Predicting Customer Churn Using Machine Learning

PABLO FUENTES CHEMES, ANDRÉS NÓ GÓMEZ, and PABLO PÁRAMO TELLE, University of A Coruña, Spain

ABSTRACT:

Customer churn is a critical challenge for telecommunication companies, significantly impacting profitability and growth. This project aims to develop a machine learning model to predict customer churn, enabling the company to implement data-driven retention strategies. By prioritizing the retention of existing customers, an approach that is often more cost-effective than acquiring new ones, the model seeks to help the company effectively minimize the churn rate while enhancing profitability.

Additional Key Words and Phrases: Telecommunications, Churn Prediction, Machine Learning,

Authors' Contact Information: Pablo Fuentes Chemes, p.fuentes@udc.es; Andrés Nó Gómez, andres.no.gomez@udc.es; Pablo Páramo Telle, pablo.paramo.telle@udc.es, University of A Coruña, A Coruña, Spain.

55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108

Contents

Abstract 1
Contents 2
1 Introduction 3
1.1 Problem 3
1.2 Objective 3
1.3 Database description 3
1.4 Approaches 4
1.5 Chosen Metrics 5
1.6 Code Structure 6
1.7 Bibliographic analysis 7
2 First Approach 8
2.1 Description of the database 8
2.2 Data preprocessing 8
2.3 Results 9
2.4 Discussion 13
3 Second Approach 14
3.1 Description of the database 14
3.2 Data preprocessing 14
3.3 Results 15
3.4 Discussion 18
4 Third Approach 19
4.1 Description of the database 19
4.2 Data preprocessing 20
4.3 Combining PCA and Cross Validation 20
4.4 Results 21
4.5 Discussion 23
5 Fourth Approach 24
5.1 Description of the database 24
5.2 Data preprocessing 24
5.3 Feature selection 25
5.4 Results 28
5.5 Discussion 31
6 Final Discussion 32
6.1 Evaluation of model performances 32
6.2 Tracking of results obtained across the approaches 33
6.3 Final Model Selection 34
7 Final Model Training 34
8 Conclusion 34
9 Future Work 35
References 36

1 Introduction

1.1 Problem

In the telecommunications industry, customer churn poses a significant threat to companies, as it directly impacts both revenue and long-term growth. JB Link, a small telecommunications provider based in California, has experienced rapid expansion over the past six years, increasing its infrastructure in underserved regions and utilizing a highly skilled sales team. Despite this success, the company now faces a critical issue: a high customer churn rate.

1.2 Objective

By identifying at-risk customers early, the company can implement targeted interventions, potentially reducing churn. This approach is more cost-effective than a strategy purely based on engaging, as acquiring new customers can be up to five times more expensive than retaining existing ones.

The goal of this project is to predict which customers are most likely to churn using machine learning algorithms, implemented through various techniques such as Artificial Neural Networks (ANNs), Support Vector Machines (SVM), Decision Trees, k-Nearest Neighbors (kNN) and ensembles of the models previously mentioned. For each model, several different architectures and hyperparameters will be evaluated.

The models will be evaluated using cross-validation on four different approaches to the same dataset. This will allow not only to find the most optimal model, but also to explore how different ways to tackle the problem affect the obtained results. The approaches have been designed to explore the impact of different data preprocessing methods and dimensionality reduction techniques.

1.3 Database description

The dataset we decided to work with is a variation of the dataset IBM Telco Customer Churn, with additional columns to provide a more realistic composition. This dataset is JB Link Telco Customer Churn, found in the Kaggle website, whose base composition consists of 46 columns structured as follows: 17 Boolean, 20 Numerical and 9 String. Below is a brief description of these features grouped by categories:

Customer Information. These features describe the customer's demographic and personal attributes, including some general ones such as: Customer ID, Referred a Friend, Number of Referrals, Gender, Age, Married, Under 30, Senior Citizen, Dependents, Number of Dependents.

Location Data. These variables relate to the customer's location and include: City, Zip Code, Latitude, Longitude, Population.

Contracted Services Data. This category includes the services the customer has with the company: Phone Service, Multiple Lines, Internet Service, Internet Type, Streaming TV, Streaming Movies, Streaming Music, Online Security, Online Backup, Device Protection Plan, Premium Tech Support, Unlimited Data.

Usage and Billing Data. These variables reflect the customer's behavior and financial relationship with the company: Tenure in Months, Monthly Charge, Total Regular Charges, Avg Monthly Long Distance Charges, Total Long Distance Charges, Avg Monthly GB Download, Total Refunds, Total Extra Data Charges.

Preferences and Contracts. This section details the payment and contract options selected by the customer: Offer, Contract, Paperless Billing, Payment Method.

Customer Service Interactions. These variables measure the customer's relationship with customer service and perceived quality: Total Customer Svc Requests, Product, Service Issues Reported, Customer Satisfaction.

Key Business Metrics. This section includes essential information for analyzing the customer's relationship with the company: Churn Value, CLTV (Customer Lifetime Value).

The following variables were removed from the dataset: Customer ID, Churn Category, Churn Reason and Customer Satisfaction. The reasons for each case are as follows:

- **Customer ID**: This variable simply identified the user and was not useful for analysis beyond removing duplicates during preprocessing.
- **Churn Category** and **Churn Reason**: As mentioned earlier, our work focuses on predicting whether a customer will leave the company based on different features to anticipate and prevent it. These two categories were deemed irrelevant to our objective since they reflect cases where customers had already left, which is captured by the "Churn Value" parameter.
- **Customer Satisfaction**: During preprocessing, we observed over 5,000 missing values for this feature. Given that more than half of the users had not provided responses, it was not worth including it in the analysis.

1.4 Approaches

The project has been organized into four different approaches on the same dataset. The aim is to illustrate how different interpretations, preprocessing techniques and strategies affect the results obtained by machine learning models.

These approaches follow an intuitive sequence that facilitates tracking the evolution of the results. In the first two approaches, we aim to demonstrate the impact of using advanced preprocessing techniques, particularly on categorical variables. In the latter two approaches, after identifying the most effective preprocessing method, we will focus on examining whether the results achieved using all available features are robust to dimensionality reduction, as computational efficiency is critical to the models' viability. A comparison between the two dimensionality reduction techniques performed will also be carried out.

1.4.1 First approach:

In this initial tackle to the problem, we will use basic preprocessing techniques to evaluate the performance of the models with data in its rawest form. However, some variables will not be used as they require more advanced transformations beforehand. All available features in the dataset that are continuous (but non-cyclical) or binary will be included. Numerical features will be normalized or standardized to prevent scale issues. For categorical variables, only those with a limited number of categories will be considered, mapping each class to predefined specific values. Variables with a large number of categories require more complex preprocessing, which will be addressed in the second approach.

1.4.2 Second approach:

In this approach, more advanced preprocessing techniques will be used to broaden the range of information provided to the models. Multi-class categorical variables will be assigned numerical values based on the churn proportion of each class. Additionally, cyclical features will be transformed using trigonometric functions. The models will then be evaluated using these updated features to determine their effectiveness in improving performance and capturing patterns in the data.

1.4.3 Third approach:

Principal Component Analysis (PCA) will be performed for dimensionality reduction purposes. PCA will be applied to the dataset version from the approach that delivers the best results between the first two analyses.

1.4.4 Fourth approach:

In the final approach, we will apply another dimensionality reduction technique to the version of the dataset that achieved the best results among the first two approaches. A correlation analysis with the target variable will be conducted to identify the features that are more linearlly associated with customer churn. After extracting the most significant features, a second correlation analysis will be performed to assess multicollinearity among them. Finally, only the most relevant features will be used for model training and evaluation. This approach enables dimensionality reduction that, unlike PCA, remains easily interpretable.

1.5 Chosen Metrics

When choosing the metrics to evaluate the performance of the models, we must consider the characteristics of our problem and stablish our goals.

In this study, churn cases are considered positive, while non-churn cases are negative. The model's outputs can be classified into four categories, forming the confusion matrix shown in Figure 1.

- True Negatives: Negative values predicted as negative.
- False Positives: Negative values predicted as positive.
- False Negatives: Positive values predicted as negative.
- True Positives: Positive values predicted as positive.

		Prediction	
		Negative	Positive
Real	Negative	TN	FP
	Positive	FN	TP

Fig. 1. Standard Confusion Matrix

When predicting customer churn, it is critical for the company to effectively identify high-risk customers to implement retention measures on time, but without overestimating churn risk to avoid offering advantages and deals to customers who were not actually planning to leave.

For this reason, the primary evaluation metric for the models, which will guide our decision-making, will be accuracy. It can be interpreted as the ratio of customers, regardless of whether they actually leave the company or not, for which the prediction of a model is correct. It is calculated as follows:

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP}$$

Although we have discussed that we aim for equilibrated model that has no biases towards positive or negative predictions, in the context of the problem at hand, the strategies aimed at customer retention are more cost-effective to those focused on acquiring new customers. For this reason, we will introduce a second metric, sensitivity, to help us reflect this preference.

Our inclination towards a strategy that prioritizes retention can be interpreted as a preference for a false positive (churn is predicted, but the customer stays) over a false negative (the customer is predicted to stay but ends up leaving).

Sensitivity can be interpreted as the probability of predicting "Churn" for a customer who actually leaves the company. It is calculated as follows:

$$Sensitivity = \frac{TP}{FN + TP}$$

1.6 Code Structure

The project is organized into several folders, each serving a specific purpose to facilitate project management and development. In the root folder, two key files are found: the main.jl file, which runs the core code of the project and the execution.txt file, which contains the output generated by the main file and through which this report was created.

The "Dataset" folder stores the original dataset in a CSV file, while the "Utils" folder contains the files with the functions developed throughout the course and the necessary for performing correlation analysis. Furthermore, the "Images" folder holds the images generated for this report.

Finally, the "Approaches" folder contains the files for the preprocessing required for each approach. Below is the file hierarchy in which the project is organized.

```
|- Approaches
|    |- Approach1.jl
|    |- Approach2.jl
|    |- Approach4.jl
|    |- Approaches.jl
|
|- Dataset
|    |- telco_churn_data.csv
|
|- Images
|    |- correlation.png
|    |- heatmap.png
|    |- loss.png
|
|- Utils
|    |- utils.jl
|    |- Correlations.jl
|
|- .gitignore
|- execution.txt
```

```
|- main.jl
|- README.md
```

The code is designed to efficiently preprocess the dataset, train machine learning models and evaluate their performance. The main components are organized into various modules, each serving a distinct purpose within the pipeline.

The main.jl file acts as the entry point for the project. It imports the necessary modules, loads the dataset and calls functions from each preprocessing method available in their respective files, except for Approach 3, which internally applies PCA to each fold after using the preprocessing from Approach 2, ensuring that PCA is fitted only on the training data of each fold to prevent data leakage and maintain the integrity of the Cross-Validation process.

