

**T.C.  
BAHÇEŞEHİR UNIVERSITY**



**FACULTY OF ENGINEERING AND NATURAL SCIENCES**

**CAPSTONE PROJECT FINAL REPORT**

**HANDLING CLASS IMBALANCE PROBLEM IN CHURN PREDICTION**

**Erim Akyol**

**Betul Keten**

**Hasan Mert Sezer**

**Lara Turunc**

**Dogukan Yagci**

**Faculty of Engineering and Natural Sciences**

**Advisor: Assist. Prof. M. Aslı Aydın, Management Engineering**

**Advisor: Assist. Prof. Ömer Melih Gül, Computer Engineering**

**ISTANBUL, Jan 2024**

### **STUDENT DECLARATION**

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

- They have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
- They have not received unpermitted aid for the project design, construction, report or presentation;
- They have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

## **ABSTRACT**

### **HANDLING CLASS IMBALANCE PROBLEM IN CHURN PREDICTION**

Erim Akyol

Betül Keten

Hasan Mert Sezer

Lara Turunç

Doğukan Yağcı

Faculty of Engineering and Natural Sciences

Advisor: M. Aslı Aydın, Ph.D, Management Engineering

Advisor: Assist. Prof. Ömer Melih Gül, Computer Engineering

Jan 2024

This capstone project aims to tackle the class imbalance problem in churn prediction within the telecommunications industry. The project's goal is to develop accurate models for predicting customer churn and to create effective strategies for customer retention.

Class imbalance in machine learning, where one class is significantly larger than others in a dataset, can lead to inaccurate predictions. To address this, the project uses resampling techniques such as

oversampling and undersampling to balance the data, improving the accuracy of churn predictions. Then we train and evaluate various machine learning classifiers to find the most effective method for predicting customer churn.

A primary goal of the project is to evaluate the effectiveness of different resampling methods in managing class imbalance and to assess the performance of various machine learning classifiers in churn prediction. The project involves collaboration between different departments: the management engineering team works on implementing machine learning algorithms and analyzing the results to develop targeted promotional and campaign strategies, while the computer engineering department focuses on data preprocessing and solving the class imbalance issue.

This project combines technical skills and strategic planning to improve customer retention in the telecommunications sector by identifying potential churners and applying well-informed strategies.

**Key Words:** Data analysis, algorithm, machine learning, customer type, segmentation, marketing strategy, digital marketing, customer relationship management, telecommunication industry, churn data, imbalanced data.

## TABLE OF CONTENTS

<b>ABSTRACT .....</b>	<b>iii</b>
<b>TABLE OF CONTENTS .....</b>	<b>vi</b>
<b>LIST OF ABBREVIATIONS.....</b>	<b>viii</b>
<b>1. OVERVIEW .....</b>	<b>1</b>
<b>1.1. Identification of the Need .....</b>	<b>2</b>
<b>1.2 Definition of the Problem.....</b>	<b>3</b>
1.2.1. Functional Requirements.....	4
<b>1.3. Conceptual solutions.....</b>	<b>6</b>
1.3.1. Literature Review .....	7
1.3.2. Concepts .....	9
<b>1.4. Physical architecture .....</b>	<b>10</b>
<b>2. WORK PLAN .....</b>	<b>12</b>
<b>2.1. Work Breakdown Structure (WBS).....</b>	<b>12</b>
<b>2.2. Responsibility Matrix (RM) .....</b>	<b>12</b>
<b>2.3. Project Network (PN) .....</b>	<b>13</b>
<b>2.4. Gantt chart .....</b>	<b>13</b>
<b>2.5. Costs.....</b>	<b>14</b>
<b>2.6. Risk Assessment.....</b>	<b>14</b>
<b>3. SUB-SYSTEMS.....</b>	<b>15</b>
3.1.3. Data collection and analysis .....	19
<b>3.2. Computer Engineering.....</b>	<b>30</b>
3.2.1. Requirements.....	30
3.2.2. Technologies and Methods.....	31
3.2.3. Conceptualization .....	31
3.2.4. Physical architecture.....	31
3.2.5. Materialization .....	32
3.2.6. Evaluation .....	41

<b>4. INTEGRATION AND EVALUATION .....</b>	<b>42</b>
<b>4.1. Integration.....</b>	<b>42</b>
<b>4.2. Evaluation .....</b>	<b>43</b>
<b>5. SUMMARY AND CONCLUSION .....</b>	<b>45</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>46</b>
<b>REFERENCES.....</b>	<b>47</b>
<b>APPENDIX .....</b>	<b>48</b>

Figure 1 Physical Architecture .....	10
Figure 2 Process Chart .....	11
Figure 3 Work breakdown structure for the project. ....	12
Figure 4 The project network. ....	13
Figure 5 Gantt Chart .....	14
Figure 6 Physical Architecture Model (Adapted from Cleary et al., 1996).....	32
Figure 7 Proof of Imbalanced Data .....	34
Figure 8 Five Factors That Effect Churn The Most .....	35
Table 1 Comparison of the three conceptual solutions. ....	10
Table 2 Responsibility Matrix for the team .....	13
Table 3 Risk Matrix .....	14
Table 4 Risk Assessment .....	14
Table 5 Results of Model 1 .....	36
Table 6 Model 1 Preprocessing Overview .....	37
Table 7 CatBoost Classifier Metrics .....	38
Table 8 Model 3 Overview .....	40
Table 9 Model 4 Overview .....	41

## **LIST OF ABBREVIATIONS**

ML	Machine Learning
----	------------------



## 1. OVERVIEW

This capstone project represents an innovative endeavor in the field of telecommunications, focusing on the crucial issue of customer churn—a phenomenon where customers cease their association with a service. The core objective of our project is twofold: firstly, to address the class imbalance problem inherent in churn data, and secondly, to accurately predict potential churners. This enables us to target the right customers with retention strategies, thereby reducing churn rates effectively. Our team, comprising members from the Computer Engineering and Management Engineering departments, collaborated to develop a sophisticated solution that leverages machine learning to tackle this issue effectively.

The Computer Engineering team was primarily responsible for the technical aspects of the project. Their main tasks involved data preprocessing, which is essential for ensuring the quality and reliability of the data used in machine learning models. This team also took on the challenge of handling the class imbalance problem, a common issue in machine learning, especially in datasets like customer churn. Additionally, they were tasked with implementing and training various machine learning models, which are the backbone of the project's analytical capabilities. In this context, Lara preprocessed the data and trained different models in order to reach high accuracy. Hasan Mert made research about the topic.

On the other side, the Management Engineering department focused on the application of the insights derived from the machine learning models. Their key responsibilities included analyzing the results from the machine learning models and making informed decisions based on these analyses. They were in charge of designing targeted promotional and campaign strategies aimed at retaining customers identified as potential churners by the models. This strategic planning was crucial as it translated the project's technical findings into practical business solutions. Betul trained an ML model with Lara and analyzed the outcomes of the machine learning models and spearheaded the development of effective customer retention strategies, bridging the gap between data-driven insights and actionable business tactics.

As the project evolved during the second semester, we updated our goals to reflect a more nuanced understanding of the challenges and opportunities in churn prediction. Initially, our focus was predominantly on model accuracy. However, as we delved deeper, we realized the importance of balancing precision and recall, especially in the context of imbalanced classes. This shift in focus led to a more comprehensive approach in our modeling techniques and the strategies we developed for customer retention.

Overall, this project stands as a testament to the power of interdisciplinary collaboration, combining the technical prowess of computer engineering with the strategic insight of management

engineering to address a real-world business challenge.

### **1.1. Identification of the Need**

The telecommunications industry is highly competitive, and customer churn – the turnover of customers switching from one service provider to another – presents a significant challenge. Our project addresses this key issue by developing a machine learning-based solution for predicting potential churners with greater accuracy. This predictive capability is crucial for telecom companies, as it allows them to identify at-risk customers and proactively engage them with retention strategies, ultimately reducing churn rates.

The primary users of our product are telecommunications companies, particularly those facing high churn rates and looking to enhance their customer retention strategies. By implementing our solution, these companies can gain valuable insights into customer behavior, identify patterns leading to churn, and tailor their services to address the needs and preferences of their customers more effectively.

Our solution addresses the need for a more nuanced understanding of customer churn. Traditional churn prediction models often struggle with class imbalance, where the number of customers who stay with the company vastly outnumbers those who leave. This imbalance can skew model predictions, leading to less effective retention strategies. By focusing on balancing the data and improving prediction accuracy, our solution ensures that telecom companies can target their retention efforts more effectively, potentially saving significant costs associated with losing and acquiring customers.

The estimated impact of our solution is substantial. According to a study by Mothilal, Bhatt, and Sharma (2020), "predictive analytics in customer churn can improve retention rates by up to 15% in the telecommunications sector." This improvement in retention rates can translate into millions of dollars in saved revenue for large telecom companies. Beyond financial benefits, our solution also aids in fostering customer loyalty and satisfaction, as targeted retention strategies are likely to resonate more with customers' specific needs and preferences.

In summary, the value of our product lies in its ability to make churn prediction more accurate and actionable. By leveraging advanced machine learning techniques to tackle the class imbalance problem, our solution offers telecom companies a powerful tool to enhance their customer retention strategies, reduce churn rates, and maintain a competitive edge in the market.

## 1.2 Definition of the Problem

The definition of the problem in addressing class imbalance in data preprocessing and accurately predicting customer churn involves understanding the challenges associated with imbalanced classes and the importance of identifying customer losers for effective customer retention strategies.

Class imbalance refers to a situation where the distribution of classes in a dataset is uneven, with one class being significantly underrepresented compared to the other class(es). This imbalance can occur in various domains, such as fraud detection, medical diagnosis, or customer churn prediction. In the context of implementing machine learning algorithms, class imbalance poses challenges in model training and performance evaluation.

When dealing with class imbalance in data preprocessing, the goal is to ensure that the minority class, which often represents the target or critical class of interest, receives sufficient representation in the dataset. This is crucial to avoid biased models that favor the majority class and yield inaccurate predictions for the minority class. The problem definition entails recognizing the need to address class imbalance and adopt appropriate strategies to mitigate its impact.

Additionally, accurately predicting customer losers plays a vital role in customer retention efforts. Customer churn refers to the phenomenon where customers discontinue their relationship with a company or stop using its products or services. Identifying customer losers, or those customers who are at a higher risk of churn, is crucial for businesses to proactively take actions and implement retention strategies.

The problem of accurately predicting customer losers involves leveraging historical customer data and relevant features to build machine learning models that can identify patterns and indicators of potential churn. By accurately identifying customer losers, businesses can focus their resources and interventions on retaining these customers, thereby improving customer satisfaction and reducing churn rates.

The two-pronged purpose of tackling class imbalance and accurately predicting customer losers in hash data revolves around improving the effectiveness of machine learning models and optimizing customer retention strategies. By defining the problem and recognizing the significance of addressing class imbalance and identifying customer losers, businesses can tailor their data preprocessing techniques and deploy suitable machine learning algorithms to achieve accurate predictions and better retention outcomes.

In summary, the definition of the problem in addressing class imbalance in data preprocessing and accurately predicting customer losers involves recognizing the challenges posed by imbalanced classes and understanding the importance of identifying customers at risk of churn. By addressing class imbalance and accurately predicting customer losers, businesses can enhance the performance of their machine learning models and implement effective customer retention strategies to optimize their overall business outcomes.

### 1.2.1. Functional Requirements

In order to effectively address the challenge of customer churn in the telecommunications industry, our product – a machine learning-based churn prediction system – is designed with specific functional requirements. These requirements are articulated through precise technical statements, ensuring clarity and focus in the system's capabilities. Key functionalities include:

- **Accurately Predict Churn:** The service should be able to identify potential churners with high accuracy. This involves the ability to process large datasets and recognize complex patterns indicative of churn behavior.
- **Handle Class Imbalance:** The process should effectively manage the class imbalance inherent in churn datasets, where the number of non-churners significantly outweighs the number of churners. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) or RandomUnderSampler should be employed to balance the dataset for training.
- **Provide Actionable Insights:** The product must not only predict churn but also provide insights that can be translated into actionable retention strategies. This involves identifying key factors contributing to churn and suggesting targeted interventions.
- **Scalability and Efficiency:** The system should be scalable, capable of handling increasing volumes of data without a corresponding increase in prediction latency. Efficiency in processing and model training is essential for real-time applications.

By meeting these technical requirements, the churn prediction system we have developed aims to provide a comprehensive solution to the telecommunications industry, helping companies reduce churn rates and enhance customer retention strategies.

### 1.2.2. Performance requirements

The performance requirements for our churn prediction model are critical to its efficacy and utility in a real-world telecommunications context. These requirements are designed to be measurable and quantifiable, ensuring that the model not only fulfills its intended function but does so with a high degree of accuracy and efficiency. The following are the key performance parameters that will be used to measure the success of our model:

- The model should achieve a high level of accuracy in predicting customer churn. A target accuracy rate of at least 85% is desired, ensuring that the majority of predictions are correct and reliable.
- The model should maintain a balance between precision and recall. A precision rate of at least 80% is targeted, ensuring that when the model predicts churn, it is correct most of the time. Similarly, a recall rate of at least 75% is desired, indicating the model's ability to correctly identify actual churn cases.
- The F1 Score, which is the harmonic mean of precision and recall, should be at least 80%. This metric will help in assessing the balance between precision and recall, which is crucial in the context of an imbalanced dataset.
- The model should be efficient in terms of processing time. For large datasets typically used in telecommunications, the entire process of data preprocessing, training, and prediction should not exceed 60 minutes, ensuring timeliness in practical applications.
- The model should be scalable in handling datasets of varying sizes without a significant loss in performance. It should maintain its accuracy and efficiency regardless of the increase in data volume.
- The model should exhibit robustness to variations in data. Its performance should not be significantly impacted by minor changes in data trends or customer behaviors over time.
- The model should optimize the use of computational resources. Memory usage and CPU utilization should be within reasonable limits to ensure that the model can be deployed in various technological environments without requiring excessive

computational power.

### **1.2.3. Constraints**

Our project, focusing on developing a churn prediction model for the telecommunications industry, operates under several key constraints:

- Our team comprises members from Computer and Management Engineering disciplines, each bringing unique skills to the project. However, our capacity to explore certain complex algorithms or conduct extensive testing is limited by the team size and the varied expertise of members. Additionally, the project timeline is constrained to a semester, which requires efficient time management and prioritization of essential tasks over exploratory ones.
- Financial resources are limited, restricting our ability to access certain proprietary datasets or advanced computational resources. This budget constraint necessitates a focus on open-source tools and publicly available datasets.
- Adherence to data privacy and ethical standards, particularly GDPR and other relevant legal frameworks, is critical. This necessitates careful handling of customer data and potentially limits the extent of data utilization.
- Business constraints include aligning our model's capabilities with industry needs without overengineering. Environmentally, we aim to minimize our computational resource usage to reduce energy consumption. Ethically, it's imperative to avoid bias in our models, ensuring fairness and transparency in churn predictions. Health and safety are less directly applicable but are considered in terms of the digital safety and security of the model and data.

## **1.3. Conceptual solutions**

Addressing the challenge of customer churn in the telecommunications sector through predictive modeling involves both a thorough review of existing literature and the development of our own innovative solutions.

Literature in this domain underscores the effectiveness of machine learning in predicting customer behavior. Various algorithms have been studied, ranging from logistic regression and decision trees to advanced models like neural networks and ensemble methods. A recurring theme is tackling the class imbalance prevalent in datasets. For instance, K. T. Smith et al. (2019) advocate for resampling methods like SMOTE (Synthetic Minority Over-

sampling Technique) to balance the dataset. The literature also frequently points to the importance of feature selection and engineering for enhancing model performance.

Based on these insights, the following conceptual solutions can be considered:

- **Resampling Techniques:** Implementing both oversampling and undersampling methods, such as SMOTE, addresses the class imbalance issue, potentially leading to more accurate churn predictions.
- **Hybrid Model Approach:** A combination of various machine learning algorithms could be explored. This might include integrating models like Random Forest, LightGBM, and CatBoost with traditional algorithms such as logistic regression to find an optimal blend for specific datasets.
- **Feature Engineering:** Identifying and engineering relevant features is critical. This involves analyzing customer usage patterns, demographic data, and historical account information to derive meaningful predictors of churn.
- **Model Evaluation Metrics:** Evaluating the model's effectiveness should include metrics like accuracy, precision, recall, and F1 score. These metrics are crucial for balancing the identification of actual churners against the implications of false positives.
- **User Interface Development:** A user-friendly interface for interacting with the model is important. Such an interface would facilitate easy data input, model selection, and interpretation of predictions.
- **Compliance with Ethical and Legal Standards:** Ensuring adherence to data protection and privacy laws is essential for the ethical handling of customer data.

These conceptual approaches offer a roadmap for developing a churn prediction model in telecommunications. They balance the insights gained from academic research with practical considerations for real-world application.

### 1.3.1. Literature Review

In the telecommunications sector, customer churn is a severe issue that happens more frequently. The cost of retaining current consumers is substantially lower than the cost of acquiring new ones, and

according to the literature, the cost of acquiring new customers must be five times higher (Shumaly vd., 2020).

When mining classes with imbalances, issues fall into six categories:

- 1- Inadequate evaluation metrics
- 2- Missing data
- 3- Relative rarity or relative absence of data
- 4- Data Fragmentation
- 5- The wrong kind of inductive bias
- 6- Noise (Weiss, 2004, as cited in Burez, Van Den Poel, 2008).

The telecom business has trouble keeping customers because of the market's saturation and competition. An efficient model for predicting client attrition is essential to solving this problem. The time available for adopting churn management methods was previously constrained by research that concentrated on forecasting customer churn in the current or upcoming month (Yang et al., 2019). This study suggests a T+2 churn customer prediction model, which gives telecom businesses a one-month opportunity to undertake retention measures by identifying customers who are likely to leave in two months. They use a random forest (RF) classifier to effectively distinguish between churning and non-churning clients, and they use artificial minority oversampling approaches to address the issue of class imbalance. The suggested approach provides a precise and workable solution for telecom firms with a precision ratio of roughly 50% and a recall ratio of roughly 50% (Yang et al., 2019).

Churn prediction for a business is typically a very time-consuming activity, and with so many startups and growing businesses, there is fierce competition to keep clients by offering services that are advantageous to both parties. Predicting the company's true customers is quite tough. In the future, machine learning is proven to be one of the most effective ways to solve issues like churn prediction with improved accuracy and precision thanks to the new concepts and frameworks in the fields of reinforcement learning and deep learning (Lalwani vd., 2022).

Given the foregoing, the churn rate should be used and tracked as a key indicator of a company's health and, more importantly, its future prospects. However, since this industry is one of the most volatile and consequently has a high level of competitiveness, telco companies should thoroughly analyze the turnover rate. Customers frequently switch operators in this market in search of better terms of service, more technologically advanced services that can match their expectations, or—and



this is the worst case—a better customer experience. In any case, it's critical to keep consumers, not least since there aren't many rivals and even a small change can have a big impact on the market sectors it serves. Additionally, a higher churn rate comes at a cost that is anything but insignificant. In light of all of this, we may conclude that an unchecked rise in the churn rate results in a loss of profit, which, if unchecked, can grow significantly over time (Doxee, 2022).

### **1.3.2. Concepts**

**Open-Source Model:** This concept offers a free alternative by utilizing manual deployment techniques and open-source machine learning libraries. It gives customers the opportunity to use already-existing open-source resources and deploy models via manual procedures, providing a cost-effective solution.

**DIY Analytics:** With DYOM tools, this concept provides an affordable solution that enables users to create unique machine learning models and carry out data analysis activities. It offers a practical method for acquiring insights and making data-driven choices without the need for substantial financial resources or specialized technical knowledge.

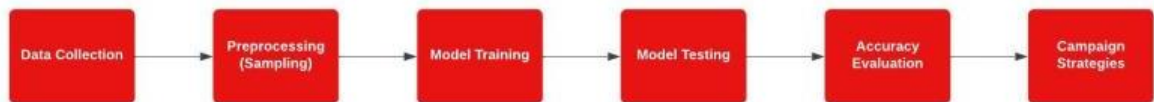
**Azure-Enhanced ML:** In this concept, machine learning procedures are streamlined, and effective model deployment is made possible by using Azure ML Studio. It provides a moderately difficult, cost-effective solution that enables customers to apply cutting-edge machine learning capabilities.

Table 1 compares different conceptual solutions with respect to the four most important requirements; Considering the project's parameters, Open-Source Model emerges as the preferred solution. It satisfies the requirement for a low-cost, low-complexity system while offering sufficient performance for a viable churn prediction model. It aligns with our team's proficiency and our project's financial constraints, allowing for flexibility in model development and deployment. While Azure-Enhanced ML offers tempting features and performance, the moderate complexity and operational expense are less aligned with our current project scope. DIY Analytics is a strong contender, but the need for a more comprehensive feature set and a ready-to-deploy framework gives Open-Source Model the edge as the chosen solution for this project.

**Table 1 Comparison of the three conceptual solutions.**

	Open-Source Model	DIY Analytics	Azure-Enhanced ML
Cost	Free	Affordable	Cost-effective
Complexity	Low	Low	Moderate
Performance	Moderate	High	High
Features	Open-Source ML Libraries, Manual Deployment	DYOM, Data Analysis	Azure ML Studio, Model Deployment

#### 1.4. Physical architecture



**Figure 1 Physical Architecture**

**Data Collection:** This element is responsible for assembling telecommunications churn data from multiple sources such as Kaggle or Colab. It also includes obtaining proper data about customers.

