. Upgrade now to avoid extra charges."*).
- **Persona 3:** this group churns due to confusion and switching ease, not dissatisfaction, so we can arrange entry-level bundles to psychologically anchor to simple tariffs and no surprise overage. Telecoms rarely push hard contracts anymore, but they use instead contract incentives like a discount after three months, bonus for data if the contract is renewed. Switching to on-boarding techniques, top telecoms actually offer a "golden window" during the first 30-60 days during which they explain the plan usage to avoid bill shock and suggest optimization tips. At the end of the period, they advise the plan that fits the type of user the best. It works because most early churn happens because customers don't understand the plan.
- **Persona 2:** these users don't churn, but they upgrade with premium data plans or add-on services like cloud storage, early 5G access.
- **Persona 1:** we should not break what works, at most we retouch retention lightly through anniversary rewards, priority support or small loyalty perks. We don't discount unnecessarily if they are already loyal.