After preprocessing, the `execute_approaches` function from the Approaches.jl file is called to perform model cross-validation using the selected models. Finally, the main is responsible for training the final model, based on the configuration selected from the project results.

The Approaches.jl file contains several important functions, such as the `obtain_hyperparameters` function, which defines the hyperparameters for all the models tested during the project, as well as functions for preprocessing different types of variables, including label encoding, proportion encoding and handling latitude and longitude by applying sine or cosine transformations. Moreover, this file contains the `execute_approaches` function, which performs cross-validation on each model. The results are stored in a list, which is then sorted by accuracy to identify the top four models from each approach. These top models are used to create two ensemble models: a Soft Voting model and a Stacking model, both of which are evaluated for performance.

The Utils.jl file includes utility functions such as build and training models, normalizing data and performing other essential operations throughout the process.

Lastly, the Correlations.jl file contains functions used in the fourth approach to analyze correlations between features and the target variable and among features themselves.

1.7 Bibliographic analysis

The issue we aim to address has been extensively studied over the years, as customer churn remains a significant challenge for businesses seeking to stay ahead of their competition.

To tackle this problem, companies have consistently adopted state-of-the-art algorithms available at the time. For instance, in 2015, Vafeiadis et al. conducted a comprehensive comparison of popular machine learning techniques, including Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Decision Trees, Naive Bayes and Logistic Regression, specifically applied to customer churn prediction [3]. These simulations revealed that the two top-performing methods in terms of testing error were an Artificial Neural Network (ANN) with two layers and a Decision Tree classifier, both of which we will explore in this project.

On the other hand, the article by Ibrahim Adedeji and co-authors compares the effectiveness of traditional methods for churn prediction, which, while useful, often fall short due to the complexity and scale of modern telecommunications data [1]. The authors explore decision trees, SVM and ensemble methods to predict customer churn. Additionally, the article analyzes the limitations of traditional methods compared to machine learning models, particularly in terms of scalability and accuracy.

Lastly, Barsotti and co-authors provide a comprehensive review that emphasizes the magnitude of the problem at hand [2]. A survey of the past 10 years examines the evolution of techniques and algorithms for churn prediction in telecommunications. The study highlights the use of models such as Decision Trees, Random Forest, SVM, ANN and ensemble methods. It emphasizes that identifying a "best algorithm" is impractical due to the diversity of datasets and model configurations. Instead, the effectiveness of

algorithms depends on the dataset, the period analyzed and the specific characteristics of the company. The authors argue that the future of machine learning in telecommunications lies in its ability to handle dynamic data, provide personalization and foster innovation in a competitive and constantly evolving environment.

2 First Approach

In the first approach, we will address scaling issues by normalizing or standardizing the numerical variables and we will encode the categorical variables using label encoding, assigning a unique number between 0 and 1 to each class. These preprocessing techniques are very basic and will not allow us to feed the models with all the available features. However, it allows us to set a good starting point analyzing the performance of the models when considering the data in its rawest form.

2.1 Description of the database

All numerical variables will be used, except for *Longitude* and *Latitude*, as these require preprocessing beyond simple normalization or standardization, which will be addressed in approach 2.

All categorical variables will be used, except for *City* and *Zip Code*, as these contain an excessively high number of classes (over a thousand in both cases). Using label encoding within a range of 0 to 1 for these variables would result in a loss of distinction between classes. In approach 2, they will be handled with a different method.

The target feature will be the binary variable *Churn Value*.

2.2 Data preprocessing

- Numerical features: In addition to addressing scaling issues, it is important to ensure that a machine learning model does not receive input values outside the range used during training. For this reason, we will apply min-max normalization to the range [0,1] only to variables where it is safe to assume that the value range observed during training will remain valid over time. In our dataset, only *Age* and *Population* meet this criterion. All other variables have been standardized by setting the mean to 0 and the standard deviation to 1.
- Binary features: All variables originally containing "Yes" or "No" values are encoded into one-hot vectors. The same applies to *Gender*: "Male" or "Female."
- Categorical Features: The previously mentioned label encoding assigns uniformly spaced numbers between 0 and 1 to each class. For example, the four classes of *Internet Type* are encoded as $[0, 0.33, 0.66, 1]$.

A summary of the 37 input features that will be used for training and testing the models is presented in Table 2.

Row	variable Symbol	mean Float64	min Real	median Float64	max Real	nmissing Int64	eltype DataType
1	Referred a Friend	0.457476	0	0.0	1	0	Int64
2	Number of Referrals	9.68509e-17	-0.650362	-0.650362	3.01484	0	Float64
3	Tenure in Months	-1.12993e-16	-1.2789	-0.137998	1.6141	0	Float64
4	Offer	0.259832	0.0	0.0	1.0	0	Float64
5	Phone Service	0.903166	0	1.0	1	0	Int64
6	Avg Monthly Long Distance Charges	-3.03164e-16	-1.4862	-0.00446356	1.7498	0	Float64
7	Multiple Lines	0.421837	0	0.0	1	0	Int64
8	Internet Service	0.783331	0	1.0	1	0	Int64
9	Internet Type	0.448862	0.0	0.333333	1.0	0	Float64
10	Avg Monthly GB Download	-5.04432e-17	-1.00775	-0.196238	3.47945	0	Float64
11	Online Security	0.286668	0	0.0	1	0	Int64
12	Online Backup	0.344881	0	0.0	1	0	Int64
13	Device Protection Plan	0.343888	0	0.0	1	0	Int64
14	Premium Tech Support	0.290217	0	0.0	1	0	Int64
15	Streaming TV	0.384353	0	0.0	1	0	Int64
16	Streaming Movies	0.387903	0	0.0	1	0	Int64
17	Streaming Music	0.353259	0	0.0	1	0	Int64
18	Unlimited Data	0.383927	0	0.0	1	0	Int64
19	Contract	0.377396	0.0	0.0	1.0	0	Float64
20	Paperless Billing	0.592219	0	1.0	1	0	Int64
21	Payment Method	0.249823	0.0	0.0	1.0	0	Float64
22	Monthly Charge	-2.09844e-16	-1.54504	0.210058	1.88014	0	Float64
23	Total Regular Charges	4.03545e-18	-0.997953	-0.390885	2.82604	0	Float64
24	Total Refunds	3.83368e-17	-0.248295	-0.248295	6.05215	0	Float64
25	Total Extra Data Charges	-2.52216e-17	-0.406545	-0.406545	9.04838	0	Float64
26	Total Long Distance Charges	1.53347e-16	-0.88477	-0.410624	3.32556	0	Float64
27	Gender	0.504756	0	1.0	1	0	Int64
28	Age	0.450979	0.0	0.442623	1.0	0	Float64
29	Under 30	0.198921	0	0.0	1	0	Int64
30	Senior Citizen	0.162147	0	0.0	1	0	Int64
31	Married	0.483033	0	0.0	1	0	Int64
32	Dependents	0.23101	0	0.0	1	0	Int64
33	Number of Dependents	2.25985e-16	-0.4868	-0.4868	8.86092	0	Float64
34	Population	0.2102	0.0	0.166641	1.0	0	Float64
35	CLTV	-2.74411e-16	-2.02636	0.107099	1.77481	0	Float64
36	Total Customer Svc Requests	-1.61418e-16	-0.935899	-0.236828	5.35573	0	Float64
37	Product/Service Issues Reported	-4.84255e-17	-0.42941	-0.42941	7.9328	0	Float64

Fig. 2. Approach 1 Inputs summary

2.3 Results

In this section, we proceed to comprehensively evaluate the performance of the four types of machine learning models addressed in this project: Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Decision Trees and k-Nearest Neighbors (kNNs). Eight hyperparameter configurations will be tested for the first two models and six configurations for the latter two.

For this evaluation, the results obtained for the chosen metrics, accuracy and sensitivity, after a 10 fold cross-validation will be presented in tables.

ANNs:

We trained several artificial neural networks (ANNs) with different topologies, using configurations with 1 and 2 hidden layers. To evaluate their performance, we are running 10 executions for each fold to ensure robust results. All models incorporated a validation set comprising 20% of the data and leveraged early stopping to optimize training. This mechanism halted the training process if the validation error did not improve for 15 consecutive cycles, preventing overfitting. Additionally, the minimum training

loss value was set to 0. The selected loss function is the binary crossentropy and the optimizer chosen was Adam (Adaptative Moment Estimation). The results obtained for each model are summarized in Table 1.

Model	**Topology**	**Learning Rate**	max_{Epoch}	$\mu_{Acuracy}$	$\sigma_{Acurary}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	[5]	0.02	250	0.9045	0.0096	0.7808	0.0331
2	[10]	0.02	200	0.9050	0.0097	0.7781	0.0337
3	[40]	0.01	300	0.9051	0.0097	0.7803	0.0331
4	[5 5]	0.02	300	0.9033	0.0082	0.7831	0.0310
5	[20 20]	0.01	200	0.9051	0.0098	0.7830	0.0320
6	[50 50]	0.01	200	0.9058	0.0096	0.7836	0.0331
7	[50 40]	0.01	200	0.9056	0.0093	0.7843	0.0317
8	[100 100]	0.01	200	0.9060	0.0093	0.7852	0.0323

Table 1. ANN models trained on the First Approach

We found the best value for model 8 with an accuracy of 90.60%, although all models exhibit similar behavior in terms of accuracy. The minimal variation in results regardless of topology suggests that the dataset is not complex enough for the ANN to require establishing highly intricate relationships.

The parameters for `maxEpoch` and `learning rate` were selected by analyzing the training curves as shown in the Fig.3, ensuring a steady and continuous learning process while setting the maximum number of epochs to prevent overfitting.

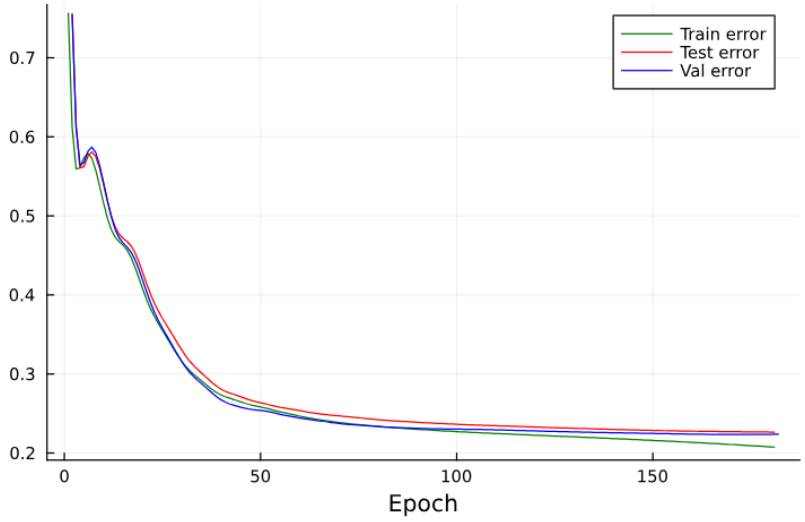

Fig. 3. Loss evolution for the first fold of Model 1

SVMs:

Several Support Vector Machine (SVM) models using different kernel types have been trained to compare their performance on the dataset. These models include linear, polynomial, radial basis function (RBF) and sigmoid kernels, with varying hyperparameters

such as the C value and kernel degree. The models were evaluated based on test sensitivity and accuracy, with the results summarized in Table 2.