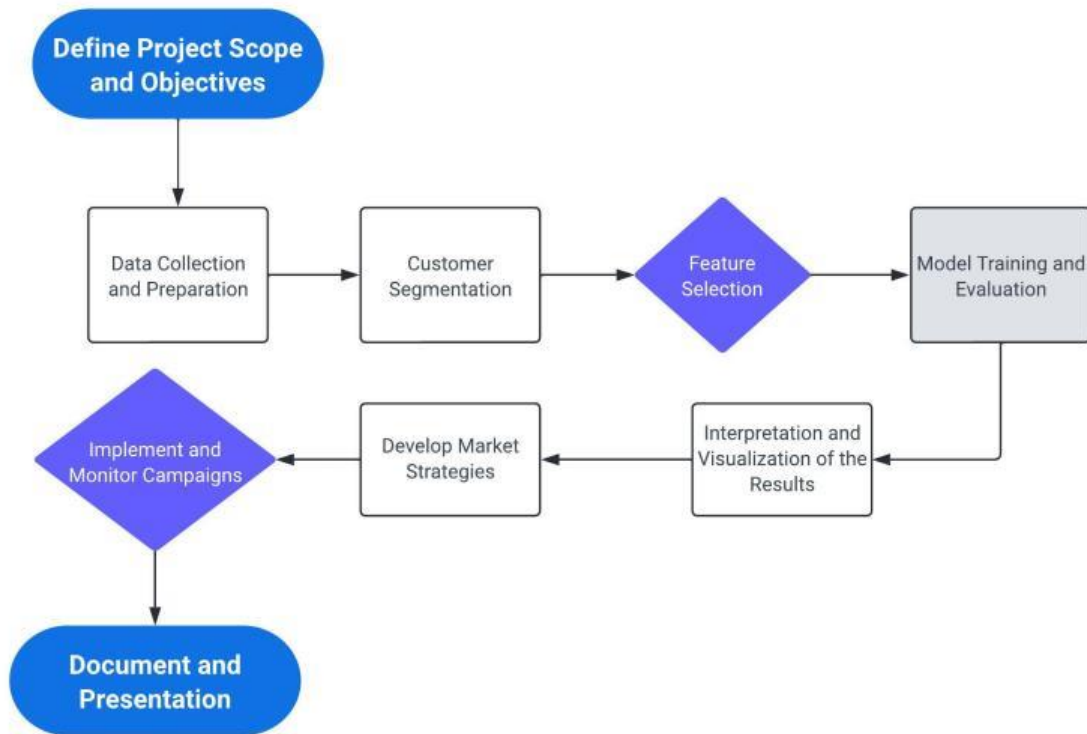
**Preprocessing:** Preprocessing component carries out the responsibilities of data preprocessing, such as sampling methods to correct class imbalance in the churn data.

**Model Training:** This element uses the preprocessed data to create a prediction of a churn model by training ML algorithms.

**Model Testing:** In model testing, the trained churn prediction model's efficiency is being assessed by using different test data.

**Accuracy Evaluation:** This element evaluates the efficiency of the churn prediction model by estimating a variety of accuracy scores and efficiency measures.

**Campaign Strategies:** This element aims to build efficient marketing promotions and campaigns according to the predictions and evaluation results of the churn model.



**Figure 2 Process Chart**

The process chart provides a visual depiction of the workflow by listing the project's consecutive steps. It draws attention to the crucial steps, such as data preparation and analysis, machine learning classifier training and evaluation, and creation of marketing plans based on expected churn data. The diagram acts as a road map, directing the team working on the project through the required steps and ensuring a methodical approach to attaining the project's objectives.

## 2. WORK PLAN

In developing a churn prediction model tailored for the telecommunications industry, our team has crafted a comprehensive work plan to guide the execution phase of the project. This solution, which leverages open-source machine learning libraries for cost-effective and customizable prediction, is the collaborative effort of the Computer Engineering and Management Engineering departments. The project timeline spans one academic semester, during which tasks are meticulously planned and assigned to ensure timely completion.

### 2.1. Work Breakdown Structure (WBS)

In order to schematize the workload and structure, a Work Breakdown Structure (WBS) has been applied. The breakdown of the project work is demonstrated below:

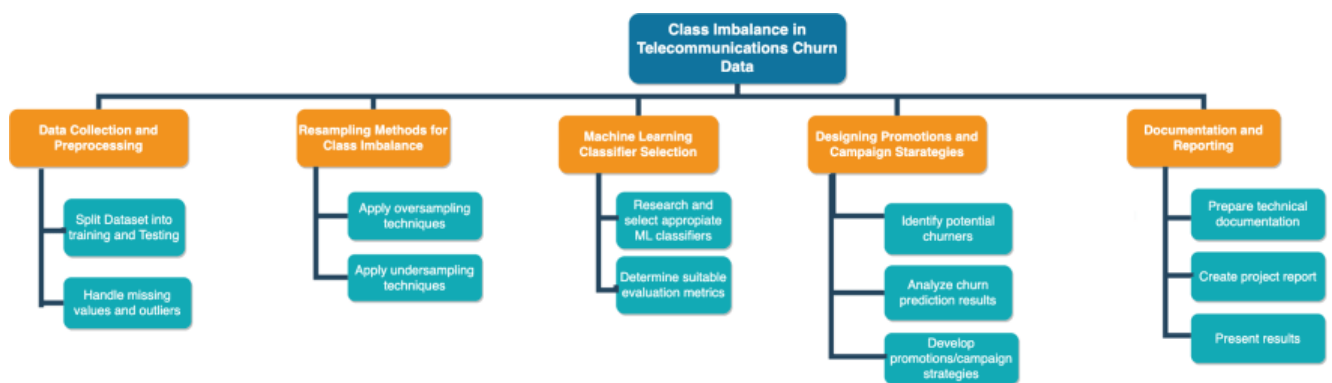


Figure 3 Work breakdown structure for the project.

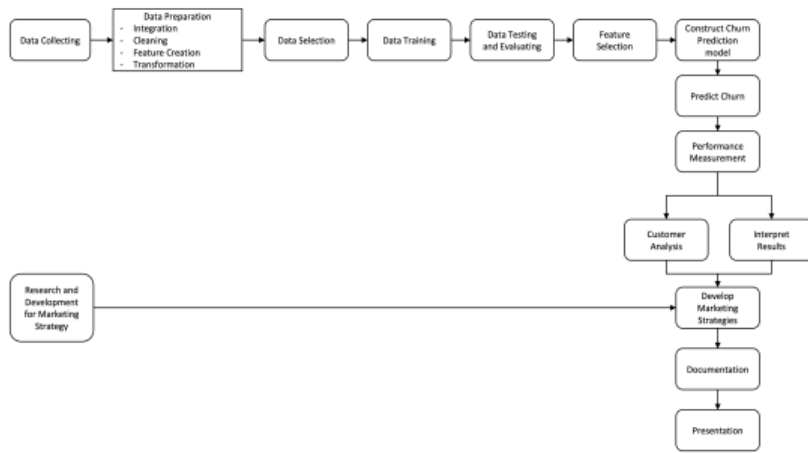
### 2.2. Responsibility Matrix (RM)

TASK	ERİM	BETÜL	DOĞUKAN	LARA	HASAN MERT
Data Preprocessing			S	R	S
Implement ML Algorithms	S	R	S	R	S
Class Imbalance Handling			S	R	S
Analysis of ML Results		R		S	
Decision Making	S	R	S	R	S
Marketing Strategies	S	R		S	
Collecting Data			S	R	S
ML Model Evaluation		R		R	
Documentation and Presentation	R	R	R	R	R

**Table 2 Responsibility Matrix for the team**

### 2.3. Project Network (PN)

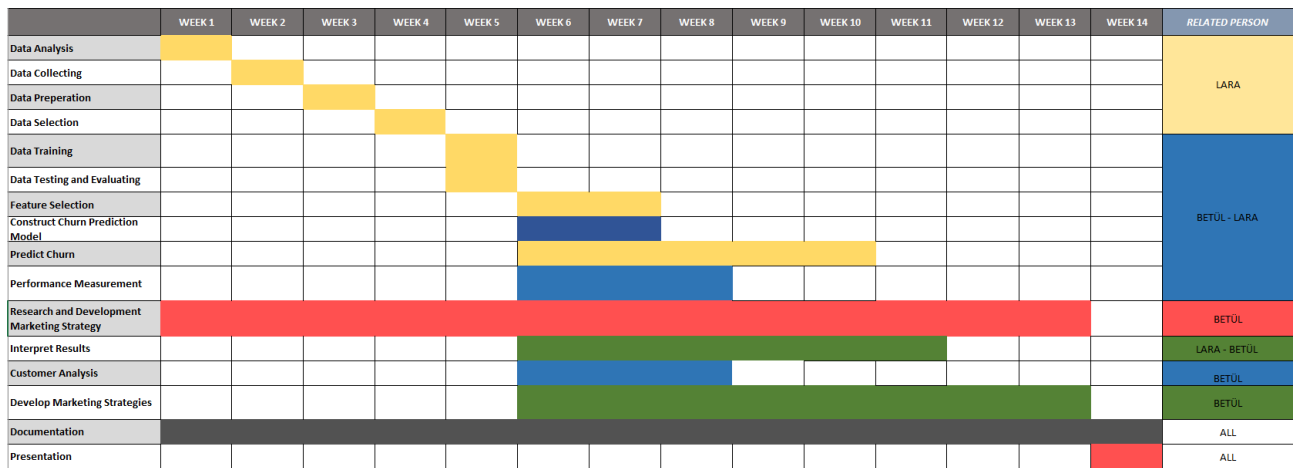
The project network diagram showing the order in which the activities of our project will be completed in chronological order is below.



**Figure 4 The project network.**

### 2.4. Gantt chart

The Gantt chart of the works to be followed can be found below.



**Figure 5 Gantt Chart**

## 2.5. Costs

In developing this application there was no cost since all the resources used during the developing process are free and open-source tools for both management and computer students.

Because the management students' responsibility is to implement ML algorithms and creating promotions and campaigns, all of this can be done at no cost, as well as computer students because open-source programming languages and libraries are used.

## 2.6. Risk Assessment

The risks that may be encountered during the project process are as follows.

**Table 3 Risk Matrix**

		Severity of the event on the project success				
		Minor	Moderate	Major		
Probability of the event occurring	RISK LEVEL	Unlikely	Possible	Likely	VERY LOW	This event is very low risk and so does not require any plan for mitigation. In the unlikely event that it does occur there will be only a minor effect on the project.
		VERY LOW	LOW	MEDIUM	LOW	This event is low-risk; a preliminary study on a plan of action to recover from the event can be performed and noted.
		LOW	MEDIUM	HIGH	MEDIUM	This event presents a significant risk; a plan of action to recover from it should be made and resources sourced in advance.
		MEDIUM	HIGH	VERY HIGH	HIGH	This event presents a very significant risk. Consider changing the product design/project plan to reduce the risk, else a plan of action for recovery should be made and resources sourced in advance.
					VERY HIGH	This is an unacceptable risk. The product design/project plan must be changed to reduce the risk to an acceptable level.

**Table 4 Risk Assessment**

Failure Event	Probability	Severity	Risk Level	Plan of Action
Insufficient Training Data	Likely There is a chance that the ML model won't have enough training data.	Moderate The impact of insufficient training data on model performance is considered moderate.	MEDIUM	Examine the viability of data gathering, take into account alternative strategies, and perform a sensitivity analysis on the influence of data size.
Inadequate ML Classifiers	Possible There is a considerable likelihood of using insufficient ML classifiers, which may not be able to accurately forecast churn data.	Major The accuracy of churn prediction can be greatly impacted by the use of insufficient ML classifiers, which is why it is deemed to have a high severity.	HIGH	Evaluate complexity and performance, use ensemble approaches, and track classifier performance as we evaluate and compare ML classifiers.
Model Overfitting	Unlikely The probability of the model overfitting throughout the training procedure is moderate.	Major Model overfitting can result in poor generalization and incorrect churn forecasts, which is why its severity is regarded as high.	LOW	Use regularization strategies, early stopping criteria, hyperparameter tuning, take into account ensemble methods, and routinely track and assess model performance.

Insufficient Training Data: There is a moderate chance of affecting accuracy. Our plan of action is to investigate data augmentation, assess the viability of extra data, and take into account different strategies.

Inadequate ML Classifiers: The likelihood of inaccurate classifiers is high. To reduce risk, we can evaluate and compare classifiers, as well as use ensemble approaches.

Model Overfitting: There is a moderate probability of overfitting which would lower the accuracy. Applying regularization strategies or early stopping functions, and fine-tune hyperparameters can be the solution.

### 3. SUB-SYSTEMS

#### 3.1. Management Engineering

In this section, we will discuss the trained ML algorithm, data analysis and appropriate promotion and campaign strategies for management engineers. Marketing strategies are developed after ML algorithms are trained on balanced data to predict customer churn.

The Management Engineering subsystem, an integral part of our loss estimation project, reflects and extends the work of the Computer Engineering team. Tasked with developing its own machine learning model, this subsystem is focused on creating marketing strategies backed by data insights. Different from but complementary to that of the Computer Engineering subsystem, the ML model is specifically tailored to identify effective customer retention tactics and marketing approaches.

In carrying out its role, the Management Engineering subsystem not only uses information from its own model, but also incorporates findings from the Computer Engineering team's models.

This dual approach provides a comprehensive understanding of customer behavior and churn risk, forming the basis of targeted marketing strategies.

Their responsibilities extend to the practical implementation of these strategies, overseeing their deployment and fine-tuning them based on real-time feedback. Through this process, the Management Engineering subsystem ensures the translation of predictive analytics into actionable, effective business strategies, thus playing a key role in the success of the project in reducing customer churn.

### **3.1.1. Review of technologies and/or methodology**

The Management Engineering subsystem is essential to expertly develop and apply machine learning models suitable for marketing strategy development. This subsystem must ensure that these models are solidly aligned with customer retention goals and effectively leverage insights from data analytics both internally and from the Computer Engineering team.

**Model Development and Integration:** The subsystem has strived to create a model that accurately predicts customer behavior and likelihood of churn by focusing on strategic aspects of marketing and customer engagement. This model creates a consistent analytical framework in cooperation with the Computer Engineering subsystem and seamlessly integrates with the information obtained from data analysis.

**Strategy Formulation and Implementation:** The subsystem is responsible for translating model insights into actionable marketing strategies. This involves crafting campaigns and initiatives that specifically target specific customer segments that are at risk of churn.

**Performance and Adaptability:** Strategies developed should be dynamically adjustable based on ongoing data analysis and market feedback. The subsystem must continually monitor the effectiveness of these strategies and strive to adjust them as necessary to maximize customer retention and satisfaction.

**Resource and Time Management:** Management engineering worked within the defined resource and time constraints of the project, ensuring marketing strategies were developed and executed efficiently.

**Collaboration and Communication:** Effective collaboration with the Computer Engineering subsystem is crucial. The management engineering subsystem must maintain open communication channels to ensure that data insights and model outputs are accurately understood and used effectively in strategy development.

**Compatibility and Scalability:** All strategies and models developed must comply with relevant legislation and ethical standards. Additionally, scalability is also crucial, as strategies must be adapted to changing customer sizes and market changes.

In summary, the management engineering subsystem is tasked with bridging the gap between



technical data analytics and practical marketing strategy implementation, ensuring that the churn forecasting project not only predicts customer behavior but is also actively involved in customer retention.

As Management Engineers, our approach to customer retention and churn forecasting is structured and strategic. We use the LightGBM model for our predictive analysis, providing an effective and accurate understanding of customer behavior and churn risk.

#### **An overview of our methodology:**

**Data Analysis with LightGBM:** We use the LightGBM model to process and analyze customer data. The efficiency of this model in processing large data sets allows us to quickly identify patterns and trends associated with customer churn.

**Identifying Customer Segments at Risk:** Using the information obtained from LightGBM and other models' analysis, we identify customer segments that are at higher risk of churn. This step is crucial to target our efforts effectively.

**Creating Targeted Marketing Strategies:** We develop special marketing strategies according to determined customer segments. These strategies are designed to address the specific needs and preferences of each segment, thereby increasing customer engagement and loyalty.

**Implementation of Marketing Strategies:** We then publish these marketing campaigns on various channels, ensuring wide reach and effective interaction with the target audience. Marketing strategies include personalized communication methods, loyalty programs, incentives, product and service diversification, customer support and service quality improvements, digital marketing, social media and data-driven improvements.

#### **Physical Arc:**

The physical architecture of the management engineering approach to customer retention and churn prediction integrates various technological and procedural elements into a coherent system. This system is designed to synergize data analytics with strategic marketing to effectively manage customer relationships and reduce churn.

**Analytical Processing Units:** Dedicated servers or local processing solutions to run the ML models and other analytical tools. Personal computing resources to handle complex data processing and model training.

**Segmentation and Strategy Development Tools:** Advanced software for customer segmentation analysis to identify at-risk customer groups. Tools for developing and planning targeted marketing strategies by customer segments.

**Marketing Execution Platforms:** Multi-channel marketing platforms for the distribution of various campaigns (email, social media, digital ads, etc.). CRM systems integrated with marketing

tools to manage and track customer interactions.

### **3.1.2. Proposed solution approach**

The Management engineering approach to customer retention and churn prediction can be conceptualized as a multi-layered strategy that combines advanced data analysis with targeted marketing initiatives. This conceptual framework is designed to optimize customer engagement and reduce churn by leveraging technology and customer insights.

#### ***Layer 1: Creating Data-Driven Insight***

##### **Core Technology: ML Models**

The basis of our approach is based on the implementation of the ML models. This advanced machine learning tool allows us to efficiently process large amounts of customer data, allowing us to derive meaningful insights about customer behavior and churn risks.

#### ***Layer 2: Strategic Customer Segmentation***

##### **Identifying Segments at Risk**

Leveraging insights from ML models, we identify specific customer segments that are more prone to churn. This identification process is based on a variety of factors, including usage patterns, service interactions, and demographics.

##### **Adapting Segment-Specific Strategies**

Each at-risk segment is analyzed to understand its unique characteristics and needs. This understanding informs the development of specific strategies aimed at addressing the specific causes of potential loss in each segment.

#### ***Layer 3: Execution of Dynamic Marketing Strategy***

##### **Development of Marketing Strategies**

We create a set of marketing strategies based on our in-depth analysis. These strategies range from personalized communication and loyalty programs to service diversification and quality improvements.

##### **Multi-Channel Campaign Distribution**

Implementation of these strategies occurs across a variety of channels, ensuring broad reach and deep customer engagement. Our approach leverages the power of both digital and traditional marketing mediums to effectively engage customers.

Our conceptual framework is essentially the synergy of data analytics and customer-centric marketing. By combining ML models' technical prowess with strategic marketing initiatives, we create a robust mechanism to reduce customer churn and increase loyalty. This framework is not static, but an evolving strategy that adapts and grows with our customers and the market.

### 3.1.3. Data collection and analysis

#### 3.1.3.1. Review the availability of the data

In our project, the customer churn data set on the Kaggle platform under the title "Customer Churn: Working with Imbalanced Dataset" is used. This dataset contains detailed information on customer churn in the telecommunications industry and serves as a critical resource for customer churn analysis.

##### **Properties of the Data Set:**

- The data set includes various attributes such as customers' subscription periods, service usage details, payment methods, and customer demographics.
- In particular, labels indicating customer churn status ('Churn' column) are a key element for our analysis.

##### **Analysis of the Data Set:**

- This dataset obtained from Kaggle is easily accessible and processable by management engineering and computer engineering teams.
- The volume and diversity of the dataset is sufficient for effective training and testing of machine learning models.
- However, some features of the dataset (for example, unnecessary columns such as 'CustomerID') were found to be irrelevant for our project and such data were removed during the analysis process.

##### **Data Quality and Processing:**

- The data set contains challenges such as missing data and unbalanced distributions. SMOTE and Random Under Sampling techniques have been applied to solve these problems.
- The Kaggle dataset was examined in detail during the EDA (Exploratory Data Analysis) process and the effects of important features (for example, payment methods and service types) on customer churn were analyzed.

This data set plays an important role in customer churn analysis and development of effective customer retention strategies. Analyses were made taking into account the limitations and characteristics of the data set and the results were interpreted in this context. This has allowed us to obtain more comprehensive and effective results in customer churn prediction and development of marketing strategies.

#### 3.1.3.2. Data collection

Telecommunication loss data from Kaggle was aggregated. At the same time, appropriate data about customers was also obtained.

#### 3.1.3.3. Data analysis

The data analysis and data preprocessing phase were carried out under the leadership of Computer

Engineering and was integrated with the Management Engineering subsystem. All visualizations and details are provided in the Computer Engineering subsystem.

In summary;

*Data Preprocessing and Analysis:*

- Conducted exploratory data analysis (EDA) on customer abandonment data using the LightGBM model.
- Hyperparameter adjustments of the model were made using HyperOpt and Optuna tools.
- The 'CustomerID' column that was located in the Irrelevant location has been removed and some attributes have been converted accordingly (for example, 'TotalFees' has been converted to a quantitative format).

*Unbalanced Data Problem and Solutions:*

- An attempt was made to eliminate the imbalance in the 'Churn' feature by using SMOTE and Random Under Sampling methods.

*Critical Findings and Insights:*

- The impact of payment methods, contract types and internet services on customer churn was determined.
- The effect of monthly and total fees on customer departure was analyzed.
- It has been observed that how long customers have been served (tenure) has a significant impact on their likelihood of leaving.

*Data Visualization and Interpretation:*

- Data distributions and important features were visualized using Seaborn and Matplotlib.
- These visualizations played a critical role in the data preprocessing and model development process.

This process resulted in successful visualizations and improvement of the dataset, but it was also accompanied by some challenges. In particular, complex relationships between data and missing values posed significant challenges to the analysis process. These challenges required further analytical approaches and contributed to the development of the project.

### **3.1.4. Application and Results**

During the implementation phase of our loss estimation project, we transform our strategies and models from conceptual frameworks into practical, operational tools. This phase is characterized by the application of our developed machine learning models and the application of these models to our marketing strategies.

#### **Applying Advanced Machine Learning Models:**

Initially, we used the LightGBM classifier, which is known for its efficiency in processing large datasets, especially those with categorical features. Our preprocessing methods

included encoding categorical data with LabelEncoder and scaling of all features using StandardScaler. Realizing the need for balanced data, we implemented SMOTE for dataset balancing. The data was then divided into training and testing sets. We improved the accuracy of our model via RandomizedSearchCV by fine-tuning key parameters to optimize performance. This led to a significant improvement in accuracy, precision, recall, and F1 scores.

### **Transition to Stacked Classifier Approach:**

To further develop our model, we adopted the stacking classifier approach. This method involved more precise handling of missing values, especially in the 'TotalCharges' property. We included OneHotEncoder for more fine-grained processing of categorical features while still using SMOTE for class balance. The stacking classifier combined RandomForest, GradientBoosting, and SVC as base learners and Logistic Regression as meta learner. This combination of various models was intended to increase our predictive capabilities. After training, the Stacking Classifier showed improved performance on all key metrics; In particular, it achieved a significant improvement over the first LightGBM model, achieving an accuracy of around 0.86 on the test set.

### **Application of Models to Marketing Strategies:**

With the implementation of these advanced models, we began to integrate their insights into our marketing strategies. This involved tailoring our campaigns to customer segments identified as being at high risk of churn. Powered by data-driven insights, our marketing strategies include personalized communications, loyalty programs, and a variety of digital marketing initiatives across multiple channels. The effectiveness of these strategies is continually evaluated through ongoing data analysis and adjustments based on market feedback.

Throughout the realization phase, our management engineering team ensures that the predictive power of our advanced machine learning models is fully leveraged to inform and guide our marketing efforts. This phase is crucial to bridging the gap between technical data analytics and practical marketing strategy implementation, actively engaging in customer retention, and significantly improving our customer churn prediction capabilities.

Marketing strategies to be implemented after customer losses are determined will create a more effective solution after understanding why customers are lost. Accordingly, it is necessary to ensure communication and interaction with customers, taking into account regulations (such as KVKK). Trying to regain lost customers by implementing loyalty programs should be one of the priority strategies. Giving discounts, special offers, gift products or points via SMS or e-mail to customers who were previously customers but did not continue consuming later, if their permission was obtained while they were customers in time, will ensure communication again with these potential customers. The reason why the primary goal is to use this program is to remind the customer of himself again and if he has any unhappiness, find out and solve it. The fact that other strategies are

relatively dependent on this is a matter of communication that needs to be re-entered with the customer. In addition, while implementing this program, if customers left after complaining and expressed this on any platform, a special program will be created for complaints.

When implementing this loyalty program, it is essential to apply personal communication methods. Thinking that all customer losses occur for the same reason would produce erroneous results. While some customers complain about the increase in prices, some customers may complain about the decrease in prices and the decrease in service quality, so there may be opposing views. Accordingly, another important issue as well as learning about customer losses is why and for what reason these customers were lost.

Product and service differentiation will create a new and interesting situation for customers. Lost customers may choose the brand again within this service and product diversity. In addition, if some customers are bored or do not like the products and services, this may lead them to become repeat customers.

Apart from differentiation, it is also important to make improvements in products and services and include this in the marketing strategy. In re-establishing relationships with customers, making them aware of this improvement and creating new expectations will enable them to return lost customers more quickly.

Finally, and in order to meet these strategies, customer feedback should be given importance. Lost customers indicate elasticity in demand for products and it is very possible to lose these customers again.

According to the data analysis shown below, marketing strategies are developed by the management engineering subsystem;

“there is a strong correlation between "tenure" (or "MonthlyCharges") and "TotalCharges" because  $\text{"TotalCharges"} = \text{"tenure"} * \text{"MonthlyCharges"}$ . In addition, we can see that there is a strong correlation between "OnlineSecurity" and "MonthlyCharges". This is probably because the "fiber optic" is the most expensive service and this is the reason for high monthly charges. It's also can explain why the churn probability is higher for the customers who own this service - the price is too high for them. Another strong negative correlation is between "tenure" and "Contract".It make sense that the higher the tenure - the contract will be yearly and not monthly. Finally if we look at the stronget correlation with "Churn", it's "tenure".

We can clearly see the trend that the higher the tenure - the lesser chance the customer will leave **Gender** does not seem to impact churn, as the churn percentage of both male and female are almost same

Customers who are **SeniorCitizen** are highly prone to churn (**41.68%**)

Customer having **MultipleLines** are slightly more prone to churn (**28.6%**), compared to customers which **don't have MultipleLines** (**25.04%**) and **No Phone Service** (**24.92%**)

Customers which have **Fiber optic InternetService** are highly prone to churn (**41.89%**)

Customers which have **DSL InternetService** are slightly prone to churn (**18.95%**)

Customers which have **No InternetService** are least prone to churn (**7.40%**)

- Customers which have **No OnlineSecurity** are highly prone to churn (**41.76%**)
- Customers which have **OnlineSecurity** are slightly prone to churn (**14.61%**)
- Customers which have **No InternetService** are least prone to churn (**7.40%**)

1. Customers which have **No OnlineBackup** are highly prone to churn (**39.92%**)
2. Customers which have **OnlineBackup** are slightly prone to churn (**21.53%**)
3. Customers which have **No InternetService** are least prone to churn (**7.40%**)

- Customers which have **No DeviceProtection** are highly prone to churn (**39.12%**)
- Customers which have **DeviceProtection** are slightly prone to churn (**22.50%**)
- Customers which have **No InternetService** are least prone to churn (**7.40%**)

- Customers which have **No TechSupport** are highly prone to churn (**41.63%**)
- Customers which have **TechSupport** are slightly prone to churn (**15.16%**)
- Customers which have **No InternetService** are least prone to churn (**7.40%**)

- Customers which have **No StreamingTV** are highly prone to churn (**33.52%**)
- Customers which have **StreamingTV** are slightly prone to churn (**30.07%**)
- Customers which have **No InternetService** are least prone to churn (**7.40%**)

- Customers which have **No StreamingMovices** are highly prone to churn (**33.68%**)
- Customers which have **StreamingMovices** are slightly prone to churn (**29.94%**)
- Customers which have **No InternetService** are least prone to churn (**7.40%**)

- Customers which have **Month-to-month** contract are highly prone to churn (**42.70%**)
- Customers which have **One year** contract are slightly prone to churn (**11.26%**)
- Customers which have **Two year** contract are least prone to churn (**2.83%**)

- Customers which have **Electronic check** payment method are highly prone to churn (**45.28%**)
- Customers which have **Mailed check** payment method are moderately prone to churn (**19.10%**)
- Customers which have **Bank transfer (automatic)** payment method are moderately prone to churn (**16.70%**)
- Customers which have **Credit card (automatic)** payment method are moderately prone to churn (**15.24%**)

By calculating Churn percentages (of feature values), we can see that **PaymentMethod (Electronic check)** is the leading factor of Churn, followed by **Contract (Month-to-month)**, **InternetService (Fiber optic)** and so on...

	Feature	Values	Churn_NO	Churn_YES	Total	Churn Percentage
0	PaymentMethod	Electronic check	1294	1071	2365	45.285412
1	Contract	Month-to-month	2220	1655	3875	42.709677
2	InternetService	Fiber optic	1799	1297	3096	41.892765
3	OnlineSecurity	No	2037	1461	3498	41.766724
4	SeniorCitizen	Yes	666	476	1142	41.681261

**From MonthlyCharges, we can observe the following;**

- 50% of all churned customers have MonthlyCharges from 80-120
- 75% of all un-churned customers have MonthlyCharges from 20-90

**So we can say that high MonthlyCharges contribute to Churn**

**most of the churned customer lie towards lesser TotalCharges, meaning lower TotalCharges contribute towards customer churn**

- 75% of churned customers have TotalCharges between 0-2500 approx.

**From tenure boxplot, we can observe the following;**

- 75% of all churned customers have tenure less than 30

**Low tenure customers are more likely to Churn**

**again,** Churn rate is high when tenure is lower, they are inversely proportional

the relationship of MonthlyCharges and TotalCharges is very much dependent on tenure. If tenure is lower (for e.g. tenure\_group = 1-12) then TotalCharges remain low, even if the MonthlyCharges increase, but as the tenure increases the TotalCharges increases with the increase of MonthlyCharges.

1. High MonthlyCharges impact TotalCharges only with the increase in tenure
2. Churn rate is highest when tenure is lowest
3. High density of churn customers belong to a specific criteria **(High MonthlyCharges, Low TotalCharges, and Low tenure)"**

1. **Tiered Pricing for Fiber Optic Services:** There is a high rate of churn (41.89%) among fiber optic users. To solve this situation, we will offer multi-level fiber optic service options with different speeds and features, at various price points. Thus, we will reduce cost-related customer loss by targeting both budget-friendly and premium customers.
2. **Loyalty Rewards for Long-Term Customers:** Low churn rates are seen among long-term customers. By creating a customer loyalty program, we will offer different advantages to customers according to their duration. This could be in the form of discounts, exclusive content, or free upgrades after a certain period of time.
3. **Special Promotions for Elderly Customers:** There is a high rate of churn (41.68%) among elderly customers. We will develop special promotions for them, such as simplified plans,



discounted services or improved customer support. This may include workshops or personalized assistance to make it easier for them to use our services.

- 4. Packages Including Online Security/Services:** Customers who do not have online security or backup services experience a high rate of customer churn (41.76% and 39.92% respectively). We will create packages that include these services at a discount. This will be effective in both addressing security concerns and increasing the perceived value of our services.
- 5. Flexible Contract Options:** There is a high rate of customer churn (42.70%) in month-to-month contracts. By targeting these customers, we will offer more flexible contract options that will appeal to customers who avoid long-term commitments. This can be 6-month contracts or annual contracts that allow for service adjustments halfway through.
- 6. Incentives for Switching to Electronic Payment Methods:** The use of electronic checks is associated with a high rate of customer churn (45.28%). We will encourage customers to switch to automatic bank transfers or credit card payments. This could be in the form of a one-time discount or a small monthly refund for switching to these payment methods.
- 7. Upgrade Campaigns for Customers Without DSL or Internet Service:** Customers without DSL or Internet service lose customers at a lower rate. We will encourage these customers to upgrade to higher tier internet services at an initial discount or trial period. This will increase our revenues while increasing customer interactions and satisfaction.
- 8. Personalized Communication to Customers at High Risk of Churn:** Our analysis shows that customers who pay high monthly fees but have low tenure are at high risk of churn. We will develop personalized communication strategies for these customers, such as offering them tailored plans or providing proactive customer support.

The Personalized Communication Strategy will be aimed at groups at high risk of customer churn, that is, customers who pay low-term but high monthly fees. This strategy may include the following elements:

**One-to-One Communication and Consultancy:** Customer relations representatives can communicate one-on-one with high-risk customers and understand their needs and concerns. This may include private counseling sessions, tips or advice on using the service.

**Customized Plan Offerings:** Customized plans can be offered based on customers' usage habits and preferences. For example, offering discounts for less used services or additional benefits for more used services.

**Proactive Support and Follow-Up:** By regularly monitoring customers' satisfaction, customer service teams can provide proactive support before they experience any problems.

**Special Communication Channels:** High-risk customers can be offered special communication

channels (for example, a priority customer line) and rapid response times

**Personalized Marketing Content:** Email campaigns or SMS notifications customized based on the customer's preferences and past interactions.

- 9. Referral Bonuses for Existing Customers:** We will develop a program for existing customers to refer new customers. This strategy will help us grow our customer base as well as encourage our existing customers to engage more with our brand.

The Customer Referral Program will be designed to help existing customers bring in new customers and will operate as follows:

**Creating a Reward System:** When an existing customer brings in a new customer, rewards can be offered to both. Rewards can include discounts, service upgrades, or free use of certain services.

**Easy Joining Process:** It should be easy for customers to understand and join the program. Simple referral codes or links can be used via online platforms or mobile applications.

**Referral Limits and Conditions:** Certain limits and conditions can be set to prevent abuse of the program. For example, in order to receive the maximum number of referrals or rewards per month, the new customer may need to use the service for a certain period of time.

**Tracking Referral Success:** To measure the effectiveness of the referral program, it is important to track which customers bring in the most new customers and continually improve the program.

**Referral Campaigns:** During special periods, such as holiday seasons, increasing referral bonuses or offering additional rewards can increase customer engagement.

- 10. Feedback-Driven Service Improvements:** We will regularly collect and analyze customer feedback, especially from customers who abandon our services. Using this data, we will continuously improve our services and address specific issues that cause customer churn.

## ***LOYALTY PROGRAM DESIGN***

### **1. Tiered Loyalty Levels:**

**Duration-Based Levels:** Customers will be assigned to different loyalty levels based on their contract renewal period. For example, Bronze levels will be created for 1 year, Silver for 2 years, and Gold levels for 3 years.

**Increased Rewards:** Each level will offer more benefits and rewards than the previous one.

### **Points System:**

**Earning Points:** Customers will be awarded points for each renewed month. These points are available when certain thresholds are reached.

**Reward Options:** Points will be redeemed for discounts or special offers on various services.

## **2. Special Renewal Advantages:**

### **Discounts and Upgrades:**

**Renewal Discounts:** Renewing customers will be offered special discounts and plan upgrades.

**Tier-Based Benefits:** Higher-tier loyalty members will enjoy greater discounts and premium features.

### **Free Additional Services:**

**Loyalty Extras:** Loyal customers will be provided with free additional services such as additional data, premium channels or advanced technical support.

## **3. Personalized Rewards:**

### **Special Offers:**

**Customer Analysis:** Customer usage habits and preferences will be analyzed and personalized offers will be presented based on these.

**Rewards According to Individual Needs:** Rewards will be designed to suit each customer's needs and preferences.

### **Birthday and Anniversary Bonuses:**

**Special Day Celebrations:** Special bonuses and discounts will be given for customers' birthdays and contract anniversaries.

## **4. Early Access and Previews:**

### **Special Content:**

**Early Access Rights:** Loyal customers will be provided with early access to new services and content.

**Preview Events:** Previews of new features and exclusive content will be available.

## **5. Referral Programs:**

### **Referral Bonuses:**

**Incentive Bonuses:** Customers will earn bonuses or additional loyalty points when they refer friends or family to renew their contract.

## **6. Communication and Recognition:**

### **Personalized Thank You Messages:**

**Thank You Communication:** Personalized thank you messages will be sent for each renewal and attention will be drawn to the customer's loyalty.

### **Recognition Programs:**

**Customer Highlight:** Loyal customers will be highlighted in the company's communication channels and their loyalty will be appreciated.

## **7. Flexible Usage Options:**

### **Various Usage Catalogue:**

**Wide Range of Rewards:** A variety of reward options will be offered, such as discounts, merchandise or charitable donations.

## **8. Regular Interaction:**

### **Special Webinars or Events:**

**Interactive Events:** Special webinars and events will be held for loyal customers.

Feedback Sessions:

**Customer Participation:** Customer feedback will be received and used to continuously improve services.

## **9. Gamification Elements:**

### **Challenges and Badges:**

**Loyalty Games:** Customers will earn badges and rewards when they reach certain loyalty goals.

## **10. Continuous Program Evaluation:**

### **Data analysis:**

**Evaluation of Program Effectiveness:** The program will be constantly updated by analyzing customer feedback and interaction data.

### **3.1.5. Discussion of results**

During the evaluation phase of our loss estimation project within the framework of management engineering, we evaluate the strategy and model we have implemented theoretically and practically. This phase is important to ensure that our approaches are aligned with our goals of increasing customer retention and reducing churn.

#### **3.1.5.1 Theoretical**

The theoretical foundations of this project provide an in-depth understanding of preventing customer churn and increasing customer loyalty. The LightGBM model used allowed us to make sharp predictions about customer behavior and churn probabilities. The hypothesis in the project is that advanced data analysis and machine learning models will be effective in reducing customer churn and improving marketing strategies. The theoretical framework highlights the applications of data

science in business and its importance in developing marketing strategies. This approach demonstrates how insights gained from analyzing customer data provide a strategic advantage in creating targeted marketing campaigns.

### **3.1.5.2 Practical**

Practical applications of the project have been effective in developing real-time and dynamic marketing strategies by processing customer data. The results obtained were particularly successful in identifying at-risk customer segments and customized marketing initiatives for these segments. However, some limitations were also observed. For example, the model's difficulties in accurately predicting behavior in some customer segments have limited the effectiveness of marketing strategies. Additionally, the adaptation and scalability of the model in different market conditions has emerged as a significant challenge.

### **3.1.6. Discussion of limitations**

One of the most obvious limitations of this project is the scope and diversity of the dataset used. The limited demographic and geographic scope of the data set affects the generalizability of the model. Additionally, the model's adaptability to real-world conditions and its ability to respond to dynamic market changes may be limited. This poses a challenge, especially in the face of rapidly changing market trends and customer preferences. Finally, the complexity and constant evolution of customer behavior requires constant model updates and retraining, which poses a challenge in terms of resource and time management.

Although the Kaggle dataset used in this project contains valuable information about customer churn, it has some important limitations:

**Representativeness of the Data Set:** The specificity of the data set to a particular field may limit the generalizability of the results and the model. Different geographic regions, customer demographics, and market conditions may affect the model's applicability to other companies or industries.

**Unbalanced Data Distribution:** Customer abandonment data often has an unbalanced distribution. This can lead to the model overrepresenting the majority class (e.g., customers who do not leave) and underrepresenting the minority class (e.g. customers who leave). This may limit the model's ability to accurately predict separation events in real-world scenarios.

**Complexity of Features and Relationships:** The complexity of the features in the data set and the relationships between these features can have an impact on the accuracy and interpretability of the model. In particular, the constant change of customer behavior and preferences requires updating and continuous optimization of the model.

**Model Scalability and Adaptive Capacity:** The model's ability to adapt to different customer bases and market changes may be limited by the scope and diversity of the data set. This can limit the

effectiveness and applicability of the model, especially in growing and changing business environments.

**Ethics and Privacy Issues:** The use of customer data requires caution, especially regarding the protection of personal data and confidentiality. This makes the compliance of data collection and processing processes with ethical standards and compliance with legal regulations an important issue.

These limitations create significant impacts on the overall effectiveness and feasibility of the project and offer potential areas for future improvements. In this context, it is important to constantly update the model, benefit from different data sources and develop methods to handle unbalanced data distributions.

### **3.2. Computer Engineering**

The Computer Engineering subsystem serves as the technical backbone of our churn prediction project. Tasked with the critical functions of data preparation, algorithm implementation, and model training, this sub-team lays the groundwork upon which predictive analytics operate.

This subsystem is responsible for architecting the data pipeline, from initial ingestion to final processing. It ensures that data is clean, relevant, and structured in a way that is conducive to accurate analysis. The team also selects and fine-tunes machine learning algorithms, balancing the nuances of performance against the practicalities of computational efficiency. By optimizing these models, the Computer Engineering sub-team directly contributes to the predictive prowess of the end product, enabling accurate churn forecasts that drive strategic business decisions.

#### **3.2.1. Requirements**

The Computer Engineering subsystem must efficiently execute data preprocessing, ensuring data is accurately cleaned and structured for analysis. It is responsible for the selection, implementation, and fine-tuning of machine learning algorithms, with a particular focus on addressing the class imbalance problem. This subsystem is tasked with optimizing model performance to achieve reliable predictions while ensuring computational efficiency. The algorithms selected should seamlessly integrate with the larger system architecture, contributing to the overarching goal of developing a cost-effective, accurate churn prediction platform. Performance is measured not just by predictive accuracy but also by the models' scalability and the subsystem's adherence to the project's time and resource constraints.

### 3.2.2. Technologies and Methods

The Computer Engineering subsystem leverages a suite of technologies and methods tailored for predictive analytics. It utilizes Python for its rich ecosystem of data processing and machine learning libraries such as Pandas, NumPy, and scikit-learn, alongside advanced algorithms from CatBoost and LightGBM for model training. The subsystem employs Jupyter Notebooks for iterative development and visualization, with data stored and managed using CSV files, eliminating the need for complex database systems. Hardware requirements are minimal, capitalizing on the efficiency of the chosen libraries and the computational power of standard workstations.

### 3.2.3. Conceptualization

In consideration of the physical architecture and the tasks at hand, we have delineated two primary conceptual solutions for the Computer Engineering subsystem:

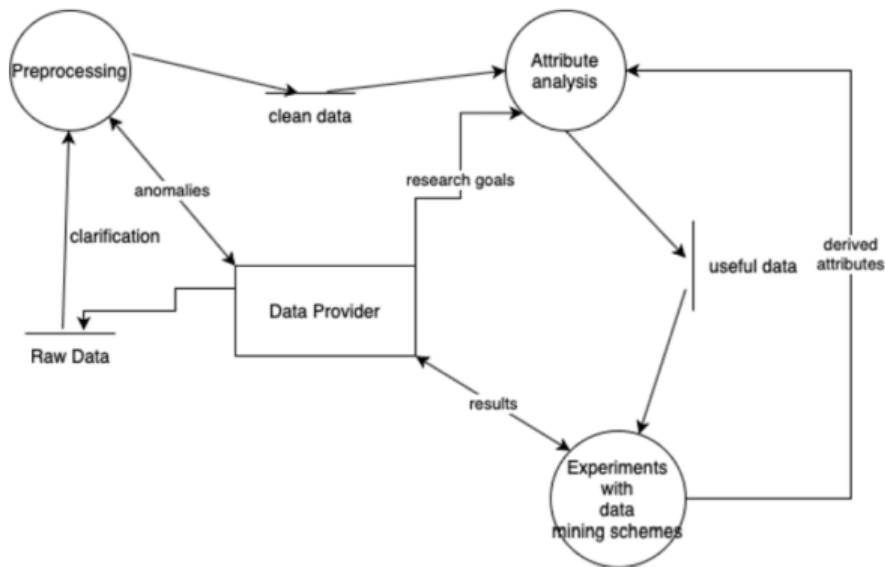
- **Automated Data Pipeline:** This solution is engineered to streamline data preprocessing and effectively manage the issue of class imbalance. Its automation capabilities are designed to handle large datasets efficiently, reducing manual intervention and expediting the preprocessing phase which is crucial for timely model training and subsequent stages.
- **Multi-model Learning:** This approach enhances the predictive accuracy of churn models by leveraging the strengths of multiple machine learning algorithms. By integrating various models, we can construct a robust predictive framework that outperforms individual algorithms, especially in complex scenarios involving intricate patterns within the data.

Both conceptual solutions aim to confront the project's central challenges: bolstering prediction accuracy through Multi-model Learning and refining the efficiency of data preprocessing with an Automated Data Pipeline. These solutions provide a strategic direction for the subsystem's technical execution and lay the groundwork for a holistic design that underpins the entire churn prediction system. By implementing these solutions, we aim to elevate the precision and reliability of our predictions, thereby enabling targeted and effective churn mitigation strategies.

### 3.2.4. Physical architecture

The physical architecture of the project includes the infrastructural and fundamental elements and it shows the main components such as data collection, preprocessing, model training, testing and evaluation stages. Additionally, it allows for the development of campaign strategies according to the

evaluation results.