Model	Kernel	K Degree	γ_{kernel}	C	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	linear	-	-	0.1	0.9056	0.0092	0.8210	0.0992
2	rbf	-	auto	0.1	0.8921	0.0079	0.7953	0.1508
3	rbf	-	scale	1.0	0.9088	0.0108	0.8404	0.1072
4	rbf	-	1	1.0	0.7410	0.0018	0.9994	0.0009
5	sigmoid	-	-	1.0	0.7346	0.0003	1.0	0.0
6	poly	1	auto	10.0	0.9061	0.0085	0.8228	0.0967
7	poly	2	scale	1.0	0.9088	0.0107	0.8351	0.1175
8	poly	3	2	0.1	0.8671	0.0104	0.7932	0.0562

Table 2. SVM Models from First Approach

The Support Vector Machine (SVM) models demonstrate that the choice of kernel and hyperparameters has a significant impact on their performance. Models with the RBF and polynomial kernels (particularly with scale settings) achieve the highest accuracy, 90.88% and sesitivities, over 83.5% each.

Models 4 and 5 deserve special attention, as both achieve exceptionally high values for the sensitivity metric, while showing poor performance regarding accuracy. This indicates that almost all positive cases are correctly predicted, while all model errors come from negative cases. The interpretation is that these models are biased towards predicting positive values. While maximizing sensitivity is important, such behavior can lead to an imbalance where the model sacrifices accuracy and the ability to generalize well. This emphasizes that an optimal model should strike a balance between sensitivity and accuracy to ensure reliable performance across all classes.

Decision Trees:

Decision trees are a non-linear model that splits the data into branches based on feature thresholds. By training multiple models with varying tree depths, we aimed to evaluate their ability to classify the data. The performance of each model was assessed through cross-validation, with results for sensitivity and accuracy summarized in Table 3.

Model	Max Depth	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	2	0.8327	0.0132	0.7112	0.2792
2	4	0.8749	0.0104	0.6833	0.0289
3	8	0.8864	0.0097	0.7552	0.1014
4	16	0.8597	0.0087	0.7739	0.0737
5	32	0.8518	0.0124	0.7636	0.0724
6	64	0.8518	0.0124	0.7636	0.0724

Table 3. Decision Tree Models from First Approach

For depths ranging from 2 to 8, decision tree models show improved performance as the tree depth increases, achieving an accuracy of 88.64% at a maximum depth of 16. However, at some point between 8 and 16, the achieved accuracies begin to decline.

What is actually happening is that, at excessively high depths, the tree becomes overly complex and starts making decisions based on very specific indicators that are only representative in the training data. This leads to a loss of generalization ability when predicting patterns the model has not seen before. This issue is known as Overfitting.

Another interesting behavior observed in this table is the fact that the results obtained are exactly the same for maximum depths of 32 and 64. The training of a Decision Tree is designed so that decisions regarding the values of the most important features (those that best divide the data) occur in the initial levels of depth. Considering that there are 37 variables in the input and that each depth level represents a decision based on the values of a variable, we can deduce that some features are irrelevant for this algorithm in relation to customer churn, as the threshold choices for their values do not impact the final predictions.

kNNs:

Turning to k-Nearest Neighbors (kNN), a algorithm that classifies data points by identifying the majority class among their closest neighbors, we assess its performance on our dataset. Unlike other models, kNN does not require a separate training phase, which makes it computationally efficient for smaller datasets. In the table 4 we present the results for different kNN configurations, examining their sensitivity and accuracy across varying parameters.

Model	k value	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	2	0.8565	0.0046	0.7122	0.2112
2	3	0.8680	0.0119	0.7844	0.1176
3	5	0.8772	0.0112	0.7901	0.1211
4	10	0.8822	0.0074	0.8057	0.1653
5	15	0.8843	0.0072	0.7882	0.1430
6	30	0.8853	0.0090	0.8040	0.1674

Table 4. kNN Models from First Approach

The kNN models display improved performance as the number of neighbors increases, with the best accuracy of 88.53% achieved at k = 30. The results suggest that while larger k values enhance stability and accuracy, very small k values (e.g., k = 2) may result in predictions that are overly influenced by local data variations, leading to less outcomes due to the limited number of neighbors considered.

Ensemble Models:

Furthermore, we explore two ensemble methods: Soft Voting and Stacking. Both approaches combine the predictions of multiple individual models to improve overall performance and provide more robust and stable results.

Soft Voting is based on the weighted average of the prediction probabilities from each classifier. In this method, each model contributes its "opinion" on the most probable class and the final result is determined by the total probability of each class. The weights for each model are assigned according to their accuracy, with the top model receiving the highest weight. In this case, the weights are distributed linearly from 2 to 1 in equal intervals, depending on the number of models in the ensemble. For instance, if there are 4 models, the weights assigned would be [2, 1.66, 1.33, 1], decreasing by an equal step (approximately 0.33) between each successive weight. With the highest weight given to the model with the highest accuracy, we ensure that the more precise models have a greater influence on the final decision.

On the other hand, Stacking employs a more complex strategy. In this case, several base models (of different types or configurations) are trained independently and their outputs are used as inputs for a higher-level model. This model learns to combine the base model predictions to make a more accurate final prediction. The stacking method here involves using a Support Vector Machine (SVM) as the model to combine the outputs of the base models.

Both ensemble methods were built by combining the four best-performing models in terms of accuracy from this approach. The selected models for the ensembles, ordered from highest to lowest accuracy, are:

(1) Model: SVM - 7 (*Acc*: 0.9088, *Sens*: 0.8351)

(2) Model: SVM - 3 (*Acc*: 0.9088, *Sens*: 0.8404)

(3) Model: SVM - 6 (*Acc*: 0.9061, *Sens*: 0.8228)

(4) Model: ANN - 8 (*Acc*: 0.9060, *Sens*: 0.7852)

The results obtained from applying these ensemble techniques are presented in the table below (Table 5).

Model	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
Soft Voting	0.9088	0.0083	0.7715	0.0348
Stacking	0.9052	0.0091	0.7277	0.0271

Table 5. Ensemble Models from First Approach

Overall, it seems that ensemble models are not very effective for this problem, at least in this initial approach. Neither of the two models managed to improve the accuracy achieved by the two best SVMs individually. Additionally, the sensitivity values of the ensembles are lower than those obtained by the four models that compose them individually. Finally, it is also noteworthy that better performance in both metrics is observed with the Soft Voting ensemble compared to Stacking, despite its lower complexity.

2.4 Discussion

In this proyect, five machine learning models will be considered: SVM, Decision Trees, kNN, ANNs and Ensemble Models. Each has its advantages and disadvantages and their application to a problem depends on its nature and complexity. To better understanding of their behavior, several different configurations were studied for each type of model.

Recapping the results obtained in the first approach, we found that the best performances were achieved by the SVM models, reaching accuracies over 90% and sensitivities over 82% in the most efficient configurations. However, care must be taken with bias issues in certain hyperparameters choices (sigmoid kernel and RBF kernel with $\gamma = 1$).

Notably, the most accurate model of the entire approach was SVM-7, with a polynomial kernel of degree 2, $\gamma : scale$ and $C = 1$, achieving an accuracy of 90.88% and a sensitivity of 83.51%.

The other models also achieved acceptable results, with ANNs standing out for obtaining accuracy values similar to those mentioned in the most optimal SVMs but with lower sensitivity of around 78%. The ANNs have also showed very little variability on the metric scores across diferent architectures.

On the other hand, kNN and Decision Trees were evaluated as the two least effective types of models, although both achieve to surpass 88% accuracy in their best configurations. They show very similar scores in accuracy, but considering sensitivity we prefer the results from kNN models.

Although ensemble techniques like Soft Voting and Stacking were implemented to combine the strengths of individual models, their performance in this initial approach did not surpass the accuracy or sensitivity of the top-performing SVMs.

3 Second Approach

In the second approach, the methodology will be similar to the first one regarding the normalization or standardization of numerical variables. However, categorical features will be handled differently by establishing a proportional relationship based on the percentage of people in each category who left the company. In this second approach we can also appreciate how more advanced normalization and encoding techinques allow us to use a broader range of features of different types, thus allowing our models to leverage more information.

3.1 Description of the database

All numerical variables will be used. However, due to the geographical properties of Latitude and Longitude, they will be treated differently using trigonometric functions.

In this approach, all categorical variables will be used, including *City* and *Zip Code*. To handle these two new variables alongside the other categorical variables, label encoding will not be used. Instead, a proportion encoder will be applied, allowing these features to obtain a numeric significance related to the target feature.

As in the previous approach, the target variable will be *Churn Value*.

3.2 Data preprocessing

- Numerical features: These have been processed in the same way as explained in approach 1, except for *Latitude* and *Longitude*, as mentioned earlier. *Latitude*, with values in the range $[-90°, 90°]$, is converted using the sine function to a value between [-1, 1], preserving all its information. This new variable will be called *Latitude_Sin*. This variable does not need further normalization as its range is already comparable to a normalized variable.

  Longitude, which takes values in the range $[-180°, 180°]$, is represented using both sine and cosine functions to retain its cyclical behavior. These transformations result in two new variables, *Longitude_Sin* and *Longitude_Cos*, each ranging between [-1, 1]. Like *Latitude*, these features do not require normalization due to their constrained ranges.

- Binary features: Processed identically to the first approach.

- Categorical Features: The proportion encoder computes the percentage of churn (value 1 in the *Churn Value* feature) for each category relative to the total number of occurrences of that category. This encoding assigns a percentage value between 0 and 1 to each category, creating a direct numerical association between the categorical features and the target variable. Additionally, after calculating the proportion for each variable, a Min-Max normalization is applied within each feature to ensure all values are scaled between 0 and 1, while preserving the relative relationships between the data.

A summary of the 42 variables that make up the input for the models in Approach 2 is presented in the Figure 4.