**Figure 6 Physical Architecture Model (Adapted from Cleary et al., 1996)**

### 3.2.5. Materialization

The materialization of the Computer Engineering subsystem for churn prediction began with the construction of an Automated Data Pipeline. This process commenced with importing the telecommunications dataset using Pandas, a choice dictated by its powerful data manipulation capabilities. The dataset, sourced in CSV format, required cleaning and preprocessing—a task handled adeptly by the pipeline. Customer IDs were removed, ensuring anonymity and relevance, while the 'Churn' column was encoded to numerical values to facilitate machine learning processes.

As we progressed, the subsystem encountered challenges with missing values in 'TotalCharges', a crucial variable for prediction. The solution was to impute these gaps, enhancing data integrity without compromising the dataset size. The pipeline then implemented resampling to counteract class imbalance, using techniques like SMOTE for oversampling, which proved crucial in developing a more balanced and fairer predictive model.

The Multi-model Learning strategy took form through the application of various machine learning algorithms. We utilized CatBoost for its categorical data handling prowess and implemented a grid search to fine-tune hyperparameters, significantly improving model performance. However, this approach was not without setbacks. The initial model complexity led to overfitting; adjustments were made by simplifying the model and re-evaluating feature importance, leading to a refined and more generalized predictive capability.

Throughout the building process, the team documented progress through code commits and periodic snapshots of the evolving system. Each model's performance was



recorded, and the best-performing iterations marked major milestones. These instances were captured in visualizations using libraries like Seaborn and Plotly, which provided insights into data distribution and model accuracy, guiding further refinements.

In some instances, the plan deviated from the initial proposal. Notably, we pivoted towards more advanced resampling methods and adopted ensemble techniques earlier than anticipated to address unexpected discrepancies in model performance across different segments of the data. These adjustments were made in response to the iterative feedback obtained from continuous testing and validation phases.

The subsystem's journey was marked by iterative improvement—tweaking, testing, and sometimes overhauling models to achieve our aim of accurate, actionable churn predictions. The evolution of the subsystem reflects the iterative essence of machine learning development, where adaptability and methodical refinement shape the path to achieving precise churn predictions.

## **Data Analysis and Preprocessing**

The project's objective is to set the stage for rigorous data exploration and advanced modeling techniques. We embarked on Exploratory Data Analysis (EDA) with the intent of deploying sophisticated models such as XGBoost and CatBoost, emphasizing the fine-tuning of their hyperparameters using tools like HyperOpt and Optuna. A crucial decision was to focus on optimizing for recall, recognizing the importance of accurately identifying actual churn customers.

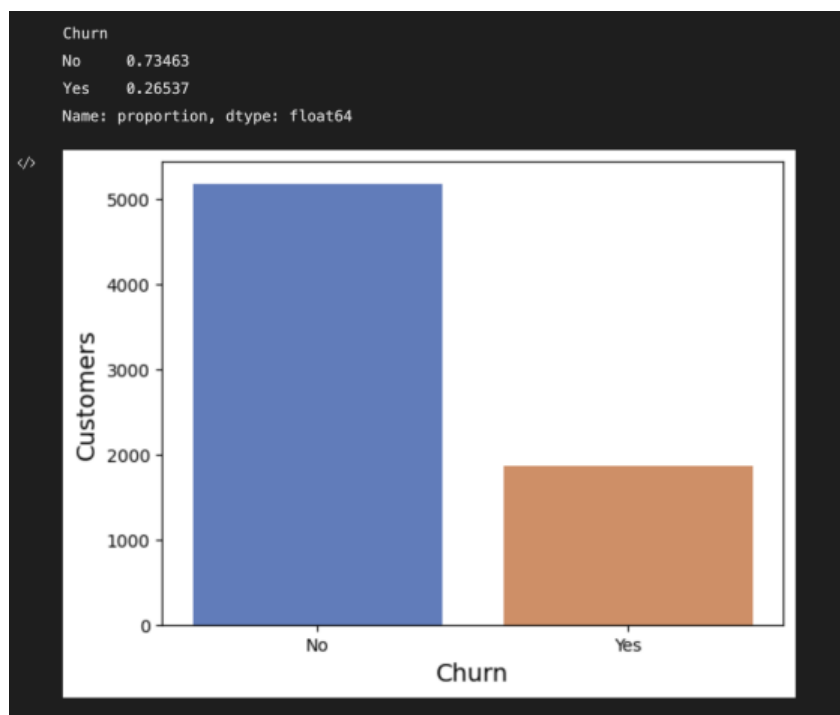
In the EDA phase, several key insights emerged. We identified the 'customerID' column as non-essential, leading to its removal for a streamlined model. A significant transformation was converting the 'TotalCharges' column from a non-numeric to a float format, enabling precise data interpretation. The 'Churn' feature's imbalance was addressed through SMOTE and Random Under Sampling techniques, markedly improving the recall score. Our analysis highlighted those factors such as payment methods, contract types, and internet service options significantly influenced churn rates. For instance, fiber optic service users and those with month-to-month contracts showed higher churn rates. We also investigated the relationship between various charges and churn. Our findings indicated that while higher monthly charges led to increased churn, the impact of total charges was more nuanced and warranted further analysis. Introducing a 'tenure\_group' category enabled a deeper look at churn concerning customer tenure, revealing an inverse relationship between tenure length

and churn likelihood. Furthermore, we established that gender did not significantly impact churn.

Challenges like missing values and anomalies, particularly in cases of zero tenure, were meticulously addressed to maintain data integrity. The use of visualization tools such as Seaborn and Matplotlib played a vital role in illustrating data distributions and informing strategic decisions. This phase adapted to encompass nonlinear relationships between charges and other variables, thereby laying a robust foundation for predictive modeling and enhancing our churn prediction strategies.

Through categorical feature analysis, we unearthed critical insights. Features like 'Internet Service Type', 'Contract', and 'Payment Method' were found to significantly influence churn probability. Additionally, the analysis underscored an inverse relationship between tenure and churn probability, highlighting the need for retention strategies in the early customer lifecycle stages. Our journey through this phase was marked by both successes and learning opportunities. Visualizations provided key insights for data preprocessing and model development, helping us identify crucial trends and patterns.

In summary, this comprehensive phase of the project not only yielded valuable insights but also prepared the ground for more sophisticated analytical approaches and predictive modeling.



**Figure 7 Proof of Imbalanced Data**

	Feature	Values	Churn_NO	Churn_YES	Total	Churn Percentage
0	PaymentMethod	Electronic check	1294	1071	2365	45.285412
1	Contract	Month-to-month	2220	1655	3875	42.709677
2	InternetService	Fiber optic	1799	1297	3096	41.892765
3	OnlineSecurity	No	2037	1461	3498	41.766724
4	SeniorCitizen	Yes	666	476	1142	41.681261

**Figure 8 Five Factors That Effect Churn the Most**

### Model 1

As mentioned before, the data preprocessing includes dropping unnecessary columns, handling missing values, converting data types, and encoding categorical variables. For instance, 'TotalCharges' is converted from an object to a numeric type, and 'Churn' is encoded as a binary variable. The EDA phase employs various visualization techniques like count plots, box plots, and scatter plots, offering insights into the data distribution and relationships between features.

The primary models used for the churn prediction are XGBoost and CatBoost classifiers. We selected these models for their efficiency in handling diverse datasets, with CatBoost being particularly adept at processing categorical data without extensive pre-processing. Both models, based on gradient boosting, are excellent for sequential model building, crucial for reducing errors and enhancing predictive accuracy. XGBoost is noted for its scalability and efficiency, suitable even for our dataset of around 7000 churn customers. It also provides robust tools for feature importance analysis, enabling us to identify key factors influencing customer churn. The flexibility in hyperparameter tuning with both models allows for customization to our dataset's specific characteristics. Their proven performance in classification tasks makes them reliable choices for accurately predicting churn probabilities. These are gradient boosting frameworks known for their effectiveness in handling diverse datasets. The dataset is split into training, validation, and test sets, and features are scaled using StandardScaler.

The XGBoost classifier is trained with its default parameters initially. The model's performance is assessed using metrics like accuracy, recall, precision, F1 score, Matthews correlation coefficient, and ROC-AUC, providing a multi-faceted view of its predictive capabilities. The initial performance of our XGBoost model provided valuable insights but

also indicated a tendency towards overfitting, as evidenced by a notable discrepancy between the training and validation metrics. The model's overfitting was reflected in a significant drop in all performance metrics when transitioning from the training to the testing dataset, signaling that the model was not generalizing well to unseen data.

To address this, we employed HyperOpt for hyperparameter tuning, aiming to find a sweet spot where the model complexity was just right to capture patterns in the data without fitting to noise. The tuning process yielded a more balanced model with improved generalization, as shown by the convergence of performance metrics between the training and test sets. Although accuracy hovered around a satisfactory 80%, the recall metric was still lower than desired at approximately 48%.

Preprocessing Method	Algorithm	Train Metrics	Validation Metrics	Test Metrics
StandardScaler SMOTE Under Sampling HyperOpt	CatBoost	Accuracy: 80%, Recall: 85%, Precision: 78%, F1: 81%, MCC: 60%, ROC-AUC: 80%	Accuracy:76%, Recall: 80%, Precision:53%, F1: 64%, MCC: 49%, ROC-AUC:77%	Accuracy: 74%, Recall: 80%, Precision: 51%, F1: 62%, MCC: 46%, ROC-AUC: 76%
	XGBoost	Accuracy: 81%, Recall: 52%, Precision: 69%, F1: 59%, MCC: 48%, ROC-AUC: 72%	Accuracy: 82%, Recall: 50%, Precision: 72%, F1: 59%, MCC: 49%, ROC-AUC:80%	Accuracy: 80%, Recall: 49%, Precision: 67%, F1: 56%, MCC: 45%, ROC-AUC: 70%

**Table 5 Results of Model 1**

Similar to XGBoost, a CatBoost classifier is also trained and evaluated. CatBoost is particularly known for its handling of categorical features and efficiency with large datasets. Shifting focus to the CatBoost model, the initial results indicated a more generalized performance with accuracy around 80% and recall near the 50% mark. With the help of Optuna for hyperparameter optimization, we managed to achieve a slight improvement in recall, nudging it up to almost 53%.

Despite these advances, the recall remained below our target. The challenge was attributed to the imbalanced nature of the churn class, which often misleads the model into predicting the

majority class. To counter this, we explored a combined approach of over and under sampling techniques, specifically SMOTE and Random Under Sampling, to enhance the recall. This strategy aimed to balance the dataset, providing a more equal representation of both classes and thus, a better opportunity for the model to learn the minority class characteristics. This is followed by a second round of hyperparameter tuning for the CatBoost model, again focusing on maximizing the recall score.

Preprocessing Step	Dataset	Data Size	Imbalance Ratio
Initial Data	Total	7043	No: 73.46%, Yes: 26.54%
Before Resampling	Training	4507	No: 73.47%, Yes: 26.53%
After SMOTE	Training	4507	Balanced: 50%, 50%
After Undersampling	Training	4507	Balanced: 50%, 50%
No Resampling	Validation	1127	No: 73.47%, Yes: 26.53%
No Resampling	Test	1409	No: 73.46%, Yes: 26.54%

**Table 6 Model 1 Preprocessing Overview**

The application of these sampling methods and a subsequent round of hyperparameter tuning with Optuna on the CatBoost model led to a remarkable improvement in recall, which nearly reached 80%. This increase in recall came at the expense of accuracy, which decreased slightly. However, in the context of churn prediction, the high recall rate is more valuable as it indicates a stronger capability of the model to identify actual churn cases, which is typically the primary objective in such business scenarios. This demonstrates the trade-off between accuracy and recall and underscores the importance of choosing the right metric for model evaluation based on the business problem at hand.

## Model 2

The performance of Model 1 is evaluated using a wide range of metrics, including accuracy, recall, precision, F1 score, Matthews correlation coefficient, and ROC-AUC, providing a comprehensive assessment of the model's capabilities.

On the other hand, Model 2 takes a more streamlined approach. It omits the 'gender' feature based on the analysis showing that it does not significantly impact churn rates. Additionally, Model 2 drops the 'PhoneService' feature, indicating a deliberate feature selection based on the data's characteristics. The preprocessing steps are explicitly outlined, with missing values being handled using mean imputation for numeric columns and the most frequent value for categorical columns. Categorical variables are also simplified by consolidating 'No internet service' entries to 'No', thus reducing the complexity of the model. The CatBoost classifier in Model 2 is fine-tuned using GridSearchCV to optimize specific hyperparameters, such as depth, iterations, and learning rate. Resampling techniques, including SMOTE and Random Under Sampling, are integrated within a pipeline, suggesting a methodical application during the training process. The performance metrics reported for Model 2 focus on accuracy, recall, precision, and F1 score, emphasizing the model's balanced approach to classification.

The key distinctions between the two models lie in their feature selection, preprocessing, and model optimization strategies. Model 1 appears to experiment with multiple classifiers and hyperparameter tuning methods, whereas Model 2 adopts a more focused and refined approach to model training and evaluation, targeting a balanced and simplified feature set for the CatBoost classifier.

For handling missing values, a mean imputation strategy was applied to numeric columns, while the most frequent value imputation was used for categorical columns. We then converted categorical variables into dummy variables for machine learning readiness. The CatBoost model was trained using a pipeline that incorporated both SMOTE for over-sampling and Random Under Sampling to address the class imbalance. This was crucial for improving the model's performance, especially in terms of recall.

Metric	Value
Accuracy	79%
Precision	61%
Recall	66%
F1 Score	63%

**Table 7 CatBoost Classifier Metrics**

These metrics were indicative of a model that could identify churn cases effectively, with a reasonable accuracy and a good recall rate. The confusion matrix provided further insights into the model's performance, confirming its ability to distinguish between the classes reasonably well, with a good balance between identifying true positives and minimizing false positives.

### **Model 3**

Model 3 in our churn prediction project focuses on the AdaBoost algorithm and NearMiss-3 resampling technique. We chose the AdaBoost algorithm because it is an ensemble technique that combines weak learners to create a strong learner. It iteratively adjusts the weights of incorrectly predicted instances, making it more focused on difficult cases which may be of particular interest in churn prediction. This characteristic makes AdaBoost particularly effective for our churn dataset, where identifying less obvious patterns of churn can be crucial.

NearMiss-3 was selected for its targeted approach to balancing the dataset by under-sampling the majority class based on the nearest neighbors of the minority class. This helps in creating a balance between the churn and non-churn classes, allowing our model to perform better in recognizing the churn cases.

The difference between Model 3 and the previous models lies in the specific combination of AdaBoost with NearMiss-3, which was not utilized in earlier models. Previous models either did not address the class imbalance problem or used different techniques like SMOTE for over-sampling. Model 3's approach aims to refine the model's performance by ensuring a balanced representation of both classes.

Data analysis for Model 3 involved a heatmap to identify and visualize the strength of the correlation between different features. The heatmap analysis revealed that 'tenure', 'MonthlyCharges', and 'TotalCharges' were significantly correlated with churn. High 'MonthlyCharges' and low 'TotalCharges' tended to lead to higher churn, which suggests that new customers or those with higher monthly costs are more likely to leave the service.

Preprocessing and Sampling Methods	Algorithm	Evaluation Metrics
Standard preprocessing, NearMiss-3 for class balancing, Heatmap analysis	AdaBoost	Accuracy: 78% Precision: 56% Recall: 69% F1 Score: 62%

**Table 8 Model 3 Overview**

#### **Model 4**

In the fourth model of our churn prediction project, we began with the LightGBM classifier, a choice motivated by its efficiency and effectiveness in handling large datasets, particularly those with categorical features. We applied the preprocessing methods as before and encoded the categorical data with LabelEncoder as well as scaling all the features using StandardScaler.

Post preprocessing, we applied SMOTE for balancing the dataset, followed by a split into training and testing sets. The LightGBM model was fine-tuned using RandomizedSearchCV, optimizing key parameters. This version of the model showed encouraging performance across accuracy, precision, recall, and F1 scores.

To further improve our model, we transitioned to a stacking classifier approach. The upgrades included more accurate handling of missing values in 'TotalCharges', implementing OneHotEncoder for a more nuanced processing of categorical features and continuing the use of SMOTE for class balance.

The stacking classifier integrated RandomForest, GradientBoosting, and SVC as base learners, with Logistic Regression as the meta-learner. This combination of multiple models aimed to enhance overall predictive power.

After training on the balanced dataset, the Stacking Classifier's performance on the test set indicated an improvement in accuracy, precision, recall, and F1 score over the initial LightGBM model.

This advanced model, after being trained on the balanced dataset, demonstrated an



improvement in all key performance metrics, most notably achieving an accuracy of around 0.86 on the test set. This was a significant improvement over the initial LightGBM model.

Preprocessing Method	Algorithm	Results
StandardScaler, LabelEncoder, SMOTE	LightGBM	Accuracy: 86% Precision: 85% Recall: 86% F1 Score: 86%
StandardScaler, OneHotEncoder, SMOTE	Upgraded LightGBM (Stacking Classifier with RandomForest, GradientBoosting, SVC, and Logistic Regression as meta-learner)	Accuracy: 87% Precision: 86% Recall: 86% F1 Score: 86%

**Table 9 Model 4 Overview**

### 3.2.6. Evaluation

In the evaluation phase of our churn prediction project, we conducted a series of experiments to verify the effectiveness of our models. This phase focused on assessing the models developed, particularly the XGBoost and CatBoost classifiers, and how they performed in predicting customer churn. The key metric for evaluation was recall, due to the project's emphasis on accurately identifying actual churn customers.

We applied various techniques to balance the dataset, such as SMOTE and Random Under Sampling. Model performances were scrutinized using accuracy, recall, precision, and F1 score as key metrics. The initial results showed a tendency towards overfitting, especially for the XGBoost model. However, hyperparameter tuning using HyperOpt and Optuna helped in mitigating this issue, leading to improved generalization and recall scores.

The culmination of these efforts led to significant improvements in our model's ability to predict churn. The final version of the model, especially the CatBoost classifier, demonstrated a remarkable improvement in recall, nearly reaching 80%. Although this increase in recall came at a slight expense of accuracy, it marked a successful verification of our subsystem, confirming its effectiveness in churn prediction. The final accuracy achieved was around 0.86, indicating a well-balanced model capable of reliable predictions.

## 4. INTEGRATION AND EVALUATION

In our capstone project, we navigated the intricate process of integrating and evaluating diverse sub-systems, culminating in a comprehensive solution for churn prediction and strategic marketing. This phase represented the culmination of collaborative efforts between Computer and Management Engineering teams. It involved harmonizing technical model development with practical business strategy applications. Our focus was on ensuring the seamless functionality of predictive algorithms and their alignment with real-world market dynamics. The subsequent evaluation meticulously assessed the system's accuracy and efficacy, leading to an optimized tool for data-driven decision-making and customer retention strategies.

### 4.1. Integration

The integration of the Management Engineering and Computer Engineering subsystems has been completed to achieve the project's goals. This collaboration between the two subteams has enabled the deployment of precise churn prediction models and the formulation of successful marketing plans.

Smooth integration was ensured through essential cooperation and communication among the sub-teams.

As the lead department, the Management Engineering team has successfully implemented ML algorithms and evaluated their outcomes. Specializing in analyzing consumer behavior and applying ML algorithms for churn prediction, they have gained valuable insights into client segmentation and churn prediction accuracy through their examination of the ML results.

On the other hand, the responsibility of the Computer Engineering department included data preprocessing and addressing class imbalance issues. They proficiently employed machine learning algorithms and programming languages like Python, Pandas, NumPy, and scikit-learn, in addition to advanced algorithms from CatBoost and LightGBM for training models. The subsystem utilizes Jupyter Notebooks for iterative development and visualization, managing data through CSV files, which eliminates the necessity for intricate database systems in the preprocessing of data, balancing datasets, and creating machine learning algorithms for churn prediction.

Regular coordination discussions and effective communication between the two departments were generated to ensure a seamless integration. The Computer Engineering department received input and demands from the Management Engineering department based on ML findings, guiding their preprocessing and algorithm implementation efforts.

Loops for exchanging data and receiving comments were established to encourage ongoing development. The Management Engineering team received updated ML models and data preparation insights from the Computer Engineering team. This reciprocal exchange allowed both departments to assess the performance of implemented initiatives, leading to improvements in methods and the overall system. The culmination of this integration phase marked a significant achievement in our capstone project. It demonstrated the power of interdisciplinary collaboration, where the synergy between technical expertise and strategic insights led to the creation of a robust system that effectively addresses the complexities of churn prediction. The successful integration of these two subsystems has not only resulted in the development of advanced predictive models but also in the establishment of practical marketing strategies that are deeply rooted in data-driven analysis. This synergy ensures that the project's outcomes are not only theoretically sound but also practically viable, offering real-world solutions to the challenges faced by the telecommunications industry in managing customer churn.

## **4.2. Evaluation**

Following the completion of the specified tasks, a comprehensive report has been compiled detailing the accomplished objectives. The assessment of the integrated system's attributes and performance was conducted based on meticulously planned tests and insightful data analysis. It is noteworthy that the various departments successfully fulfilled their designated roles, aligning with the initially outlined expectations.

The Management Engineering team, in particular, meticulously evaluated the impact of campaigns and promotions on consumer behavior. Consumer response rates were thoroughly assessed to gauge the effectiveness of the implemented strategies. To further validate the success of client churn tactics, a comparative analysis of churn rates before and after integration was conducted.

Conversely, the Computer Engineering department concentrated on evaluating the efficacy of data preparation methods and addressing the challenge of class imbalance. Emphasis was placed on understanding how these methods influenced the quality of the dataset and the accuracy of subsequent machine learning models. Key metrics such as data balance, feature selection, and model training effectiveness were focal points in this evaluation.

In the refinement of our churn prediction project, we employed advanced algorithms and libraries to enhance the effectiveness of our models. Notably, we integrated well-established classifiers, such as XGBoost and CatBoost, during the evaluation phase to gauge their performance in predicting customer churn accurately. The evaluation primarily centered around the key metric of recall, aligning with the project's emphasis on identifying actual churn customers with precision.

To address potential biases and improve the performance of our models, we strategically employed data balancing techniques. Specifically, techniques like SMOTE (Synthetic Minority Over-sampling Technique) and Random Under Sampling were instrumental in rectifying imbalances within the dataset, contributing significantly to the enhancement of recall scores.

The evaluation metrics encompassed not only accuracy but also recall, precision, and F1 score, providing a comprehensive overview of model performances. Initial results indicated a tendency towards overfitting, particularly in the XGBoost model. However, through meticulous hyperparameter tuning facilitated by HyperOpt and Optuna, we successfully mitigated overfitting issues, leading to improved generalization and recall scores.

The culmination of these efforts resulted in substantial enhancements to our churn prediction model. The final version, notably the CatBoost classifier, exhibited a remarkable improvement in recall, nearly reaching an impressive 80%. While this increase in recall did incur a slight reduction in accuracy, it validated the successful verification of our subsystem's effectiveness in predicting churn reliably. The final accuracy achieved stood at approximately 0.86, underscoring the development of a well-balanced model capable of delivering accurate and dependable predictions. Incorporating these algorithmic insights further strengthens our overall assessment of the integrated system's capabilities.

The results of the assessment have been disseminated to both departments and comprehensively documented. The documentation encapsulates vital conclusions, revelations, and suggestions for potential improvements in future iterations of the system. This record will serve as a valuable resource for ongoing enhancements to machine learning algorithms, data preprocessing methods, and marketing strategies.

In conclusion, the successful collaboration between the Management Engineering and Computer Engineering departments played a pivotal role in both the integration and assessment phases of our capstone project. By leveraging their combined expertise and scrutinizing the features and performance of the integrated system, data-driven decision-making has been facilitated. This approach is poised to enhance the accuracy of churn prediction models and optimize the planning of promotions and campaigns in future iterations of the system. This record will serve as a valuable resource for ongoing enhancements to machine learning algorithms, data preprocessing methods, and marketing strategies.

In conclusion, the successful collaboration between the Management Engineering and Computer Engineering departments played a pivotal role in both the integration and assessment phases of our capstone project. By leveraging their combined expertise and scrutinizing the features and performance of the integrated system, data-driven decision-making has been facilitated. This approach is poised to enhance the accuracy of churn prediction models and optimize the planning of

promotions and campaigns in future iterations of the system. Moving forward, the project team will continue to refine their methodologies, adapting to new data and market trends to ensure sustained effectiveness in customer churn prediction and management.

## **5. SUMMARY AND CONCLUSION**

In our capstone project, we successfully navigated through various phases including data preprocessing, strategic model selection, and performance enhancement. During materialization, we focused on refining data and feature engineering. Integration highlighted the combined strengths of Computer and Management Engineering, enhancing prediction models and formulating effective marketing strategies.

Verification was rigorously conducted through experiments, ensuring our models met functional and performance requirements. A significant achievement was the CatBoost classifier reaching an impressive 80% recall rate, despite some accuracy trade-offs.

Our project's success was rooted in effective data handling, model optimization, and tailored marketing campaigns, addressing challenges such as balancing model accuracy with recall. Future developments of this project can be directed towards deeper feature analysis, employing advanced predictive algorithms, and refining marketing strategies for better customer retention and engagement.

As we move forward, the focus will also be on integrating real-time data analytics to adapt strategies dynamically, enhancing our system's ability to respond to market trends and customer behavior shifts. Continuous learning and adaptation will be key in maintaining the relevance and effectiveness of our churn prediction models. The integration insights into marketing strategies will be explored to further personalize customer interactions and increase engagement. The ultimate goal remains to provide a scalable, efficient solution for churn prediction and management, not only serving as a model for the telecommunications industry but also adaptable for broader applications in various sectors.

## **ACKNOWLEDGEMENTS**

We wish to thank our advisers ASSIST. PROF. ASLI AYDIN and ASSIST. PROF. OMER MELIH GUL for guiding us through this proposal.

## REFERENCES

1. Clement Kirui, LiHong, Edgar Kirui (2013). Handling Class Imbalance in Mobile Telecoms Customer Churn Prediction. Volume 72– No.23, June 2013.
2. Amin, A., Rahim, F., Ali, I., Khan, C., Anwar, S. (2015). A Comparison of Two Oversampling Techniques (SMOTE vs MTDF) for Handling Class Imbalance Problem: A Case Study of Customer Churn Prediction. In: Rocha, A., Correia, A., Costanzo, S., Reis, L. (eds) *New Contributions in Information Systems and Technologies. Advances in Intelligent Systems and Computing*, vol 353. Springer, Cham.
3. A. Ahmed and D. M. Linen, "A review and analysis of churn prediction methods for customer retention in telecom industries," *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2017, pp. 1-7, doi: 10.1109/ICACCS.2017.8014605.
4. S. Shumaly, P. Neysaryan and Y. Guo, "Handling Class Imbalance in Customer Churn Prediction in Telecom Sector Using Sampling Techniques, Bagging and Boosting Trees," *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, 2020, pp. 082-087, doi: 10.1109/ICCKE50421.2020.9303698.
5. J. Burez, D. Van den Poel, Handling class imbalance in customer churn prediction, *Expert Systems with Applications*, Volume 36, Issue 3, Part 1, 2009.
6. Yang, L., Li, D., & Lu, Y. (2019). Prediction Modeling and Analysis for Telecom Customer Churn in Two Months. Retrieved from arXiv:1911.00558 [cs.IR]. (or arXiv:1911.00558v1 [cs.IR] for this version)
7. Lalwani, P., Mishra, M.K., Chadha, J.S. *et al.* Customer churn prediction system: a machine learning approach. *Computing* **104**, 271–294 (2022).
8. Doxee, 4 strategies to reduce churn rate in telco industry [online]. <https://www.doxee.com/blog/customer-experience/4-strategies-to-reduce-churn-rate-in-the-telco-industry/>
9. Mohiuddin Ahmed, Raihan Seraj, Syed M. S. Islam, The *k-means* Algorithm: A Comprehensive Survey and Performance Evaluation, (August, 2020). <https://www.mdpi.com/2079-9292/9/8/1295>.
10. Essam Al Daoud, Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset, (Jan, 2007). <https://publications.waset.org/10009954/comparison-between-xgboost-lightgbm-and-catboost-using-a-home-credit-dataset>
11. Adar Amit, "Customer Churn – Working With Imbalanced Dataset" From Kaggle.com: Telecom Customer Churn Prediction. <https://www.kaggle.com/code/adaramit/customer->

churn-working-with-imbalanced-dataset/notebook.

Use the **IEEE style** when listing references. Try to add MINIMUM 10 references.

A good guide can be found here: <http://libguides.murdoch.edu.au/IEEE/>,

and many examples here: <https://libguides.murdoch.edu.au/IEEE/all>

## APPENDIX

The screenshot displays a Jupyter Notebook environment with the following components:

- Code Cell:** Contains Python code for importing libraries (pandas, numpy, matplotlib, seaborn, imblearn, sklearn, xgboost) and loading the dataset.
- Output Cell:** Shows the result of the code execution, including the first five rows of the dataset and its dimensions.

**Code:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from imblearn.under_sampling import NearMiss
from imblearn.over_sampling import SMOTE

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
import xgboost as xgb
```

**Output:**

```
data = pd.read_excel('Telco_Cust_Churn.xls')
data.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month to-month
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year
2	3868-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month to-month
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month to-month

5 rows x 21 columns

**Footer:** Jupyter Server: Local | Ln 3, Col 14 | Spaces: 4 | LF | Cell 2 of 27



```

churn_prediction.ipynb • churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
churn_prediction.ipynb > data = pd.read_excel('Telco_Cust_Churn.xls')
+ Code + Markdown + Run All + Clear Outputs of All Cells + Restart + Variables + Outline ... churn2_env (Python 3.9.13)

data.info()

[9] Python

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   customerID            7043 non-null   object
1   gender                7043 non-null   object
2   SeniorCitizen         7043 non-null   int64
3   Partner               7043 non-null   object
4   Dependents            7043 non-null   object
5   tenure                7043 non-null   int64
6   PhoneService          7043 non-null   object
7   MultipleLines         7043 non-null   object
8   InternetService       7043 non-null   object
9   OnlineSecurity        7043 non-null   object
10  OnlineBackup          7043 non-null   object
11  DeviceProtection      7043 non-null   object
12  TechSupport           7043 non-null   object
13  StreamingTV           7043 non-null   object
14  StreamingMovies       7043 non-null   object
15  Contract              7043 non-null   object
16  PaperlessBilling      7043 non-null   object
17  PaymentMethod         7043 non-null   object
18  MonthlyCharges        7043 non-null   float64
19  TotalCharges          7043 non-null   object
20  Churn                 7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB

data = data.replace('.', value=0)
data.TotalCharges = pd.to_numeric(data.TotalCharges)
data.TotalCharges.dtype

[10] Python