Row	variable Symbol	mean Float64	min Real	median Float64	max Real	nmissing Int64	eltype DataType
1	Referred a Friend	0.457476	0	0.0	1	0	Int64
2	Number of Referrals	9.68509e-17	-0.650362	-0.650362	3.01484	0	Float64
3	Tenure in Months	-1.12993e-16	-1.2789	-0.137998	1.6141	0	Float64
4	Offer	0.428813	0.0	0.441188	1.0	0	Float64
5	Phone Service	0.903166	0	1.0	1	0	Int64
6	Avg Monthly Long Distance Charges	-3.03164e-16	-1.4862	-0.00446356	1.7498	0	Float64
7	Multiple Lines	0.421837	0	0.0	1	0	Int64
8	Internet Service	0.783331	0	1.0	1	0	Int64
9	Internet Type	0.432732	0.0	0.259998	1.0	0	Float64
10	Avg Monthly GB Download	-5.04432e-17	-1.00775	-0.196238	3.47945	0	Float64
11	Online Security	0.286668	0	0.0	1	0	Int64
12	Online Backup	0.344881	0	0.0	1	0	Int64
13	Device Protection Plan	0.343888	0	0.0	1	0	Int64
14	Premium Tech Support	0.290217	0	0.0	1	0	Int64
15	Streaming TV	0.384353	0	0.0	1	0	Int64
16	Streaming Movies	0.387903	0	0.0	1	0	Int64
17	Streaming Music	0.353259	0	0.0	1	0	Int64
18	Unlimited Data	0.383927	0	0.0	1	0	Int64
19	Contract	0.554047	0.0	1.0	1.0	0	Float64
20	Paperless Billing	0.592219	0	1.0	1	0	Int64
21	Payment Method	0.538225	0.0	0.87125	1.0	0	Float64
22	Monthly Charge	-2.09844e-16	-1.54504	0.210058	1.88014	0	Float64
23	Total Regular Charges	4.03545e-18	-0.997953	-0.390885	2.82604	0	Float64
24	Total Refunds	3.83368e-17	-0.248295	-0.248295	6.05215	0	Float64
25	Total Extra Data Charges	-2.52216e-17	-0.406545	-0.406545	9.04838	0	Float64
26	Total Long Distance Charges	1.53347e-16	-0.88477	-0.410624	3.32556	0	Float64
27	Gender	0.504756	0	1.0	1	0	Int64
28	Age	0.450979	0.0	0.442623	1.0	0	Float64
29	Under 30	0.198921	0	0.0	1	0	Int64
30	Senior Citizen	0.162147	0	0.0	1	0	Int64
31	Married	0.483033	0	0.0	1	0	Int64
32	Dependents	0.23101	0	0.0	1	0	Int64
33	Number of Dependents	2.25985e-16	-0.4868	-0.4868	8.86092	0	Float64
34	City	0.26537	0.0	0.25	1.0	0	Float64
35	Zip Code	0.26537	0.0	0.25	1.0	0	Float64
36	Population	0.2102	0.0	0.166641	1.0	0	Float64
37	CLTV	-2.74411e-16	-2.02636	0.107099	1.77481	0	Float64
38	Total Customer Svc Requests	-1.61418e-16	-0.935899	-0.236828	5.35573	0	Float64
39	Product/Service Issues Reported	-4.84255e-17	-0.42941	-0.42941	7.9328	0	Float64
40	Latitude_Sin	0.590018	0.538121	0.590683	0.668639	0	Float64
41	Longitude_Sin	-0.867528	-0.912171	-0.869536	-0.826085	0	Float64
42	Longitude_Cos	-0.495966	-0.563546	-0.49387	-0.40981	0	Float64

Fig. 4. Approach 2 inputs summary

3.3 Results

As in the previous approach, we tested various configurations for each type of model: 8 different architectures for Artificial Neural Networks, with 1 or 2 hidden layers; 8 hyperparameter configurations for Support Vector Machines, varying kernels and C values; 6 different depth values for Decision Trees; and 6 k values for the k-Nearest Neighbors algorithm.

ANNs:

Again, all architectures use 10 executions per fold, a 20% ratio for the validation set, a maximum of 15 cycles without improvement in validation error and a minimum error of 0. The selected loss function is binary crossentropy and the optimizer used was ADAM.

The table 6 shows the results of the ANNs analyzed during this approach:

Model	Topology	Learning Rate	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	[5]	0.02	0.9336	0.0048	0.8530	0.0185
2	[10]	0.02	0.9336	0.0052	0.8522	0.0200
3	[40]	0.01	0.9342	0.0053	0.8527	0.0203
4	[5, 5]	0.02	0.9326	0.0054	0.8564	0.0170
5	[20, 20]	0.01	0.9334	0.0050	0.8536	0.0181
6	[50, 50]	0.01	0.9339	0.0052	0.8540	0.0196
7	[50, 40]	0.01	0.9338	0.0055	0.8545	0.0202
8	[100, 100]	0.01	0.9340	0.0055	0.8542	0.0198

Table 6. ANN Models from Second Approach

As observed, all models exhibit similar behavior regardless of their topology. However, it is worth highlighting that models with a topology featuring a higher number of neurons achieve the best results, reaching a maximum Accuracy of 93.42% for the case of 1 hidden layer in model number 3, outperforming all models with 2 hidden layers.

SVMs:

In Table 7, we can see the results of the executions of the Support Vector Machine models selected for this approach, which are the same as in the previous one (variations in kernel type, kernel degree, kernel gamma and C value).

Model	Kernel	Degree	γ_{Kernel}	C	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	linear	-	-	0.1	0.9328	0.0056	0.8566	0.0604
2	rbf	-	auto	0.1	0.9134	0.0068	0.8297	0.1259
3	rbf	-	scale	1.0	0.9324	0.0061	0.8446	0.0705
4	rbf	-	1.0	1.0	0.7376	0.0027	0.9990	0.0014
5	sigmoid	-	-	1.0	0.7346	0.0003	1.0000	0.0000
6	poly	1	auto	10.0	0.9333	0.0049	0.8651	0.0545
7	poly	2	scale	1.0	0.9344	0.0054	0.8491	0.0664
8	poly	3	2.0	0.1	0.8961	0.0092	0.8191	0.0456

Table 7. SVM Models from Second Approach

In this case, models with the polynomial kernel of degree 1 (auto) and 2 (scale) show the best accuracy results, achieving 93.33% and 93.44%, respectively, while models 4 and 5 continue to yield lower accuracy values compared to other models, with 73.76% and 73.46%. However, they stand out in terms of sensitivity, reaching nearly 100%. As was explained in the first approach, this is because these models are biased towards predicting positive values.

Decission Trees:

For Decision Trees, whose results are shown in Table 8, the maximum depths used are the same as those tested in the first approach.

Model	Max Depth	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	2	0.8370	0.0124	0.7293	0.2460
2	4	0.8827	0.0093	0.7438	0.1207
3	8	0.8996	0.0138	0.7993	0.0611
4	16	0.8867	0.0111	0.7997	0.0482
5	32	0.8847	0.0114	0.7921	0.0491
6	64	0.8847	0.0114	0.7921	0.0491

Table 8. Decision Tree Models from Second Approach

As in the first approach, the best result was found at a depth of 8, with an accuracy of 89.96%. Beyond this point, performance begins to decline slightly, as in the previous case, due to overfitting issues. Similarly, for sensitivity, the same pattern occurs from a depth of 16 onward, with an optimal result of 79.97%, respectively.

As in the previous approach, the exact same results are observed for maximum depths of 32 and 64. Following the same reasoning as before, there are features that do not influence the predictions of these models, as increasing the depth beyond a certain point adds no value because the patterns are sufficiently separated by the decisions already made.

kNNs:

Finally, Table 9 shows the results obtained by applying the kNN algorithm. In this case, the same hyperparameter configurations as in the first approach were also used.

Model	k Value	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	2	0.8674	0.0077	0.7645	0.2144
2	3	0.8812	0.0095	0.7964	0.1128
3	5	0.8897	0.0145	0.8079	0.1141
4	10	0.8934	0.0069	0.7916	0.1542
5	15	0.8955	0.0057	0.8030	0.1350
6	30	0.8935	0.0097	0.7902	0.1535

Table 9. kNN Models from Second Approach

In this case, the results improve until reaching the best values when $k = 15$, with 89.55% accuracy and 80.30% sensitivity, followed by a slight decrease for $k = 30$.

Thanks to the easy behavior interpretation offered by the kNN algorithm, we can clearly break down what happens internally as the values of k increase. When the model evaluates a target customer, this parameter represents the number of most similar customers considered to predict whether the target customer will leave the company or not. This evaluation is based on a majority vote. For k values between 2 and 15, considering a larger number of nearby neighbors allows the model to improve the quality of its predictions. As k increases, the accuracy progressively rises, reaching a maximum of 89.55%. However, beyond $k = 15$, the number of customers analyzed becomes excessive, leading the model to consider patterns that are too different. This causes accuracy to begin declining as k continues to increase.

Ensemble models:

The 4 best models for constructing the ensemble will now be presented, using Soft Voting where the highest weight is assigned to the model with the highest accuracy and using the Stacking method as explained in the previous approach:

(1) Model: SVM - 7 (Acc: 0.9344; Sens: 0.8491)

(2) Model: ANN - 3 (Acc: 0.9342; Sens: 0.8527)

(3) Model: ANN - 8 (Acc: 0.9340; Sens: 0.8542)

(4) Model: ANN - 6 (Acc: 0.9339; Sens: 0.8540)

The results obtained from applying these ensemble techniques are presented in the Table 10.

Model	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
Soft Voting	0.9361	0.0058	0.8523	0.0245
Stacking	0.9354	0.0072	0.8325	0.0263

Table 10. Ensemble Models from Second Approach

Unlike the previous approach, the ensemble models show an improvement of 0.17% and 0.10% in accuracy for Soft Voting and Stacking, respectively, compared to the best model in this approach, SVM-7. This suggests that they are better capturing the relationship between the data and providing improved results. However, this improvement is still minimal and not a good practice, as it significantly increases the model's complexity and computational cost.

3.4 Discussion

As mentioned in the section 1.4, the goal of the first two approaches is to evaluate how more advanced data preprocessing affects the performance of the different models. Therefore, this discussion will consist of two parts: a discussion focused on the results found within this approach itself and an analysis of how the results obtained after these more advanced preprocessing techniques compare to those in approach 1.

Evaluation of models performance

For ANNs, a slight improvement is observed as the number of neurons per layer increases. However, performance differences between architectures are minor: all accuracy scores are around 93.3%.

Regarding SVMs, their performance once again proves to be more variable depending on hyperparameter configurations. The most optimal configurations surpass ANNs in terms of both accuracy and sensitivity. Notably, among SVMs, we find the most accurate model in this approach, SVM-7, with 93.44%. Its kernel is polynomial of degree 2, with $\gamma : scale$.

For Decision Trees and kNN, the maximum accuracy values are 89.96% and 89.55%, respectively. Approximately 3% lower than SVMs and ANNs, which deliver better results.

Regarding the 4 best models used to build the ensemble and the ensemble itself, we believe that these techniques do not maximize model performance, given the added complexity to the process and the minimal improvement of only 0.17%.

Among our top models –3 ANNs and 1 SVM– it is important to note that outside of these 4, there are other models like SVM-6, with an accuracy of 93.33%, just 0.11% behind our best model (another SVM) and with a better sensitivity of 86.51%. This model could potentially outperform the 4 chosen for the ensemble, as the second metric, while not our primary goal, also holds importance.

Evaluation of advanced preprocessing techniques

By comparing model performance in approaches 1 and 2, we can deduce whether the new preprocessing techniques and the additional features they enabled to introduce were beneficial to the machine learning models performances.

Compared to the first approach, the results of the second approach show notable improvements in several models, particularly in Artificial Neural Networks (ANNs). On average, accuracy and sensitivity increased by 3% and 7%, respectively, for ANN models. This indicates that the more advanced dataset preprocessing provides valuable information that ANNs can effectively leverage.

In SVM models, the best configurations in approach 2 also improved their accuracies compared to approach 1. For instance, SVM-7, the most accurate model in this approach, improved from 90.88% in the previous approach to 93.44% in this one.

For Decision Trees, the behavior is similar across both approaches, but with a 2% improvement in accuracy and a 5% improvement in sensitivity in the second approach.

On the other hand, kNN models show less variability between the two approaches. The maximum accuracy in the second approach slightly surpasses the first (by 1%), while sensitivity is slightly lower (by 0.27%). The fact that kNN models did not improve that much is less concerning as our main focus is on the best performing ANNs and SVMs.

Regarding the ensemble models, the values for accuracy and sensitivity increased by 3% and 7%, respectively. In this case, the 4 most accurate models include 3 ANNs and 1 SVM, whereas in the previous approach, they consisted of 3 SVMs and 1 ANN. This shift suggests that the new preprocessing particularly benefited the ANN models compared to the SVMs.

In conclusion, the new preprocessing techniques and the additional features they enabled allow the machine learning models to gain new insights from the data, improving their performance in both metrics evaluated. For this reason, the dataset generated by the preprocessing of approach 2 will serve as the foundation for the dimensionality reduction technique carried out in the next approach.

4 Third Approach

In this third approach, the focus is on reducing the dimensionality of the dataset in order to improve the computational efficiency of the models. So far in this project, the second approach has obtained the best results, so we will use the dataset obtained from the preprocessing performed for the approach 2 as the base for a Principal Components Analysis (PCA). This method allows to lower the number of input features while capturing most of the variability in the dataset by transforming the data with a projection onto a set of orthogonal axes. The downfall of this method is the lost of explainability of the model inputs, as the new components are linear transformations of the original features and usually they do not have a clear interpretation.

4.1 Description of the database

Since approach 2 is used as the base, the data remains the same. However, after applying PCA setting it to capture 95% of the variability from the original data, the number of features that form the input of the models will be reduced.

4.2 Data preprocessing

The data preprocessing regarding normalization, standardization and categoricals encoding is the same as the one carried out in Approach 2.

4.3 Combining PCA and Cross Validation

When performing Principal Component Analysis (PCA) on a dataset, it is essential that the transformation is fitted exclusively using the training data. This involves calculating parameters such as the means of the variables and their principal components (eigenvectors). Once the reduced space is defined using the training set, the test dataset is transformed into that same space.

This procedure is crucial to avoid data leakage: the characteristics of the test dataset must not influence the training of the model. In the cross-validation technique, different training and test sets are defined before each fold. Therefore, it is necessary to generate independent PCA transformations for each training set used in each fold for a realistic evaluation of its impact, as it simulates how the model would perform in a real-world scenario with previously unseen data transformed solely based on the information available during training.

After configuring the PCA transformations to capture 95% of the variability contained in the 10 training sets generated during the 10 folds of cross-validation, it was determined that 21 principal components are required for all of them. This means we have successfully reduced the 42 features from Approach 2 to 21 features while retaining 95% of the variability they contained.

Table 5 presents the 21 principal components generated for the dataset of the first fold in cross-validation. As explained, the whole dataset was transformed with a PCA that was fitted using the training subset. Unlike the other approaches, in this table the variables have no specific names, they are simply referred to as components.

Row	variable	mean	min	median	max	nmissing	eltype
	Symbol	Float32	Float32	Float64	Float32	Int64	DataType
1	x1	1.73321e-8	-3.45149	-0.397859	7.6005	0	Float32
2	x2	6.93286e-8	-4.61261	0.0724112	6.13399	0	Float32
3	x3	0.0	-3.38559	-0.0631704	5.84287	0	Float32
4	x4	8.66607e-9	-3.108	-0.063016	5.22922	0	Float32
5	x5	2.59982e-8	-3.70299	-0.0444683	5.06547	0	Float32
6	x6	7.58281e-9	-1.1319	-0.219076	6.28107	0	Float32
7	x7	4.11639e-8	-3.16304	-0.010242	4.69986	0	Float32
8	x8	5.19964e-8	-3.48516	-0.012468	5.32547	0	Float32
9	x9	-1.51656e-8	-3.03661	0.0108968	4.4874	0	Float32
10	x10	-1.94987e-8	-3.35802	0.0233194	4.23231	0	Float32
11	x11	-1.29991e-8	-2.57395	-0.0263028	3.09168	0	Float32
12	x12	4.33304e-9	-1.5614	0.0314287	1.62507	0	Float32
13	x13	-3.57476e-8	-1.07039	0.0403597	1.11182	0	Float32
14	x14	2.16652e-8	-1.75066	-0.0202296	1.77609	0	Float32
15	x15	2.16652e-9	-1.17521	0.0311753	1.27343	0	Float32
16	x16	-1.08326e-8	-1.11334	-0.116907	1.25457	0	Float32
17	x17	3.03313e-8	-1.42293	-0.0121694	1.60686	0	Float32
18	x18	2.11236e-8	-1.28783	0.00685476	1.32712	0	Float32
19	x19	1.51656e-8	-1.28233	-0.0191038	1.47127	0	Float32
20	x20	-1.73321e-8	-1.2496	-9.13802e-5	1.28798	0	Float32
21	x21	-2.22068e-8	-1.1695	-0.0208912	1.21769	0	Float32

Fig. 5. Approach 3: summary of the first fold training inputs obtained applying PCA

4.4 Results

As in the previous approaches, we evaluated the same models using the same configurations and the evaluation was carried out using the same 10-fold cross-validation method, maintaining accuracy and sensitivity as the key metrics.

ANNs:

Again, all architectures use 10 executions per fold, a 20% ratio for the validation set, a maximum of 15 cycles without improvement in validation error and a minimum error of 0. The results for each Artificial Neural Network models are summarized in Table 11.

Model	Topology	Learning Rate	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	[5]	0.02	0.9068	0.0126	0.7871	0.0417
2	[10]	0.02	0.9075	0.0119	0.7896	0.0429
3	[40]	0.01	0.9080	0.0124	0.7913	0.0429
4	[5, 5]	0.02	0.9069	0.0107	0.7924	0.0375
5	[20, 20]	0.01	0.9076	0.0127	0.7932	0.0433
6	[50, 50]	0.01	0.9074	0.0128	0.7940	0.0443
7	[50, 40]	0.01	0.9083	0.0121	0.7953	0.0417
8	[100, 100]	0.01	0.9080	0.0121	0.7946	0.0415

Table 11. ANN Models from Third Approach

As in the previous two approaches, all ANNs achieve very similar results across configurations. Accuracies range between 90.68% and 90.83%, while sensitivities range between 78.71% and 79.46%. Regardless of whether the architectures have one or two layers, a higher number of neurons slightly improves the precision of the models.

SVMs:

Table 12 shows the results of the executions of the Support Vector Machine models selected for this approach, which are the same as in the previous cases.

Model	Kernel	Degree	γ_{Kernel}	C	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	linear	-	-	0.1	0.9079	0.0125	0.8243	0.0778
2	rbf	-	auto	0.1	0.8959	0.0168	0.7803	0.1333
3	rbf	-	scale	1.0	0.9123	0.0162	0.8347	0.0947
4	rbf	-	1.0	1.0	0.7501	0.0051	0.9957	0.0022
5	sigmoid	-	-	1.0	0.7346	0.0003	1.0000	0.0000
6	poly	1	auto	10.0	0.9086	0.0134	0.8260	0.0769
7	poly	2	scale	1.0	0.8866	0.0145	0.7717	0.1871
8	poly	3	2.0	0.1	0.8610	0.0091	0.8060	0.0689

Table 12. SVM Models from Third Approach

Once again, we observe that the results obtained by SVMs are much more variable depending on the chosen hyperparameters. The most efficient SVMs are on pair with ANNs in terms of accuracy and far surpass them in sensitivity. Model 3 stands out above the rest, with the RBF kernel and γ set to "scale", outperforming all ANNs in both metrics: it achieves 91.23% accuracy and 83.47% sensitivity.

As in the previous two approaches, models 4 and 5 continue to show bias issues toward positive predictions, indicated by exceptionally high sensitivities and low accuracies.

Decision Trees:

Table 13 shows the results obtained by Decision Trees with different maximum depths.

Model	**Max Depth**	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	2	0.8492	0.0167	0.8135	0.1660
2	4	0.8657	0.0150	0.8358	0.1334
3	8	0.8645	0.0135	0.7876	0.1141
4	16	0.8454	0.0095	0.7780	0.0988
5	32	0.8396	0.0109	0.7812	0.0908
6	64	0.8396	0.0109	0.7812	0.0908

Table 13. Decision Tree Models from Third Approach

Increasing the maximum depth to at least 4 helps Decision Trees make more informed decisions, thereby improving their accuracies. However, as previously explained, they eventually encounter overfitting issues. In this table, the highest recorded accuracy is 86.45%, achieved with a depth of 4.

Once again, the results for maximum depths of 32 and 64 are identical, indicating that certain principal components do not influence the predictions of the Decision Trees.

kNNs:

Let us now examine the results obtained by the kNNs for different values of k, as summarized in Table 14.

Model	**k Value**	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	2	0.8536	0.0117	0.7202	0.2107
2	3	0.8661	0.0099	0.7961	0.1142
3	5	0.8753	0.0147	0.7998	0.1228
4	10	0.8833	0.0120	0.7875	0.1551
5	15	0.8864	0.0104	0.7742	0.1297
6	30	0.8863	0.0124	0.7587	0.1463

Table 14. kNN Models from Third Approach

The accuracy scores of the kNN algorithm increase significantly as k increases, from 85.36% for $k = 2$ to 88.64% at $k = 15$. However, beyond this value, increasing k does not improve the algorithm's precision, as it begins to consider patterns that are too diverse for accurate predictions, as previously explained.

Ensemble models:

The 4 models with the best accuracy values will now be presented and they will be used to evaluate the two ensembling techniques addressed in this project: Soft Voting and Stacking. The results obtained from applying these techniques are shown in the table below (Tab. 15).