```

```

churn_prediction.ipynb • churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
churn_prediction.ipynb > data = pd.read_excel('Telco_Cust_Churn.xls')
+ Code + Markdown + Run All + Clear Outputs of All Cells + Restart + Variables + Outline ... churn2_env (Python 3.9.13)


dtype('float64')

[11] Python

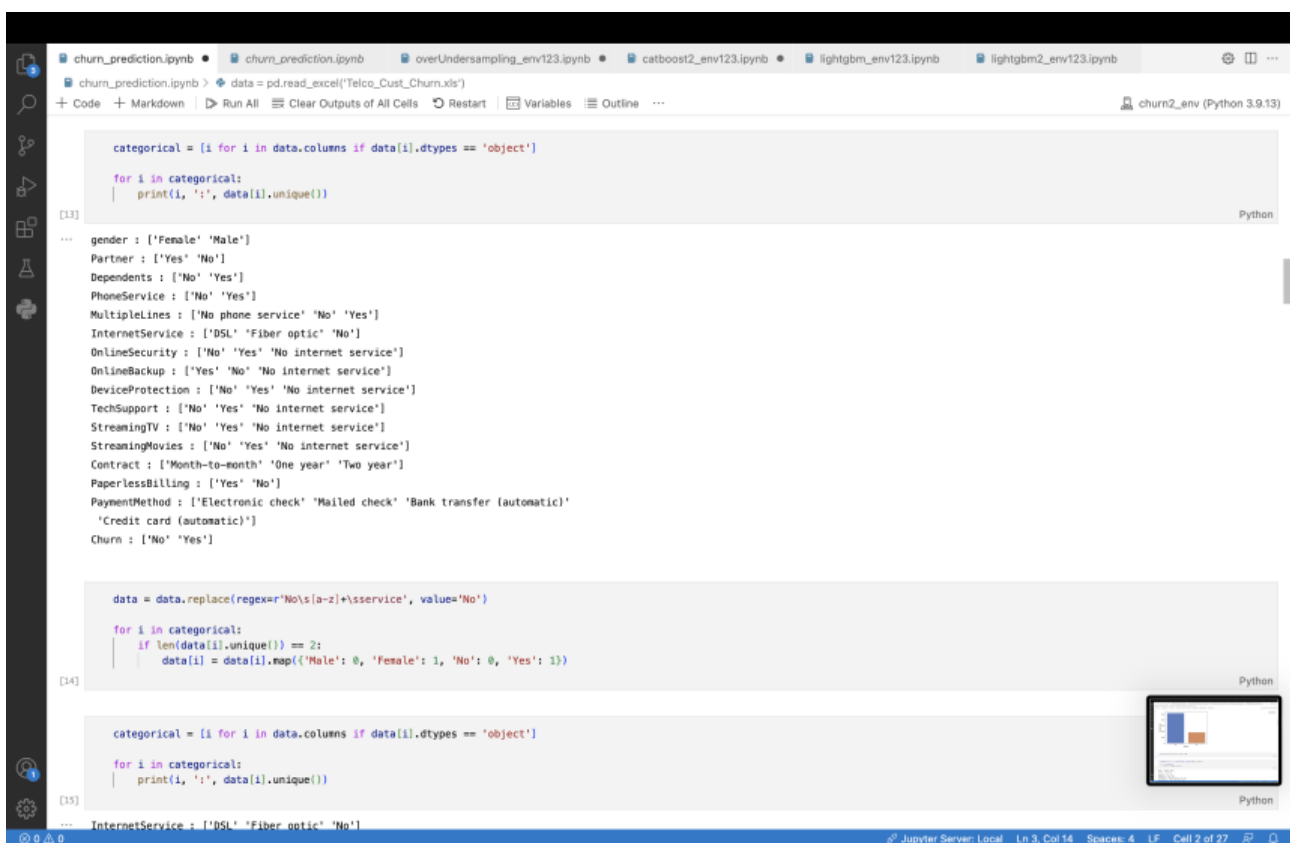
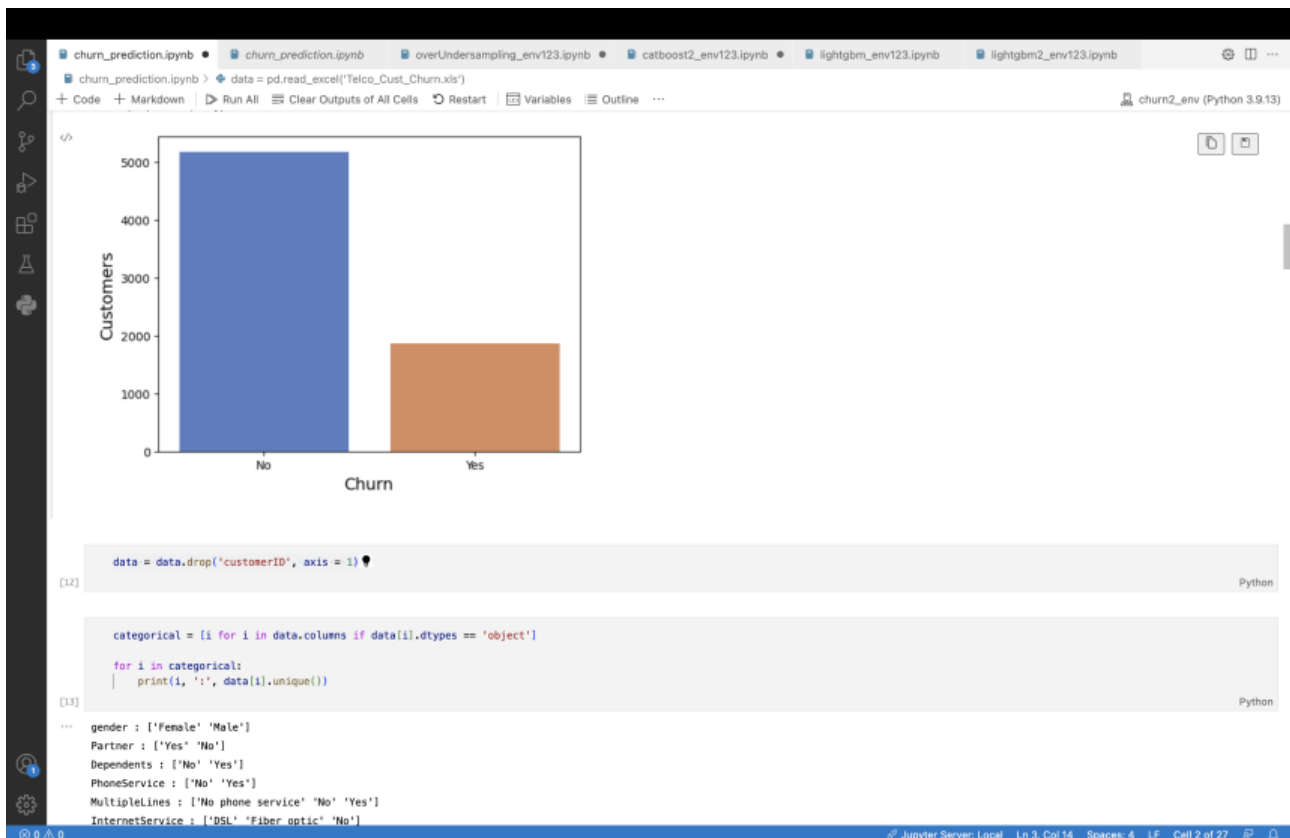
g = sns.countplot(x="Churn", data=data, palette="muted")
g.set_ylabel("Customers", fontsize=14)
g.set_xlabel("Churn", fontsize=14)
data.Churn.value_counts(normalize=True)

/Users/laraturunc/Desktop/churnPredictionProject/churn2_env/lib/python3.9/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
if pd.api.types.is_categorical_dtype(vector):
/Users/laraturunc/Desktop/churnPredictionProject/churn2_env/lib/python3.9/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
if pd.api.types.is_categorical_dtype(vector):
/Users/laraturunc/Desktop/churnPredictionProject/churn2_env/lib/python3.9/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
if pd.api.types.is_categorical_dtype(vector):

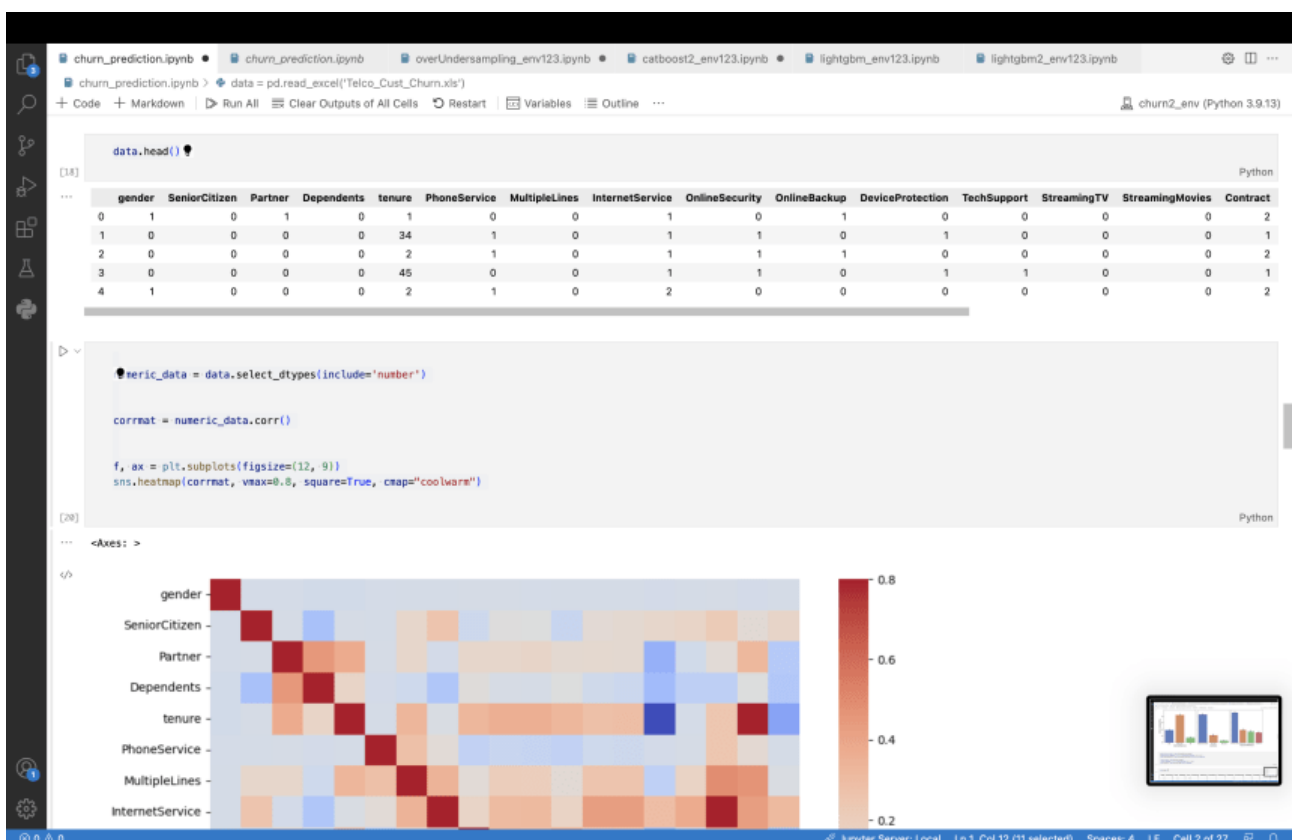
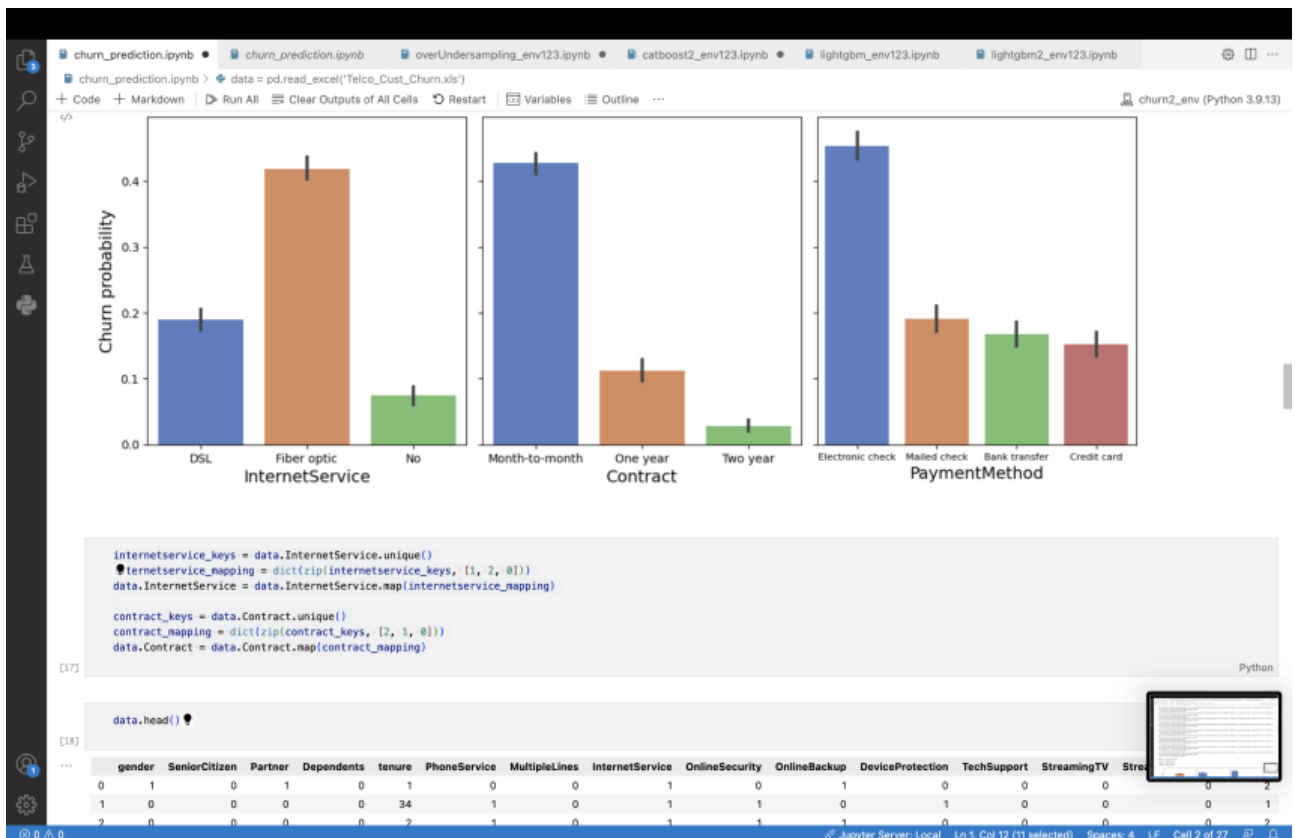
Churn
No    0.73463
Yes   0.26537
Name: proportion, dtype: float64

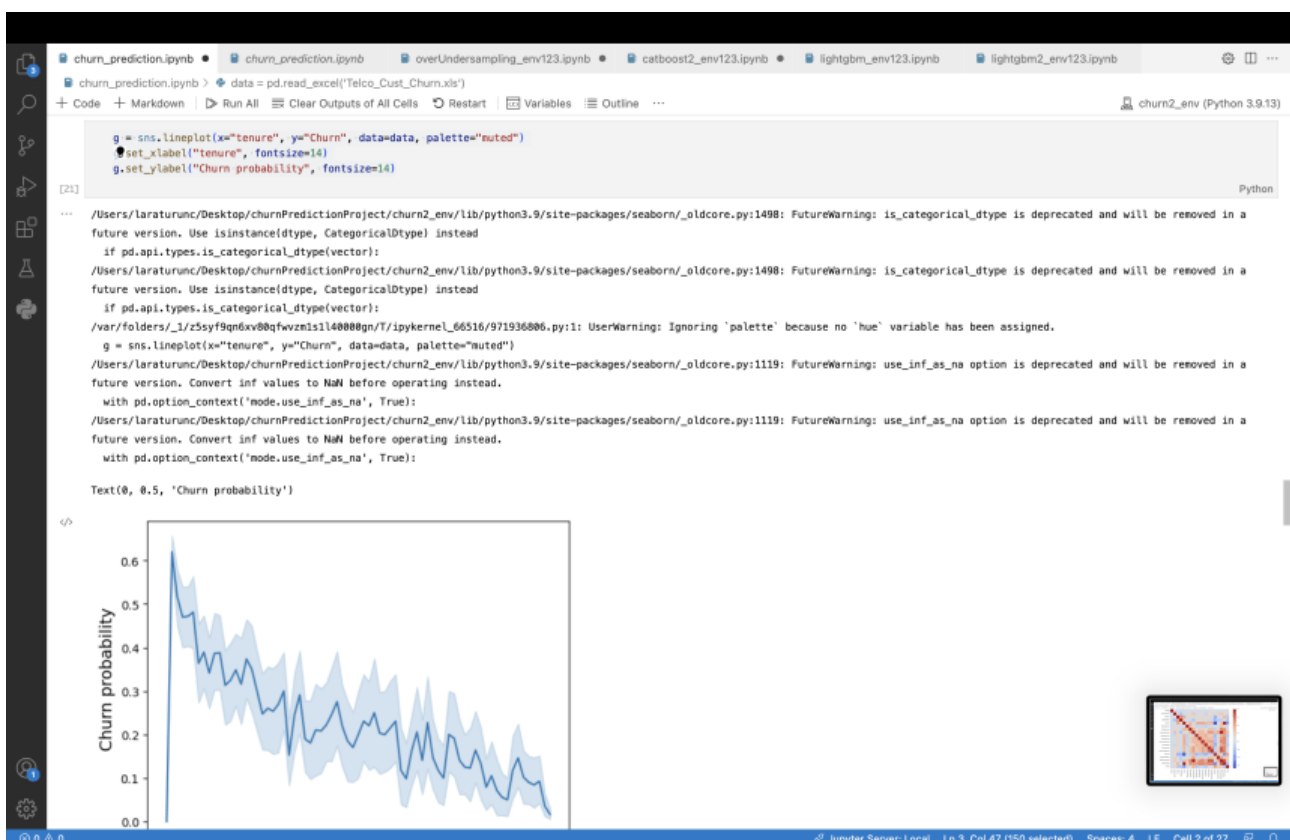
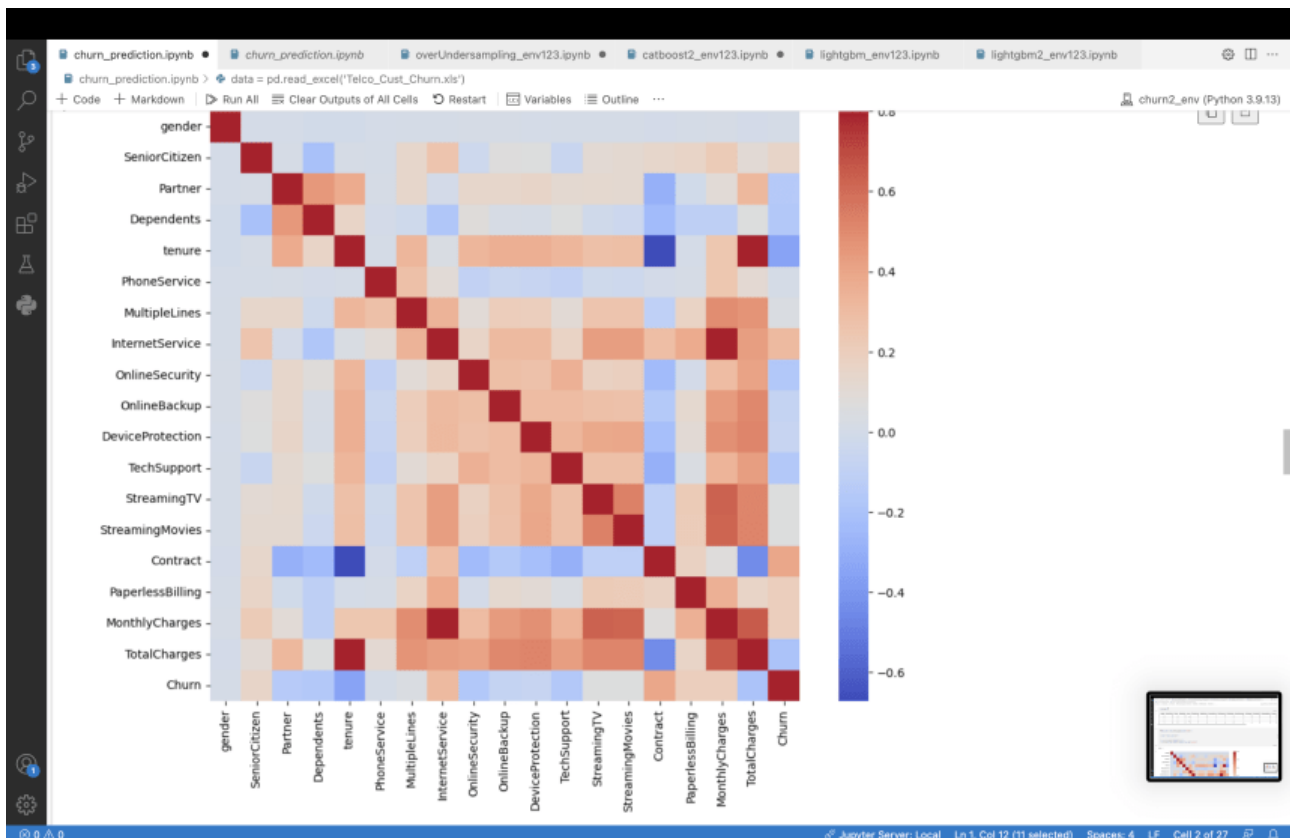
</>


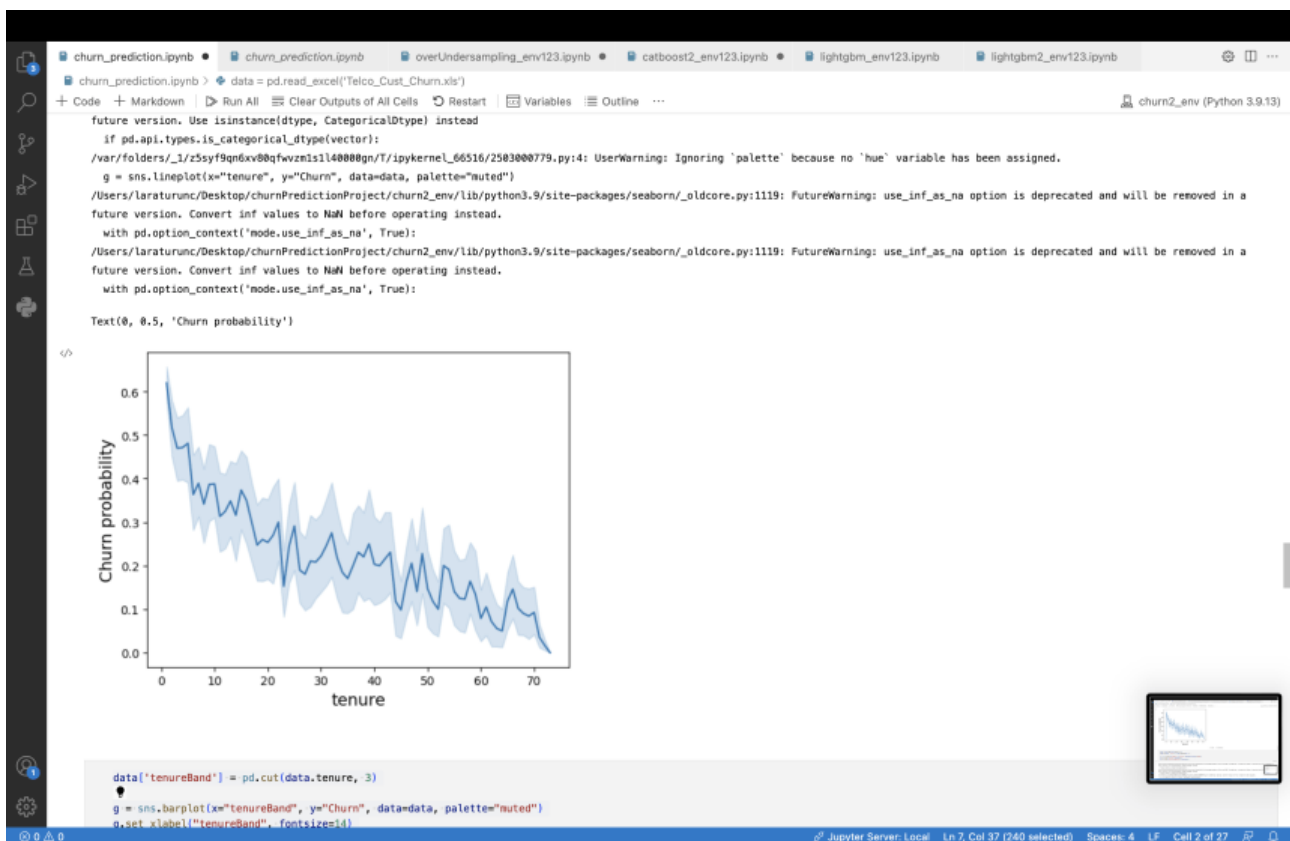
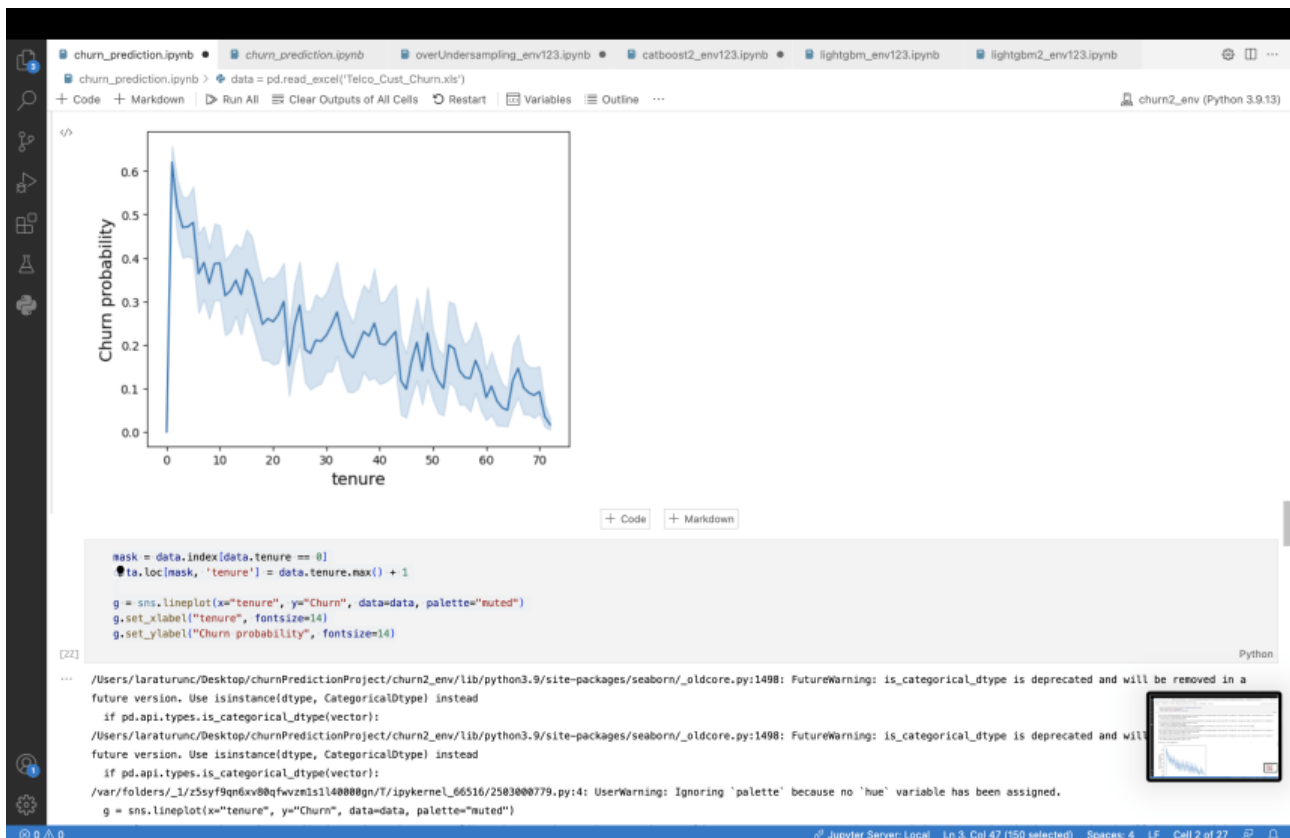
```

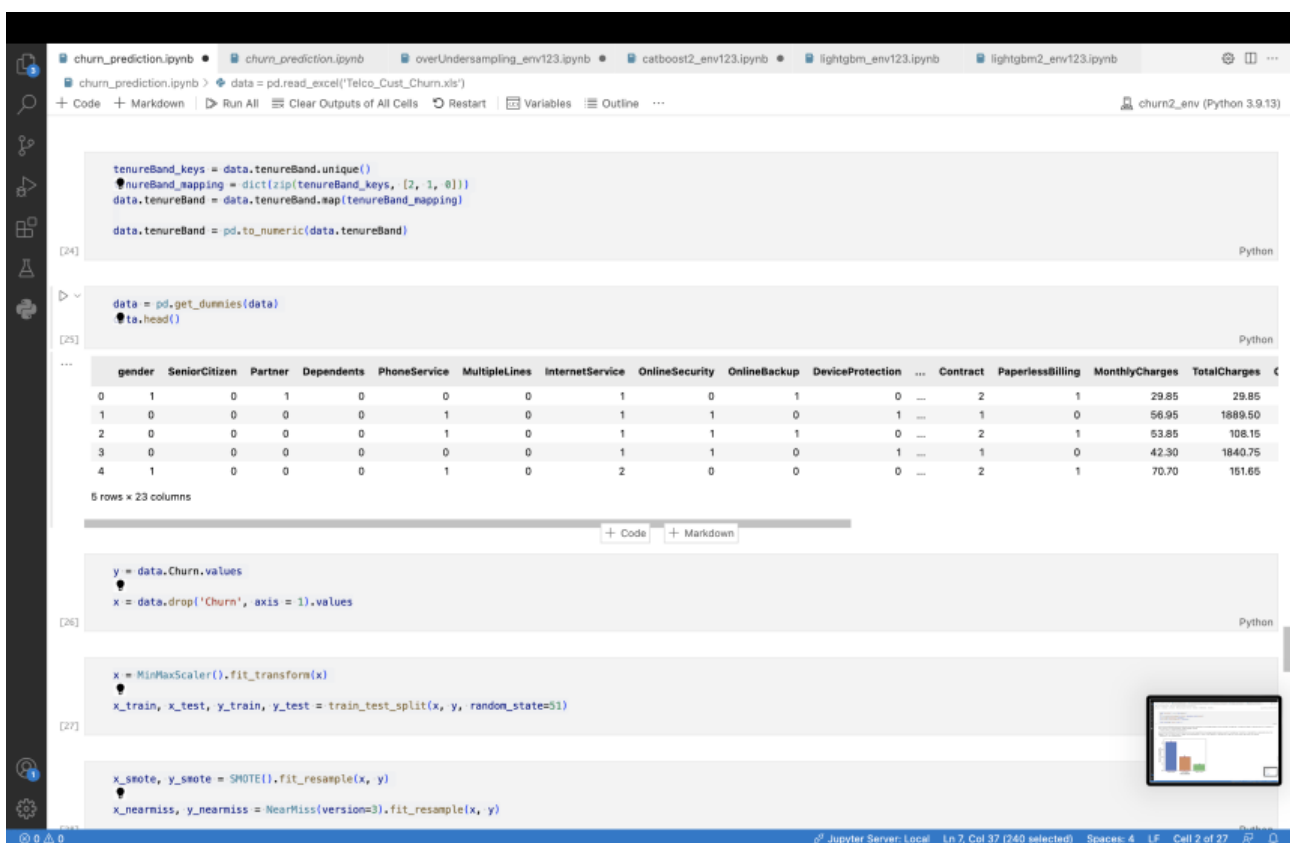
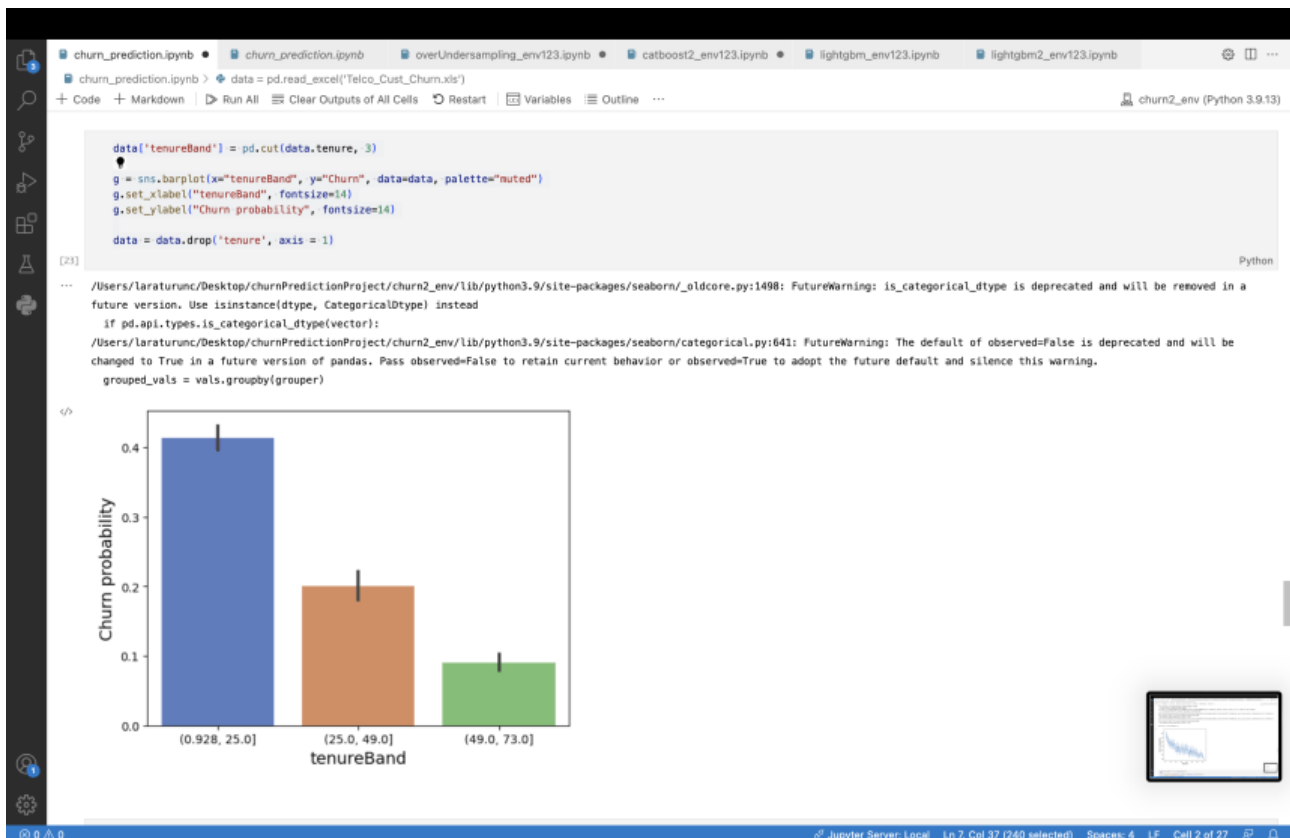












```

churn_prediction.ipynb • churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
churn_prediction.ipynb > data = pd.read_excel('Telco_Cust_Churn.xls')
+ Code + Markdown + Run All + Clear Outputs of All Cells + Restart + Variables + Outline ... churn2_env (Python 3.9.13)

[28]
x_smote, y_smote = SMOTE().fit_resample(x, y)
x_nearmiss, y_nearmiss = NearMiss(version=3).fit_resample(x, y)

Python

[29]
classifiers = []
classifiers.append(LogisticRegression(max_iter=500))
classifiers.append(RandomForestClassifier())
classifiers.append(AdaBoostClassifier())
classifiers.append(xgb.XGBClassifier())
classifiers.append(SVC(C = 10))

Python

[30]
def cross_validate(classifiers, x_train, y_train, x_test, y_test):
    cv_results = []
    for classifier in classifiers:
        cv_results.append([cross_val_score(classifier, x_train, y_train,
                                           scoring = "f1_weighted").mean()])
    best_clf = classifiers[np.argmax(cv_results)]
    return best_clf

best_clf_wo_balancing = cross_validate(classifiers, x_train, y_train, x_test, y_test)
best_clf_smote = cross_validate(classifiers, x_smote, y_smote, x_test, y_test)
best_clf_nearmiss = cross_validate(classifiers, x_nearmiss, y_nearmiss, x_test, y_test)

Python

[31]
def print_best_clf_results(best_clf, x_train, y_train, x_test, y_test):
    print(f'\n\nThe classifier with the best f1 result is: (best_clf)')
    best_clf.fit(x_train, y_train)
    y_pred = best_clf.predict(x_test)
    print('\n\nClassification Report:')
    print(classification_report(y_test, y_pred))
    print('\n\n')
    f_i = best_clf.feature_importances_
    return f_i

```

```

churn_prediction.ipynb • churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
churn_prediction.ipynb > data = pd.read_excel('Telco_Cust_Churn.xls')
+ Code + Markdown + Run All + Clear Outputs of All Cells + Restart + Variables + Outline ... churn2_env (Python 3.9.13)

[34]
print('~'*70)
print('Without balancing:')
print('~'*70)
f_i = print_best_clf_results(best_clf_wo_balancing, x_train, y_train, x_test, y_test)

print('~'*70)
print('Over-sampling using SMOTE technique:')
print('~'*70)
f_i_smote = print_best_clf_results(best_clf_smote, x_smote, y_smote, x_test, y_test)

print('~'*70)
print('Under-sampling using NearMiss3 technique:')
print('~'*70)
f_i_nearmiss = print_best_clf_results(best_clf_nearmiss, x_nearmiss, y_nearmiss, x_test, y_test)

Python

Output exceeds the size limit. Open the full output data in a text editor

Without balancing:

-----

The classifier with the best f1 result is: AdaBoostClassifier()

Classification Report:
      precision    recall  f1-score   support

      0       0.84      0.90      0.87      1300
      1       0.64      0.52      0.58       461

 accuracy      0.80      0.80      0.80      1761
 macro avg      0.74      0.71      0.72      1761
 weighted avg      0.79      0.80      0.79      1761

-----

Over-sampling using SMOTE technique:

-----

```



churn\_prediction.ipynb • churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

churn\_prediction.ipynb > data = pd.read\_excel('Telco\_Cust\_Churn.xls')

+ Code + Markdown ▶ Run All Clear Outputs of All Cells Restart Variables Outline ... churn2\_env (Python 3.9.13)

```
col_names = data.columns.drop('Churn')
feature_importances = pd.DataFrame({'Feature': col_names, 'Feature Importances': f_i_snote})
feature_importances = feature_importances.sort_values('Feature Importances', ascending=False, ignore_index=True).head()

feature_importances = feature_importances.style.set_properties(**{'text-align': 'left'})
feature_importances = feature_importances.set_table_styles([dict(selector='th', props=[('text-align', 'left')])])
feature_importances
```

[35]

	Feature	Feature Importances
0	TotalCharges	0.166127
1	MonthlyCharges	0.159528
2	Contract	0.153169
3	tenureBand	0.080934
4	InternetService	0.078548

Python

Python

Jupyter Server: Local Ln 10, Col 1 (478 selected) Spaces: 4 LF Cell 2 of 27

churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

overUndersampling\_env123.ipynb > # Churn columnunu 1 ve 0 yaptım

+ Code + Markdown ▶ Run All Clear Outputs of All Cells Restart Variables Outline ... env\_123 (Python 3.8.18)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]

```
data = pd.read_csv('/Users/laraturunc/Desktop/churn_data/Telco-Customer-Churn-Dataset.csv')
```

[3]

```
data.head()
```

[4]

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month to-month
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year
2	3868-QPVBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month to-month
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month to-month

5 rows x 21 columns

```
data.info()
```

[5]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7843 entries, 0 to 7842
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   customerID            7843 non-null   object
1   gender                7843 non-null   object
```

Python

Jupyter Server: Local Cell 6 of 26

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptım
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)
data.info()
[5]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7843 entries, 0 to 7842
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   customerID            7843 non-null   object
1   gender                7843 non-null   object
2   SeniorCitizen         7843 non-null   int64
3   Partner               7843 non-null   object
4   Dependents            7843 non-null   object
5   tenure                7843 non-null   int64
6   PhoneService          7843 non-null   object
7   MultipleLines         7843 non-null   object
8   InternetService       7843 non-null   object
9   OnlineSecurity        7843 non-null   object
10  OnlineBackup          7843 non-null   object
11  DeviceProtection      7843 non-null   object
12  TechSupport           7843 non-null   object
13  StreamingTV           7843 non-null   object
14  StreamingMovies       7843 non-null   object
15  Contract              7843 non-null   object
16  PaperlessBilling      7843 non-null   object
17  PaymentMethod         7843 non-null   object
18  MonthlyCharges        7843 non-null   float64
19  TotalCharges          7843 non-null   object
20  Churn                 7843 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB

data = data.drop(columns=['customerID'])
[6]

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptım
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)
# Churn columnunu 1 ve 0 yaptım
data['Churn'] = data['Churn'].map({'No': 0, 'Yes': 1})

print(data['Churn'].value_counts())
[7]
Churn
0    5174
1    1869
Name: count, dtype: int64

target_distribution = data['Churn'].value_counts()
int(target_distribution)
[8]
Churn
0    5174
1    1869
Name: count, dtype: int64

missing_values = data.isnull().sum()
#missing value var mı kontrol
missing_values_total_charges = data['TotalCharges'].isnull().sum()

print("Missing values in the entire DataFrame:")
print(missing_values)

print("\nMissing values in the 'TotalCharges' column:")
print(missing_values_total_charges)
[9]
Missing values in the entire DataFrame:
gender                0
SeniorCitizen        0
Partner              0

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptım
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells 🔄 Restart 📄 Variables 📄 Outline ... env_123 (Python 3.8.18)

missing_values = data.isnull().sum()
#missing value var mı kontrol
missing_values_total_charges = data['TotalCharges'].isnull().sum()

print("Missing values in the entire DataFrame:")
print(missing_values)

print("\nMissing values in the 'TotalCharges' column:")
print(missing_values_total_charges)

```

[9] Python

```

... Missing values in the entire DataFrame:
gender                0
SeniorCitizen         0
Partner              0
Dependents            0
tenure                0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges         0
Churn                0
dtype: int64

Missing values in the 'TotalCharges' column:
0

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptım
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells 🔄 Restart 📄 Variables 📄 Outline ... env_123 (Python 3.8.18)

summary_stats = data.describe()
int(summary_stats)

```

[10] Python

```

...
      SeniorCitizen      tenure  MonthlyCharges      Churn
count  7843.000000  7843.000000  7843.000000  7843.000000
mean    0.162147    32.371149    64.761692    0.265370
std     0.368612    24.559481    38.890047    0.441561
min     0.000000     0.000000    18.250000    0.000000
25%     0.000000     9.000000    35.500000    0.000000
50%     0.000000    29.000000    78.350000    0.000000
75%     0.000000    55.000000    89.850000    1.000000
max      1.000000    72.000000   118.750000    1.000000

```

```

>
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Churn'] = le.fit_transform(data['Churn'])

```

[11] Python

```

numerical= data.select_dtypes('number').columns
categorical = data.select_dtypes('object').columns

print(f'Numerical Columns: {data[numerical].columns}')
print("\n")
print(f'Categorical Columns: {data[categorical].columns}')

```

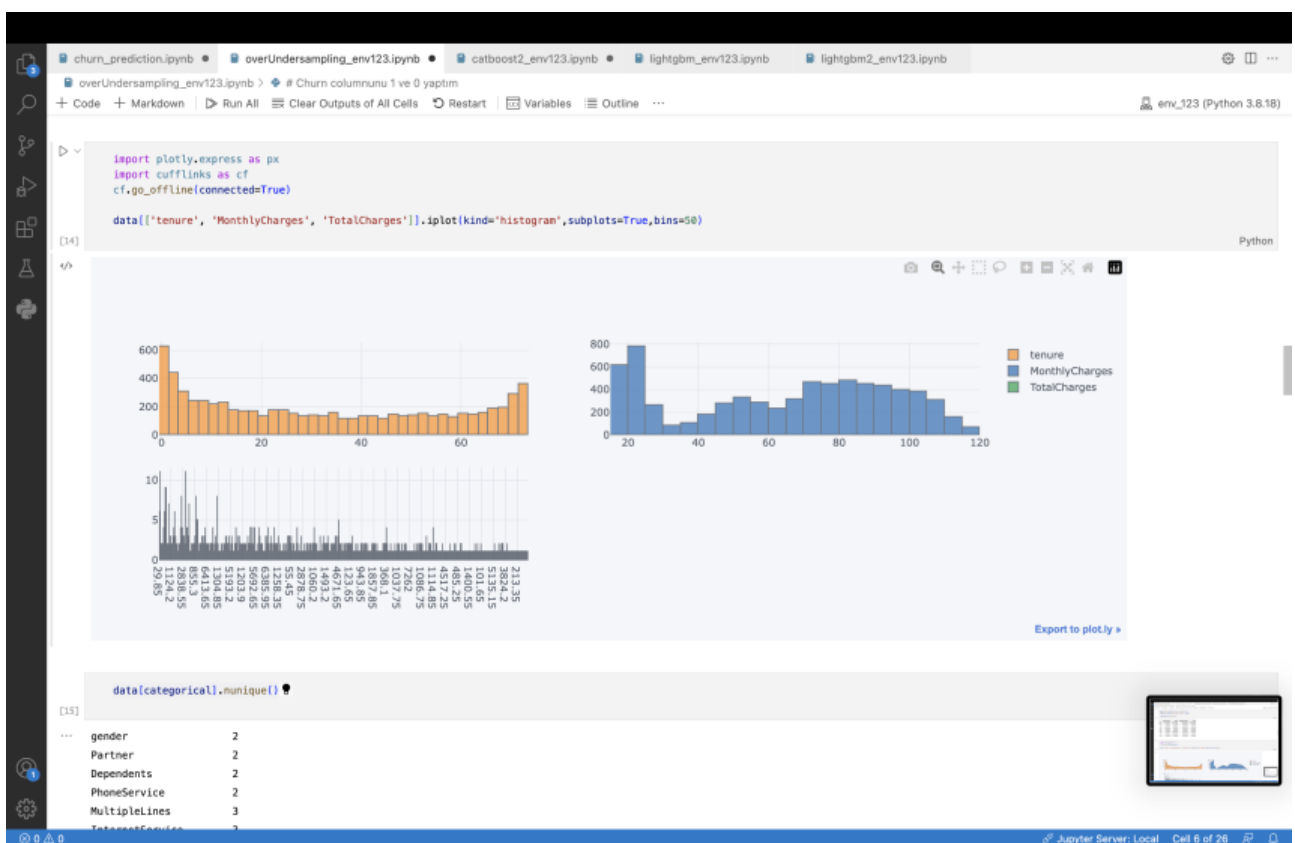
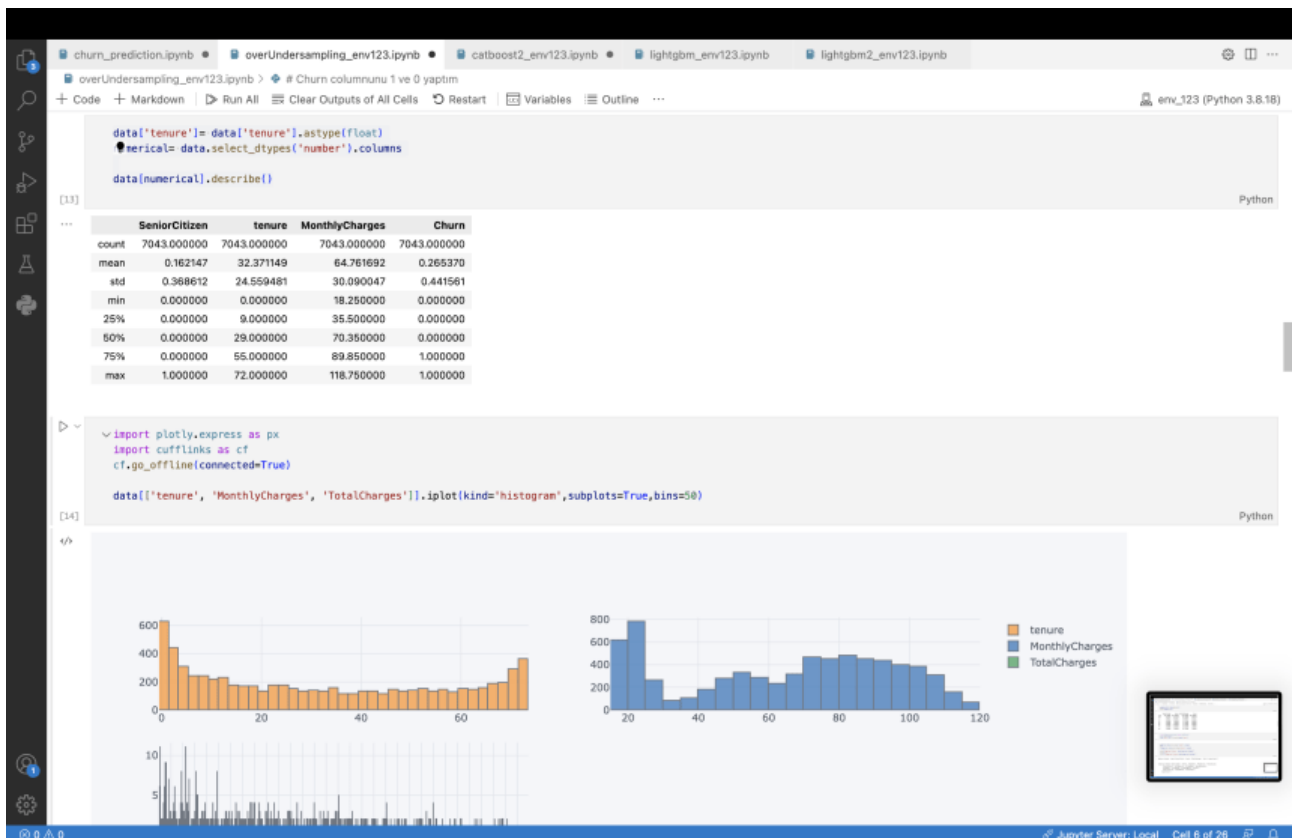
[12] Python

```

... Numerical Columns: Index(['SeniorCitizen', 'tenure', 'MonthlyCharges', 'Churn'], dtype='object')

Categorical Columns: Index(['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
'PaperlessBilling', 'PaymentMethod', 'TotalCharges'],
dtype='object')

```



```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptim
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells ⌂ Restart 📄 Variables 📄 Outline ... env_123 (Python 3.8.18)

data[categorical].nunique()
[15]
Python
gender      2
Partner     2
Dependents  2
PhoneService 2
MultipleLines 3
InternetService 3
OnlineSecurity 3
OnlineBackup 3
DeviceProtection 3
TechSupport 3
StreamingTV 3
StreamingMovies 3
Contract 3
PaperlessBilling 2
PaymentMethod 4
TotalCharges 6531
dtype: int64

for feature in data[categorical]:
    print(f'{feature}: {data[feature].nunique()}')
[16]
Python
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No phone service' 'No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes' 'No internet service']
OnlineBackup: ['Yes' 'No' 'No internet service']
DeviceProtection: ['No' 'Yes' 'No internet service']
TechSupport: ['No' 'Yes' 'No internet service']
StreamingTV: ['No' 'Yes' 'No internet service']
StreamingMovies: ['No' 'Yes' 'No internet service']

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptim
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells ⌂ Restart 📄 Variables 📄 Outline ... env_123 (Python 3.8.18)

for feature in data[categorical]:
    print(f'{feature}: {data[feature].nunique()}')
[16]
Python
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No phone service' 'No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes' 'No internet service']
OnlineBackup: ['Yes' 'No' 'No internet service']
DeviceProtection: ['No' 'Yes' 'No internet service']
TechSupport: ['No' 'Yes' 'No internet service']
StreamingTV: ['No' 'Yes' 'No internet service']
StreamingMovies: ['No' 'Yes' 'No internet service']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
'Credit card (automatic)']
TotalCharges: ['29.85' '1889.5' '188.15' ... '346.45' '306.6' '6844.5']

print(f'A female customer has a probability of {round(data[data["gender"] == 0]["Churn"].mean() * 100, 2)} % churn')
print()
print(f'A male customer has a probability of {round(data[data["gender"] == 1]["Churn"].mean() * 100, 2)} % churn')
print()
[17]
Python
A female customer has a probability of nan % churn

A male customer has a probability of nan % churn

```



churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

overUndersampling\_env123.ipynb > # Churn columnunu 1 ve 0 yaptım

```
data["MultipleLines"] = data["MultipleLines"].replace('No phone service', 'No')
data.head()
```

[20] Python

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract
0	Female	0	Yes	No	1.0	No	No	DSL	No	Yes	No	No	No	No	Month-to-month
1	Male	0	No	No	34.0	Yes	No	DSL	Yes	No	Yes	No	No	No	One year
2	Male	0	No	No	2.0	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-month
3	Male	0	No	No	45.0	No	No	DSL	Yes	No	Yes	Yes	No	No	One year
4	Female	0	No	No	2.0	Yes	No	Fiber optic	No	No	No	No	No	No	Month-to-month

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, precision_score
from catboost import CatBoostClassifier
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline

accuracy = []
recall = []
roc_auc = []
precision = []
model_names = []

data = pd.read_csv('/Users/laraturunc/Desktop/churn_data/Telco-Customer-Churn-Dataset.csv')
data = data.drop(['customerID', 'gender', 'PhoneService'], axis=1).copy()
le = LabelEncoder()
data['Churn'] = le.fit_transform(data['Churn'])

data['tenure'] = data['tenure'].astype(float)
data['TotalCharges'] = data['TotalCharges'].apply(lambda x: x if x != '' else np.nan).astype(float)

data[['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']] = \
data[['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']] \

```

Jupyter Server: Local Cell 6 of 26

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptim
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ...
env_123 (Python 3.8.18)

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, precision_score
from catboost import CatBoostClassifier
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline

accuracy = []
recall = []
roc_auc = []
precision = []
model_names = []

data = pd.read_csv('/Users/laraturunc/Desktop/churn_data/Telco-Customer-Churn-Dataset.csv')
data1 = data.drop(['customerID', 'gender', 'PhoneService'], axis=1).copy()
le = LabelEncoder()
data1['Churn'] = le.fit_transform(data1['Churn'])

data1['tenure'] = data1['tenure'].astype(float)
data1['TotalCharges'] = data1['TotalCharges'].apply(lambda x: x if x != '' else np.nan).astype(float)

data1[['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']] = \
    data1[['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']] \
    .replace('No internet service', 'No')

X = data1.drop('Churn', axis=1)
y = data1['Churn']

numeric_columns = X.select_dtypes(include=np.number).columns
categorical_columns = X.select_dtypes(include=object).columns

numeric_inputer = SimpleImputer(strategy='mean')
X[numeric_columns] = numeric_inputer.fit_transform(X[numeric_columns])

categorical_inputer = SimpleImputer(strategy='most_frequent')
X[categorical_columns] = categorical_inputer.fit_transform(X[categorical_columns])

X = pd.get_dummies(X, drop_first=True)

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptim
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ...
env_123 (Python 3.8.18)

numeric_inputer = SimpleImputer(strategy='mean')
X[numeric_columns] = numeric_inputer.fit_transform(X[numeric_columns])

categorical_inputer = SimpleImputer(strategy='most_frequent')
X[categorical_columns] = categorical_inputer.fit_transform(X[categorical_columns])

X = pd.get_dummies(X, drop_first=True)

categorical_features_indices = np.where(X.dtypes != np.float)[0]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

param_grid = {
    'model__depth': [6, 8, 10],
    'model__learning_rate': [0.01, 0.05, 0.1],
    'model__iterations': [500, 1000],
}

model = CatBoostClassifier(verbose=False, random_state=0, cat_features=categorical_features_indices)

pipeline = Pipeline([
    ('oversample', SMOTE(sampling_strategy=0.5)),
    ('undersample', RandomUnderSampler(sampling_strategy=0.7)),
    ('model', model)
])

grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_

best_model = grid_search.best_estimator_
best_model.fit(X_train, y_train)

y_pred = best_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptim
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells 🔄 Restart 📄 Variables 📄 Outline ...
env_123 (Python 3.8.18)

grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_

best_model = grid_search.best_estimator_
best_model.fit(X_train, y_train)

y_pred = best_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)

print(f"Best Hyperparameters: {best_params}")
print(f"Accuracy: {accuracy:.4f}, Recall: {recall:.4f}, ROC-AUC: {roc_auc:.4f}, Precision: {precision:.4f}")

[21] Python

/var/folders/_1/z5syf9qn6xv80qfwvzms1l40000gn/T/ipykernel_94262/3255929110.py:46: DeprecationWarning:

`np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the
numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

Best Hyperparameters: {'model__depth': 6, 'model__iterations': 500, 'model__learning_rate': 0.01}
Accuracy: 0.7960, Recall: 0.6847, ROC-AUC: 0.7611, Precision: 0.6112

target_distribution = data['Churn'].value_counts()
int(target_distribution)

[22] Python

Churn
No    5174
Yes   1869

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptim
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells 🔄 Restart 📄 Variables 📄 Outline ...
env_123 (Python 3.8.18)

target_distribution = data['Churn'].value_counts()
int(target_distribution)

[22] Python

Churn
No    5174
Yes   1869
Name: count, dtype: int64

import joblib

joblib.dump(best_model, 'catboost_model.joblib')

[62] Python

['catboost_model.joblib']

import joblib
from sklearn.metrics import accuracy_score, roc_auc_score, precision_score, recall_score

model = joblib.load('catboost_model.joblib')

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"ROC AUC: {roc_auc:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")

[63] Python