(1) Model: SVM - 3 (Acc: 0.9123; Sens: 0.8347)

(2) Model: SVM - 6 (Acc: 0.9086; Sens: 0.8260)

(3) Model: ANN - 7 (Acc: 0.9083; Sens: 0.7953)

(4) Model: ANN - 8 (Acc: 0.9080; Sens: 0.7946)

Model	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
Soft Voting	0.9144	0.0124	0.7897	0.0432
Stacking	0.9150	0.0146	0.7614	0.0546

Table 15. Ensemble Models from Third Approach

Despite the added complexity that ensembling techniques entail, they have only improved the accuracy achieved by SVM-3 individually by 0.21% and 0.27%. Moreover, the sensitivity metric has decreased significantly. Therefore, we conclude that ensembling techniques are not an appropriate approach in this case.

4.5 Discussion

This section will be divided into two parts: first, we will evaluate the overall performance of the four types of models for this third approach. Then, we will analyze whether the dimensionality reduction of the dataset from approach 2 using PCA has caused a significant decline in model performance.

Evaluation of models performance

As in the previous two approaches, we observed that the most optimal models in terms of accuracy and sensitivity are the ANNs and SVMs. The former provide very consistent results across different configurations, while the various SVM configurations lead to more varied outcomes.

The most optimal SVM configuration, SVM-3, achieved 91.23% in accuracy and 83.47% in sensitivity. Both values surpass the best results of any ANN. Notably, SVM-7, which had been the most accurate model in the previous two approaches, has underperformed in this approach compared to other configurations.

A step below are the kNN algorithms. The most accurate configuration achieved a score of 88.64%, accompanied by 77.42% in sensitivity.

Finally, Decision Trees were the least efficient models in this approach, with a maximum accuracy of 86.57%.

Once again, we have decided that Ensemble models do not justify the added complexity with performance improvements

Evaluation of dimensionality reduction by PCA

As mentioned during the description of the data processing for this third approach, the dimensionality of the input was reduced from 42 features in the second approach to 21 principal components in this approach. This reduction is highly valuable from a computational efficiency perspective. The impact of the information loss can be assessed by comparing the performance of the models between approaches 2 and 3.

Since ANNs consistently produce stable results across different configurations, they provide strong support point for this comparison. In approach 2, accuracy values hovered around 93.3%, compared to 90% after applying PCA. A percentage drop of more than 3 points represents a significant loss of information, but considering that the number of features was halved, it indicates that the dataset transformation through PCA was effective.

For SVMs, Decision Trees and kNNs, the differences in accuracy for the best configurations range between 2% and 3%.

Although the loss of information is notable, it is not severe enough to conclude that PCA was an inappropriate approach for dimensionality reduction. It achieved a significant reduction in dimensionality while maintaining reasonably high model performance.

In the fourth and final approach, we will explore another method based on feature selection, aiming for a dimensionality reduction strategy more committed to preserving information.

5 Fourth Approach

In this final approach, we will maintain the focus on dimensionality reduction and we will perform a correlation analysis between the different features to identify the most influential ones in customer churn. Unlike PCA, this approach does not compromise the model's explainability, as the variables used to train the models will be still known to us. This is useful, for instance, to make precise diagnoses to devise strategies that encourage customer retention.

5.1 Description of the database

For this approach, all the variables available in the dataset will be used. Our next goal is to determine which ones are truly influential in customer churn (*Churn Value* will again be our target feature).

5.2 Data preprocessing

For feature selection, we will rely on Pearson's correlation coefficient, which for two numerical variables x and y is defined as follows:

$$r = \frac{\mathrm{Cov}(x, y)}{\sigma_x \cdot \sigma_y}$$

This coefficient quantifies the linear relationship between two variables. Its value lies in the range [-1, 1]: the closer it is to 1, the more we interpret that an increase in x is associated with an increase in y; if the value is close to -1, then an increase in x is associated with a decrease in y. Finally, for values near 0, we conclude that there is no linear relationship between x and y.

To use Pearson's coefficient, we need our features to be numeric or binary and our preprocessing will focus on this. We will use different feature transformation methods to achieve this:

The numerical or binary features are normalized as described for the other approaches and the cyclical variables *Latitude* and *Longitude* are also included under their trigonometric transformations.

For categorical features, we have two cases depending on the number of classes in each variable:

- *City* and *Zip Code* will be encoded based on the proportion of churns in each class, as detailed in Approach 2. This method gives them numerical significance.
- Other features with a limited number of classes will be processed using one-hot encoding. This transformation will result in multiple binary features. For example, *Contract* will be split into *Contract_Month-to-Month*, *Contract_One-Year* and *Contract_Two-year*, each of which will have a value of 1 if the customer has that contract type and 0 otherwise.

5.3 Feature selection

After applying the aforementioned transformations to the original dataset, we obtain a total of 55 numerical or binary features. By calculating the Pearson correlation coefficient of each variable with respect to the target ,*Churn Value*, we can evaluate the linear relationship of each variable with customer churn. By imposing the condition $|r_i| > 0.2$, we obtain the 18 features most correlated with churn, either directly or inversely. These 18 variables and their correlations with the target feature are listed in Table 6, while Figure 7 allows visualizing them. The threshold of 0.2 was proposed as a balanced cutoff to retain variables with at least some relevance, but it can be easily changed in the source code, automatically resulting in a different set of retained features.

Row	Variable String	Correlacion Float64
1	Number of Referrals	-0.287
2	Tenure in Months	-0.353
3	Internet Service	0.228
4	Monthly Charge	0.232
5	Total Long Distance Charges	-0.224
6	Dependents	-0.249
7	Number of Dependents	-0.219
8	City	0.443
9	Zip Code	0.535
10	Churn Value	1.0
11	Total Customer Svc Requests	0.537
12	Product/Service Issues Reported	0.399
13	Offer_Offer E	0.215
14	Internet Type_Cable	0.301
15	Internet Type_None	-0.228
16	Contract_Month-to-Month	0.448
17	Contract_Two Year	-0.328
18	Payment Method_Credit Card	-0.219

Fig. 6. Features with a correlation absolute value with *Churn Value* greater than 0.2

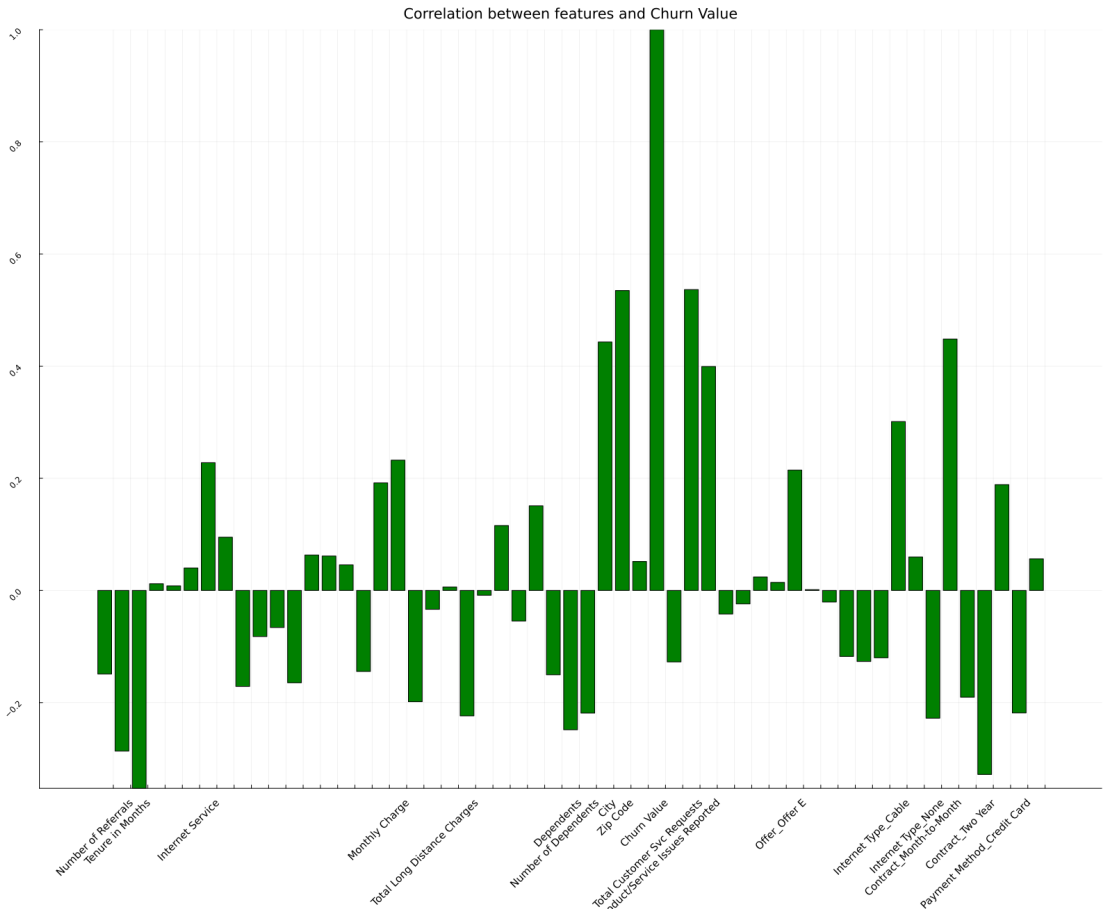

Fig. 7. Correlations with *Churn Value*

The next step in our analysis is to study the correlations between the remaining 18 features. If we detect that two features are highly linearly correlated, we will remove one to avoid redundancy, keeping in mind our goal of dimensionality reduction. To achieve this, we calculate the correlation matrix for the 18 variables, with its elements representing the correlation coefficients between every pair of features. The heatmap in Figure 8 provides a clear visualization.

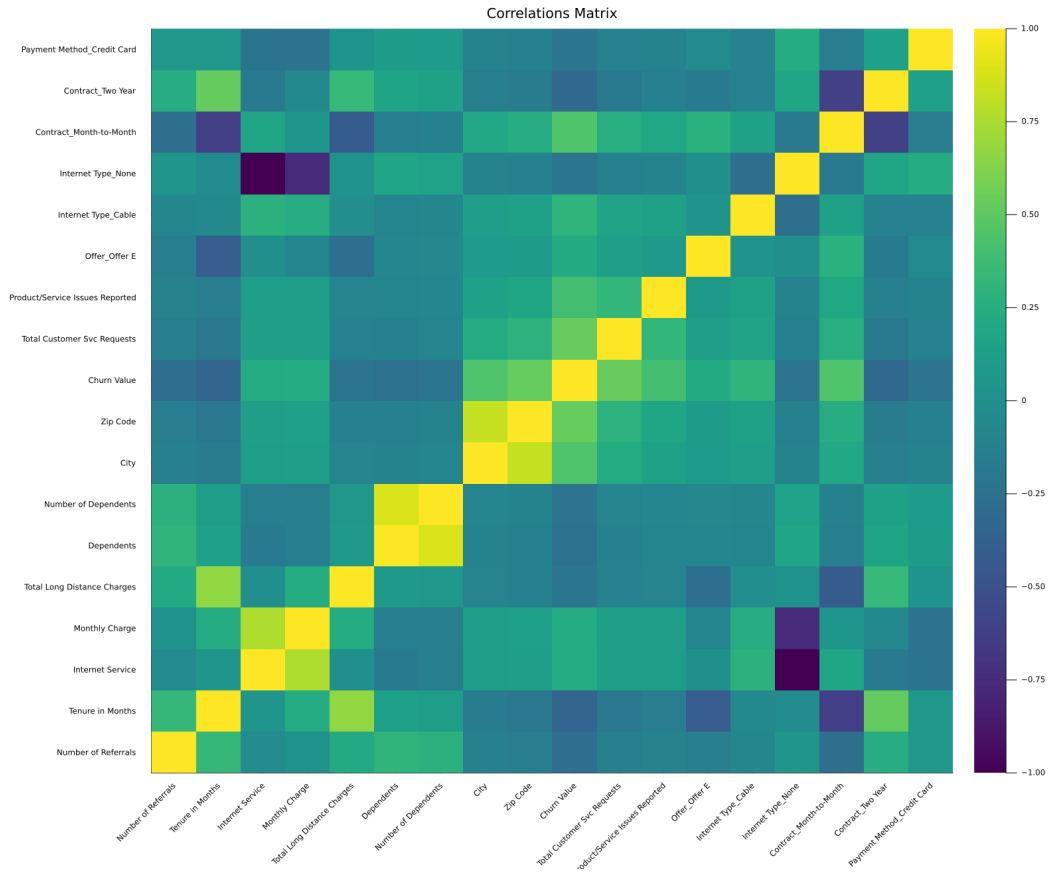

Fig. 8. Heat Map of the correlation matrix among the 18 most relevant features

The threshold established for this project is $|r| > 0.8$, although, again, it can be easily changed in the source code. The pairs of highly correlated variables are:

- *Internet Service* and *Internet Type_None*: $r = -1$.

  If a person does not have an internet service, they will obviously not have any associated internet type. Between these two features, we choose to keep *Internet Service* as it is one of the original dataset features.

- *Dependents* and *Number of Dependents*: $r = 0.89$.

  Again, the association is clear. We choose to retain *Number of Dependents*, which provides more information.

- *City* and *Zip Code*: $r = 0.83$.

  Both variables provide information about the customer's location, but we prefer to use *Zip Code* as it is slightly more detailed.

If the threshold for correlation between features is changed, the selection of variables must be made manually by the user in the source code, following reasoning similar to what has been presented here.

The summary of the inputs used to train the models in our project can be found in Table 9.

Row	variable Symbol	mean Float64	min Real	median Float64	max Real	nmissing Int64	eltype DataType
1	Number of Referrals	9.68509e-17	-0.650362	-0.650362	3.01484	0	Float64
2	Tenure in Months	-1.12993e-16	-1.2789	-0.137998	1.6141	0	Float64
3	Monthly Charge	-2.09844e-16	-1.54504	0.210058	1.88014	0	Float64
4	Total Long Distance Charges	1.53347e-16	-0.88477	-0.410624	3.32556	0	Float64
5	Dependents	0.23101	0	0.0	1	0	Int64
6	Zip Code	0.26537	0.0	0.25	1.0	0	Float64
7	Total Customer Svc Requests	-1.61418e-16	-0.935899	-0.236828	5.35573	0	Float64
8	Product/Service Issues Reported	-4.84255e-17	-0.42941	-0.42941	7.9328	0	Float64
9	Offer_Offer E	0.114298	0	0.0	1	0	Int64
10	Internet Type_Cable	0.219509	0	0.0	1	0	Int64
11	Internet Type_None	0.216669	0	0.0	1	0	Int64
12	Contract_Month-to-Month	0.512566	0	1.0	1	0	Int64
13	Contract_Two Year	0.267358	0	0.0	1	0	Int64
14	Payment Method_Credit Card	0.390317	0	0.0	1	0	Int64

Fig. 9. Approach 4 Inputs summary

5.4 Results

Once again, we will examine the same hyperparameters configurations for the four types of machine learning models discussed in this project: Artificial Neural Networks, Support Vector Machines, Decision Trees and k-Nearest Neighbors. Two Ensembling techniques will also be taken into acount.

ANNs:

The 8 ANN architectures share the following characteristics: 10 separate executions per fold, a 20% validation set ratio, a maximum of 15 cycles without improvement in validation error and a minimum training loss value of 0. The loss function is binary crossentropy and the optimizer used is ADAM. The results are presented in Table 16:

Model	Topology	Learning Rate	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	[5]	0.02	0.9208	0.0083	0.8309	0.0183
2	[10]	0.02	0.9202	0.0080	0.8272	0.0194
3	[40]	0.01	0.9213	0.0081	0.8275	0.0196
4	[5, 5]	0.02	0.9240	0.0077	0.8361	0.0206
5	[20, 20]	0.01	0.9218	0.0078	0.8322	0.0191
6	[50, 50]	0.01	0.9215	0.0078	0.8296	0.0192
7	[50, 40]	0.01	0.9217	0.0081	0.8305	0.0195
8	[100, 100]	0.01	0.9215	0.0084	0.8283	0.0193

Table 16. ANN Models from Fourth Approach

As in the previous approaches, the results obtained across the different architectures are all highly competitive, exceeding 92% accuracy, with very little variation across architectures. Nonetheless, we can observe that including 2 hidden layers of neurons achieves slightly better performance in terms of accuracy.

SVMs:

For the SVM models, we again include the eight configurations obtained by varying the kernel types, gamma values and C values. The results are summarized in Table 17:

Model	Kernel	Degree	γ_{Kernel}	C	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	linear	-	-	0.1	0.9215	0.0080	0.8613	0.0741
2	rbf	-	auto	0.1	0.9098	0.0091	0.8154	0.1138
3	rbf	-	scale	1.0	0.9253	0.0077	0.8634	0.0766
4	rbf	-	1.0	1.0	0.8976	0.0061	0.8435	0.1330
5	sigmoid	-	-	1.0	0.7346	0.0003	1.0000	0.0000
6	poly	1	auto	10.0	0.9206	0.0075	0.8646	0.0695
7	poly	2	scale	1.0	0.9218	0.0083	0.8432	0.0919
8	poly	3	2.0	0.1	0.9115	0.0090	0.8596	0.0695

Table 17. SVM Models from Fourth Approach

Once again, it is evident that the choice of different parameters used to adjust the kernel has a significant impact on the results obtained by the SVM models, as we observe considerable variability in the metric values across the eight models. Most models perform quite well; for linear, RBF and polynomial kernels, there are configurations that achieve accuracies above 92%, similar to those obtained by the ANNs.

In the previous three approaches, we observed bias issues toward positive predictions in models 4 and 5. Notably, in this fourth approach, this problem persists only in model 5 with the sigmoid kernel. As for model 4, which uses an RBF kernel and $\gamma = 1$, the sensitivity has dropped from the usual 99% to 78.84%. The elimination of bias has allowed the accuracy of this model to improve significantly compared to the previous approaches, where it remained between 73% and 75%, now reaching 89.76%.

Decision Trees:

For Decision Trees, 8 different maximum depths have been evaluated. The results are presented in Table 18.

Model	Max Depth	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	2	0.8370	0.0075	0.7731	0.2490
2	4	0.8783	0.0090	0.7819	0.1347
3	8	0.9019	0.0070	0.8611	0.0920
4	16	0.8922	0.0076	0.8413	0.0777
5	32	0.8891	0.0059	0.8398	0.0707
6	64	0.8891	0.0059	0.8398	0.0707

Table 18. Decision Tree Models from Fourth Approach

As expected, a greater maximum depth allows the models to make more informed decisions that take into account more features, thus improving both accuracy and sensitivity as the depth increases. However, as observed in previous approaches, at some point between 8 and 16, the model begins to take into account overly specific conditions, leading to overfitting issues. A clear indication

of this is that the accuracy achieved with 8 layers is not improved by the values obtained with 18, 32 and 64 layers. The results exceed 90% in accuracy and 86% in sensitivity before overfitting occurs.

Once again, since the results for maximum depths of 16 and 32 are identical, we can infer that some features have no impact on the predictions made by the Decision Trees, following the reasoning explained in approach 1.

kNNs:

Finally, Table 19 presents the results obtained after applying the kNN algorithm for the different k values already used in previous approaches.

Model	**k Value**	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
1	2	0.8833	0.0086	0.7777	0.1722
2	3	0.8978	0.0089	0.8436	0.0983
3	5	0.9002	0.0097	0.8440	0.1031
4	10	0.9029	0.0094	0.8241	0.1302
5	15	0.8998	0.0074	0.8352	0.1139
6	30	0.9005	0.0112	0.8522	0.1262

Table 19. kNN Models from Fourth Approach

As in previous approaches, beyond a certain value of K, in this case approximately $k = 10$, the accuracies begin to decrease because the algorithm starts considering patterns that are too distant from each other. The highest recorded accuracy was 90.29%.

Ensemble Models:

Let us now evaluate whether it is worth increasing the complexity of our model by combining several estimators using the ensemble techniques of Soft Voting and Stacking: The first assigns higher weights to the predictions of the most accurate models, while the second relies on a new SVM meta-model that takes the predictions of the other estimators as inputs.

The four models that achieved the highest individual accuracy scores are retrieved for ensembling. These are listed below and the results of the Ensembles are summarized in Table 20.

(1) Model: SVM - 3 (Acc: 0.9253; Sens: 0.8634)

(2) Model: ANN - 4 (Acc: 0.9240; Sens: 0.8361)

(3) Model: ANN - 5 (Acc: 0.9218; Sens: 0.8322)

(4) Model: SVM - 7 (Acc: 0.9218; Sens: 0.8432)

Model	$\mu_{Accuracy}$	$\sigma_{Accuracy}$	$\mu_{Sensitivity}$	$\sigma_{Sensitivity}$
Soft Voting	0.9249	0.0087	0.8224	0.0138
Stacking	0.9267	0.0090	0.8320	0.0173

Table 20. Ensemble Models from Fourth Approach

The only ensembling method that managed to improve the accuracy achieved by SVM-3 on its own was Stacking. However, the improvement is only 0.14% and is accompanied by a sensitivity that is 4% lower. We conclude that the added complexity is not justified. As for Soft Voting, its accuracy score does not surpass that of SVM-3 and it is therefore automatically discarded.

5.5 Discussion

We will begin this discussion of the results of the fourth approach by evaluating the overall performance of the four models. Then, since approach 2 has obtained the best results until this point while considering all available features, we will compare the results obtained in these two approaches to illustrate how much information was lost during dimensionality reduction.

We will also compare the results of approaches 3 and 4, as both focus on dimensionality reduction of the dataset, to determine which is more effective.

Evaluation of model performance:

After testing the implementation of 8 different architectures, we observed that ANNs are highly efficient models for this approach. Architectures with two hidden layers are slightly favored, but beyond that, all models exceeded 92% accuracy and 82.75% sensitivity.

SVMs have also proven to be very effective, though their results vary much more depending on hyperparameter tuning compared to ANNs. Regardless, among the SVMs, we found the model with the highest score in both metrics: SVM-3, with an RBF kernel, gamma set to scale and $C = 1$, achieved 92.53% accuracy and 86.34% sensitivity.

Decision Trees and kNN algorithms, while not reaching the scores of ANNs and SVMs, achieved their best performances across all approaches in this final approach. This is partly because these models perform better with a reduced number of variables and this approach involves the lowest-dimensional input. Both models exceeded 90% accuracy with their best configurations. However, due to better sensitivity scores, Decision Trees are preferred. Finally, as discussed earlier, it is important to consider overfitting issues in Decision Trees and excesive k values in kNN when tuning hyperparameters. Excessive values for maximum depth and k lead to poorer model performance.

Evaluation of dimensionality reduction by feature selection:

At this point, let us assess whether the dimensionality reduction carried out through correlation analysis to identify the variables most associated with customer churn has been effective. Regarding the number of features considered, we reduced the number of 42 features in approach 2 to 14. Computationally, this is highly beneficial for the feasibility of the models, but now we will evaluate whether this reduction sacrificed too much information. Let us compare the results of approach 4 with those of approach 2, which has achieved the best results up to this point using all original features from the dataset:

ANNs have demonstrated high accuracy scores that remain stable despite changes in architecture. Therefore, their performance serves as a good comparison between the two approaches. The highest accuracy achieved by an ANN in approach 2 was 93.42%, while in approach 4, the most accurate ANN achieved 92.4%. This represents only a one percentage point difference in already remarkable accuracy values.

For SVMs, whose optimal configurations have been the most accurate in both approaches, we observe even closer results than with ANNs. The most accurate model in the second approach was SVM-7, with 93.44%, while in the fourth approach, it was SVM-3, with 92.53%. The difference is less than 1%. In fact, for configurations 4 and 8, accuracy results in approach 4 are the better ones.

It is also noteworthy that feature selection eliminated the bias toward positive predictions observed in SVM-4 configurations. This is likely because the removal of certain variables reduced the noise in the data that was causing this tendency.

Decision Trees performed similarly in both approaches, achieving around 90% accuracy for optimal depth values, although model sensitivity was slightly better in the approach that used all variables.

For kNN, dimensionality reduction improved accuracy, with a score of 90.3% for optimal k values compared to 89.1% in approach 2.

Overall, we highly value the dimensionality reduction of the dataset through feature selection based on correlation analysis, as the performance of the models does not reflect significant information loss: ANNs and SVMs are only 1% less accurate, Decision Trees yield similar results and kNNs improve their performance. From this, several conclusions can be drawn:

- Many features in the dataset were not relevant to a customer's decision to leave the company.
- The relationships between the dataset variables and customer churn are primarily linear, as the correlation analysis considers only associations of this type.
- The one-hot encoding of categorical variables, which transforms them into multiple binary variables, proves to be highly effective. This is because it highlights the subset of patterns that are truly critical. For instance, out of the six categories in the *Offer* variable, only the binary feature *Offer_E* was considered in this approach.

Comparison between the two dimensionality reduction techniques:

In this final section, we compare the dimensionality reduction techniques used in this project: PCA in approach 3 and feature selection through correlation analysis in approach 4.

As in the previous comparison, we will focus on the differences between the most optimal configurations for each of the four types of models. we observe that approach 4 achieves between 2% and 5% higher accuracies than approach 3, with sensitivity values also at least 2% higher. This indicates that the approach 4 method retained more information after the dimensionality reduction.

In addition to this, the approach 4 method offers further advantages. First, the dimensionality reduction is greater, as its input consists of 14 variables compared to the 21 principal components used with PCA. Second, it is easily interpretable and allows the company to develop precise diagnostics and strategies, as the variables remain recognizable, unlike the principal components. Finally, PCA does not guarantee the elimination of potential noise in the dataset, as its components are linear combinations of the original features, meaning any inconsistencies in the original data will still affect the results.

The only advantage of PCA in this case is that it requires much less complex and labor-intensive preprocessing compared to the process carried out in approach 4.

6 Final Discussion

This section provides a general evaluation of the results across models and approaches based on the selected metrics of accuracy and sensitivity.

6.1 Evaluation of model performances

First, we will analyze the performance of the four types of machine learning models considered: ANNs, SVMs, Decision Trees and kNNs and the two ensembling techniques that merge them.

- Overall, the results have been satisfactory: the best configurations of each model type did not fall below 86% accuracy in any of the approaches (with the exception of the two SVMs that exhibited bias issues).
- Throughout the four approaches, the best-performing models have been the ANNs and SVMs. The general trend is that ANNs achieve very similar results in both accuracy and sensitivity across different architectures, while the performance of SVMs varies significantly with different hyperparameter configurations. The SVMs with the most optimal configurations have consistently outperformed the ANNs in both metrics across all approaches.

  The model with the highest accuracy in the first two approaches, which used higher-dimensional inputs, was SVM-7, configured with a polynomial kernel of degree 2, $C = 1, \gamma : scale$.

  In the last two approaches, which used reduced-dimensional inputs, the model with the highest accuracy and sensitivity was SVM-3, configured with an RBF kernel, $C = 1, \gamma : scale$.
- Decision Trees and kNNs performed a step under ANNs and SVMs in both metrics. In no approach did the best configurations of these models rank among the top 4 most accurate. However, their results are still satisfactory, considering their lower computational cost. Additionally, due to their higher explainability compared to ANNs and SVMs, they have provided valuable insights with their behaviours.
- Ensemble models have not been effective. In none of the four approaches did the added complexity of these techniques translate into significant improvements in either metric.
- Finally, the standard deviations of the metric results across the folds have remained low. This has been particularly true for ANNs and SVMs, where deviations rarely exceeded 0.15. For Decision Trees and kNNs, which are slightly less stable models and more dependent on the distribution of each fold, deviations did not exceed 0.3. Furthermore, for higher values of maximum depth and k, the standard deviation was lower, as the specific characteristics of the fold start having less influence. We conclude that the models generalize well and that the data provided sufficient representation of different patterns across all folds. This suggests that the models are reliable and robust.

6.2 Tracking of results obtained across the approaches

In this section, we evaluate how the models performed under the different approaches applied to the same dataset. The first two approaches focused on preprocessing techniques, while the last two focused on dimensionality reduction.

- The advanced preprocessing techniques employed in approach 2 significantly improved the results of approach 1. In addition to incorporating new information, the inclusion of proportion encoding allowed introducing numerical relationships between categorical variables and the target feature. This resulted in all models achieving higher accuracy and sensitivity scores.
- PCA successfully reduced the dataset's dimensionality by half while retaining most of the information. Through a simple and automated transformation of the inputs from approach 2, the number of features was reduced from 42 to 21, with only a 2.2% drop in accuracy for the most optimal models between the two approaches.
- Feature selection through correlation analysis with the target feature succesfully combined dimensionality reduction with information retention. In approach 4, the input dimensionality was reduced to 14 features and the models performances were only slightly lower than that of approach 2, which achieved the best overall results but relied on 42 features.

6.3 Final Model Selection

After completing the analysis of models performances across the four approaches using cross-validation, it is time to determine which model and preprocessing approach will be selected for the final training phase.

It has been determined that approach 4 achieved excellent results in both dimensionality reduction and information retention. Therefore, prioritizing computational efficiency over a slight performance improvement, this is the approach that has been selected.

The model with the best accuracy performance was SVM-3, which uses an RBF kernel with a gamma value of "scale" and a C parameter equal to 1. Following cross-validation, its accuracy score was 92.53% and its sensitivity score was 86.34%.

Referring back to the interpretation of metrics explained in Section 1.5, the high accuracy value ensures that we will have a model with great reliability in predicting the future of the customers, wether is leaving or staying. The high sensitivity score gives us confidence that this model will be highly effective at identifying customers with intentions to leave the company, which is crucial for customer retention strategies.

7 Final Model Training

To carry out the final training, the dataset is split using the hold-out method, reserving 90% of the data for training and the remaining 10% for testing, in order to evaluate the model's performance objectively. The confusion matrix results obtained with the test data are presented in the table 21.

	Prediction Negative	**Prediction Positive**
Real Negative	489	22
Real Positive	38	155

Table 21. Confusion matrix on test data of SVM-3 with Approach 4 preprocessing

The SVM-3 model with Approach 4 preprocessing has demonstrated good performance on the test data. The accuracy achieved is 0.9148, indicating that the model correctly classifies approximately 91.48% of the cases. Additionally, the sensitivity is 0.8031, showing that it accurately identifies about 80.31% of positive cases.

These results highlight the strength of the chosen preprocessing method and model configuration, achieving a solid balance between overall classification performance and the detection of positive cases. This performance validates the choice of SVM-3 and Approach 4 as an effective combination for this problem.

8 Conclusion

In this project, addressing a widely studied problem of great interest to various companies, proper preprocessing can make the difference in accurately predicting whether a customer decides to leave. Additionally, computational cost has been significantly reduced thanks to the dimensionality reduction techniques developed throughout this project, while still guaranteeing good results with minimal loss of accuracy in predicting customer churn.

In our case and with the dataset used, the best performances were undoubtedly achieved by ANNs and SVMs. However, this does not necessarily mean they are inherently better than kNN or Decision Trees, as these models have their own advantages, such as lower computational complexity. Also, depending on the characteristics of the dataset, they could even outperform ANNs and SVMs.

9 Future Work

After evaluating and testing the final model, the next step is to proceed with training the model without using a separate test set. This maximizes the amount of data available for training, allowing the model to learn from a larger dataset.

Although this only increases the training data by the 10% that was reserved for testing in the previous training, this additional data helps to improve the model's ability to generalize, thereby enhancing its performance when applied to future, unseen data. This final training step ensures that the model is as robust and effective as possible when deployed in real-world scenarios by the JB Link telecommunications company.

References

[1] Ibrahim Adedeji Adeniran, Christianah Pelumi Efunniyi, Olajide Soji Osundare, and Angela Omozele Abhulimen. 2024. Implementing machine learning techniques for customer retention and churn prediction in telecommunications. *Computer Science amp; IT Research Journal* 5, 8 (Aug. 2024), 2011–2025. https://doi.org/10.51594/csitrj.v5i8.1489

[2] Annalisa Barsotti, Gabriele Gianini, Corrado Mio, Jianyi Lin, Himanshi Babbar, Aman Singh, Dr. Fatma Taher, and Ernesto Damiani. 2024. A Decade of Churn Prediction Techniques in the TelCo Domain: A Survey. *SN Computer Science* 5 (04 2024). https://doi.org/10.1007/s42979-024-02722-7

[3] T. Vafeiadis, K.I. Diamantaras, G. Sarigiannidis, and K.Ch. Chatzisavvas. 2015. A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory* 55 (2015), 1–9. https://doi.org/10.1016/j.simpat.2015.03.003