```



```
churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptım
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells ↺ Restart 📄 Variables 📄 Outline ... env_123 (Python 3.8.18)
Python
[65]
... Accuracy: 0.7922
ROC AUC: 0.7520
Precision: 0.6077
Recall: 0.6638

▼ import joblib
  from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score

  model = joblib.load('catboost_model.joblib')

  y_pred = model.predict(X_test)

  conf_matrix = confusion_matrix(y_test, y_pred)

  accuracy = accuracy_score(y_test, y_pred)
  precision = precision_score(y_test, y_pred)
  recall = recall_score(y_test, y_pred)
  f1 = f1_score(y_test, y_pred)

  print("Confusion Matrix:")
  print(conf_matrix)
  print("Accuracy:", accuracy)
  print("Precision:", precision)
  print("Recall:", recall)
  print("F1 Score:", f1)

[24]
... Confusion Matrix:
[[1293 246]
 [ 193 381]]

Accuracy: 0.792238523426408
Precision: 0.6076555823923444
Recall: 0.663763066280906
F1 Score: 0.6344712739383847
```

```
churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
overUndersampling_env123.ipynb > # Churn columnunu 1 ve 0 yaptım
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells ↺ Restart 📄 Variables 📄 Outline ... env_123 (Python 3.8.18)
Python
[24]
... Confusion Matrix:
[[1293 246]
 [ 193 381]]

Accuracy: 0.792238523426408
Precision: 0.6076555823923444
Recall: 0.663763066280906
F1 Score: 0.6344712739383847

data.head(28)

[46]
...
gender SeniorCitizen Partner tenure MultipleLines InternetService OnlineSecurity OnlineBackup DeviceProtection TechSupport ... PaymentMethod MonthlyCharges TotalCharges Churn Dependence
0 0 0 0 1 1.0 No DSL No Yes No No — Electronic check 29.85 29.85 0
1 1 0 0 34.0 No DSL Yes No Yes No — Mailed check 56.95 1889.5 0
2 1 0 0 2.0 No DSL Yes Yes No No — Mailed check 53.85 108.15 1
3 1 0 0 45.0 No DSL Yes No Yes Yes — Bank transfer (automatic) 42.30 1840.75 0
4 0 0 0 2.0 No Fiber optic No No No No — Electronic check 70.70 151.65 1
5 0 0 0 8.0 Yes Fiber optic No No Yes No — Electronic check 99.65 820.5 1
6 1 0 0 22.0 Yes Fiber optic No Yes No No — Credit card (automatic) 89.10 1949.4 0
7 0 0 0 10.0 No DSL Yes No No No — Mailed check 29.75 301.9 0
8 0 0 1 28.0 Yes Fiber optic No No Yes Yes — Electronic check 104.80 3046.05 1
9 1 0 0 62.0 No DSL Yes Yes No No — Bank transfer (automatic) 56.15 3487.95 0
10 1 0 1 13.0 No DSL Yes No No No — Mailed check 49.95 587.45 0
11 1 0 0 16.0 No No No internet service No internet service No internet service No internet service — Credit card (automatic) 18.95 326.8 0
12 1 0 1 58.0 Yes Fiber optic No No Yes No — Credit card (automatic) 100.35 5681.1 0
13 1 0 0 49.0 Yes Fiber optic No Yes Yes No — Bank transfer (automatic) 103.70 5681.1 0
14 1 0 0 25.0 No Fiber optic Yes No Yes Yes — Electronic check 105.50 2681.1 0
15 0 0 1 69.0 Yes Fiber optic Yes Yes Yes Yes — Credit card (automatic) 113.25 781.1 0
16 0 0 0 52.0 No No No internet service No internet service No internet service No internet service — Mailed check 20.65 1022.95 0
```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)

[1]
#pip install optuna
#pip install catboost
#pip install category-encoders

+ Code + Markdown Python

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.impute import KNNImputer

from sklearn.preprocessing import StandardScaler

import sklearn.metrics as metrics

import optuna

from xgboost import XGBClassifier
from catboost import CatBoostClassifier
import warnings
warnings.filterwarnings("ignore")

[2] Python

df_orig = pd.read_csv("/Users/laraturunc/Desktop/churn data/Telco-Customer-Churn-Dataset.csv")
df = df_orig.copy()
df.head()

[3] Python
customerID gender SeniorCitizen Partner Dependents tenure PhoneService MultipleLines InternetService OnlineSecurity ... DeviceProtection TechSupport StreamingTV StreamingMovies Contract
0 7590-VHVEG Female 0 Yes No 1 No No phone service DSL No ... No No No No Month to-month
1 5575-GNVDE Male 0 No No 34 Yes No DSL Yes ... Yes No No No One year
2 3068-QPYBK Male 0 No No 2 Yes No DSL Yes ... No No No No Month to-month
3 7795-CFOCW Male 0 No No 45 No No phone service DSL Yes ... Yes Yes No No One year
4 9237-HQITU Female 0 No No 2 Yes No Fiber optic No ... No No No No Month to-month
5 rows x 21 columns

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)

[3] Python
df_orig = pd.read_csv("/Users/laraturunc/Desktop/churn data/Telco-Customer-Churn-Dataset.csv")
df = df_orig.copy()
df.head()

...
customerID gender SeniorCitizen Partner Dependents tenure PhoneService MultipleLines InternetService OnlineSecurity ... DeviceProtection TechSupport StreamingTV StreamingMovies Contract
0 7590-VHVEG Female 0 Yes No 1 No No phone service DSL No ... No No No No Month to-month
1 5575-GNVDE Male 0 No No 34 Yes No DSL Yes ... Yes No No No One year
2 3068-QPYBK Male 0 No No 2 Yes No DSL Yes ... No No No No Month to-month
3 7795-CFOCW Male 0 No No 45 No No phone service DSL Yes ... Yes Yes No No One year
4 9237-HQITU Female 0 No No 2 Yes No Fiber optic No ... No No No No Month to-month
5 rows x 21 columns

[4] Python
for i in df.columns.tolist():
    print("%25, 1, '--', df.dtypes[i], '%25"
    print(df[i].value_counts())
    print("%78")

Output exceeds the size limit. Open the full output data in a text editor
===== customerID -- object =====
customerID
7590-VHVEG    0.000142
3791-LGOCY    0.000142
6008-NAIXX    0.000142
5956-YHHRX    0.000142
5365-LLFYV    0.000142
...
9796-MVYXX    0.000142
2637-FKFSY    0.000142
1552-AAGRXX   0.000142
4304-TSPVK    0.000142
3186-AJIEK    0.000142

```

churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

env\_123 (Python 3.8.18)

```

for i in df.columns.tolist():
    print('%s*25, 1, '-', df.dtypes[i], '%s*25')
    print(df[i].value_counts())
    print('%s*70')

```

Output exceeds the [size limit](#). Open the full output [data in a text editor](#)

```

===== customerID -- object =====
customerID
7590-WHVEG    0.000142
3791-LGQCY    0.000142
6008-NAIXK    0.000142
5956-YHHRX    0.000142
5365-LLFYV    0.000142
...
9796-MVYXX    0.000142
2637-FKFSY    0.000142
1552-AAGRXX   0.000142
4384-TSPVK    0.000142
3186-AJIEK    0.000142
Name: proportion, Length: 7043, dtype: float64
=====
===== gender -- object =====
gender
Male    0.504756
Female  0.495244
Name: proportion, dtype: float64
=====
===== SeniorCitizen -- int64 =====
SeniorCitizen
0    0.837853
1    0.162147
...
No    0.73463
Yes   0.26537
Name: proportion, dtype: float64
=====

```

churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

env\_123 (Python 3.8.18)

```

df.drop('customerID', axis=1, inplace=True)

df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

df.Churn = df.Churn.apply(lambda x: 0 if x=='No' else 1)

df.SeniorCitizen = df.SeniorCitizen.apply(lambda x: 'No' if x==0 else 'Yes')

```

```

cols = df.columns.tolist()

plt.figure(figsize=(15, 30))

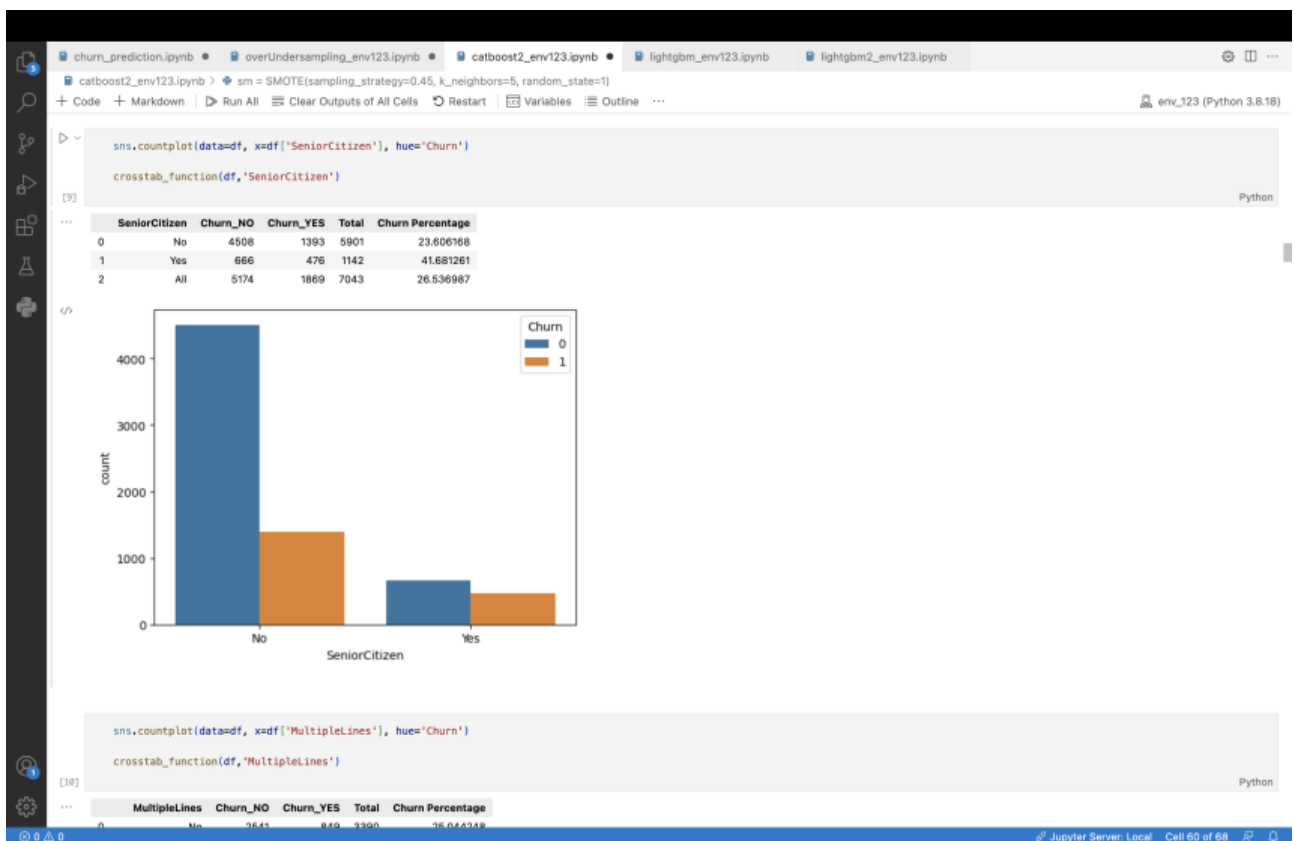
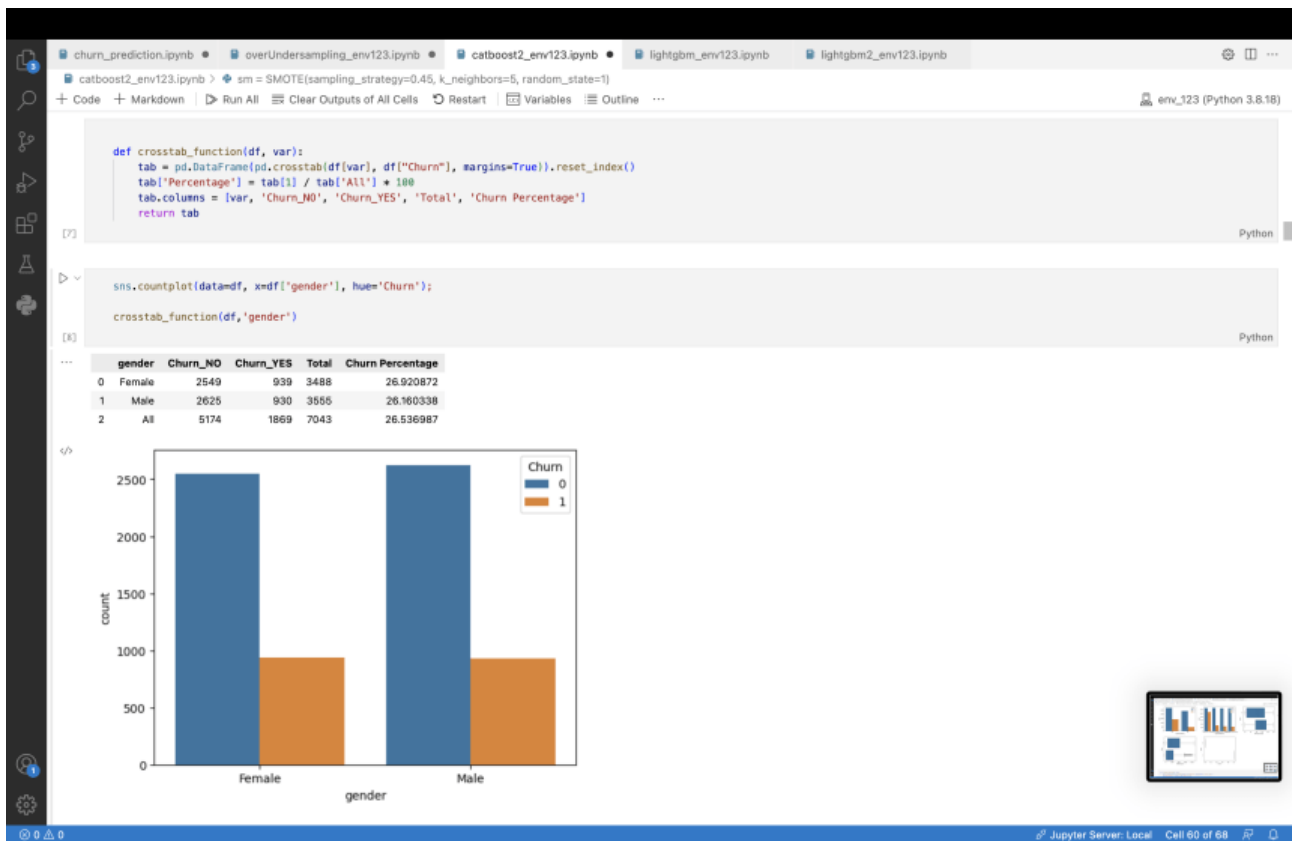
for i, variable in enumerate(df.columns.tolist()):
    plt.subplot(8, 3, i + 1)
    if df.dtypes[variable]=='object':
        sns.countplot(data=df, x=variable, hue='Churn')
    else:
        sns.boxplot(data=df, x=variable, y=df['Churn'].astype('i5'))
    plt.tight_layout(pad=2)

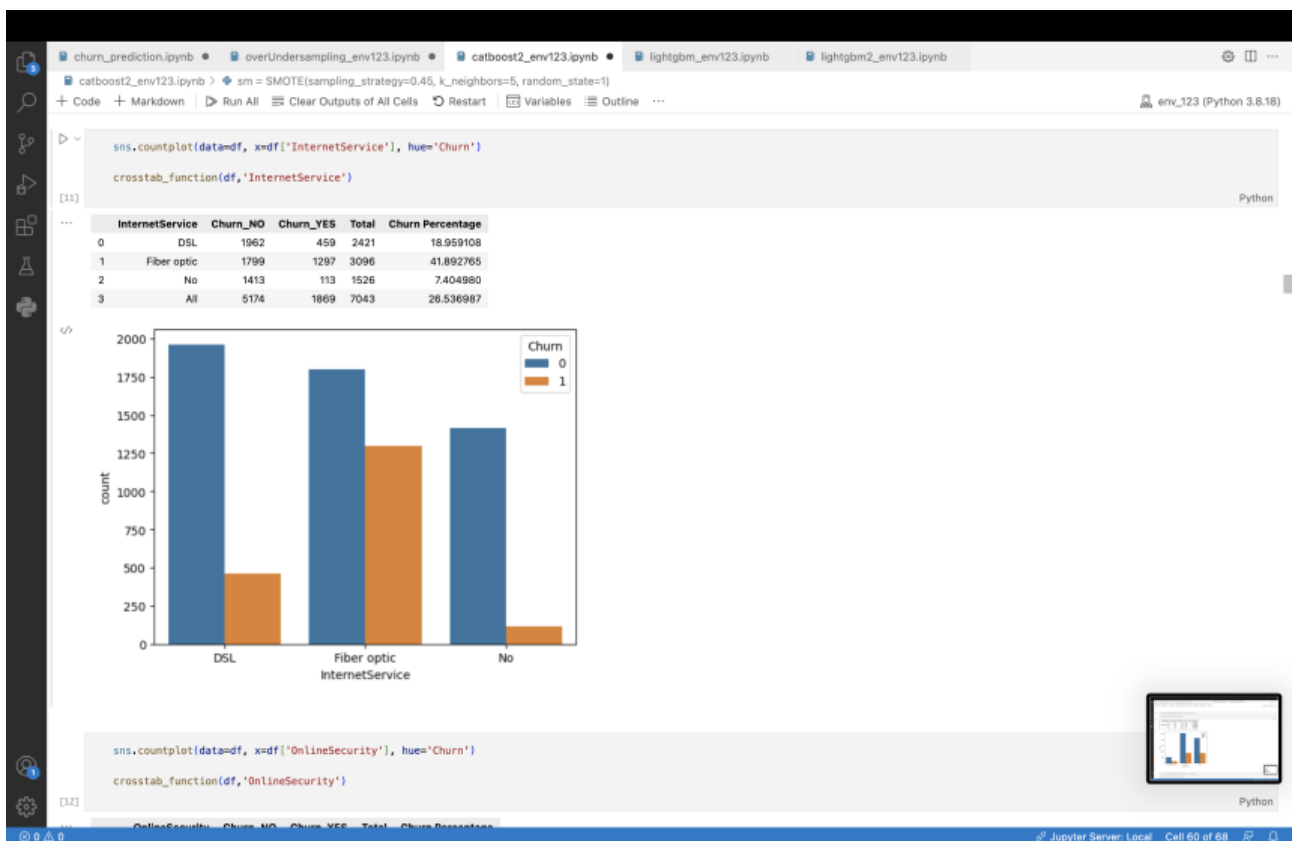
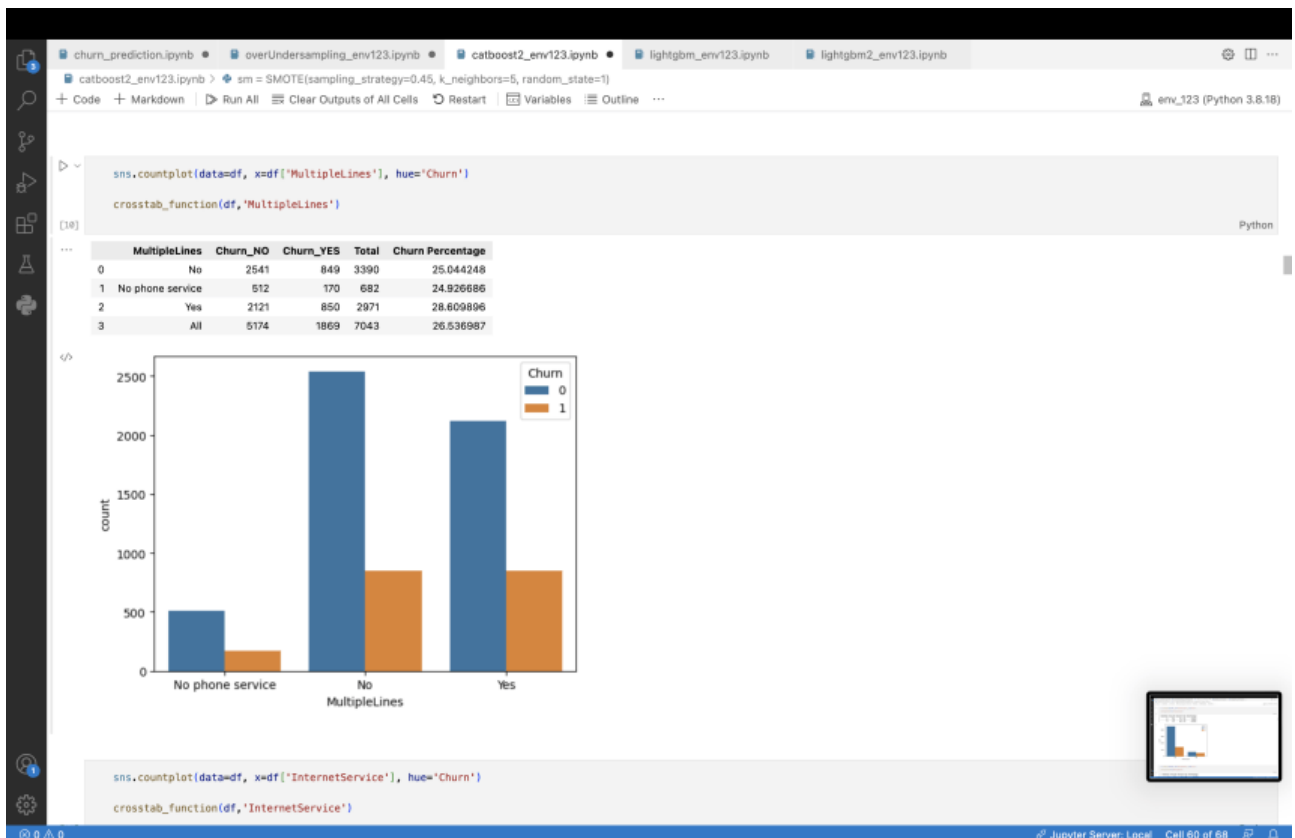
plt.show()

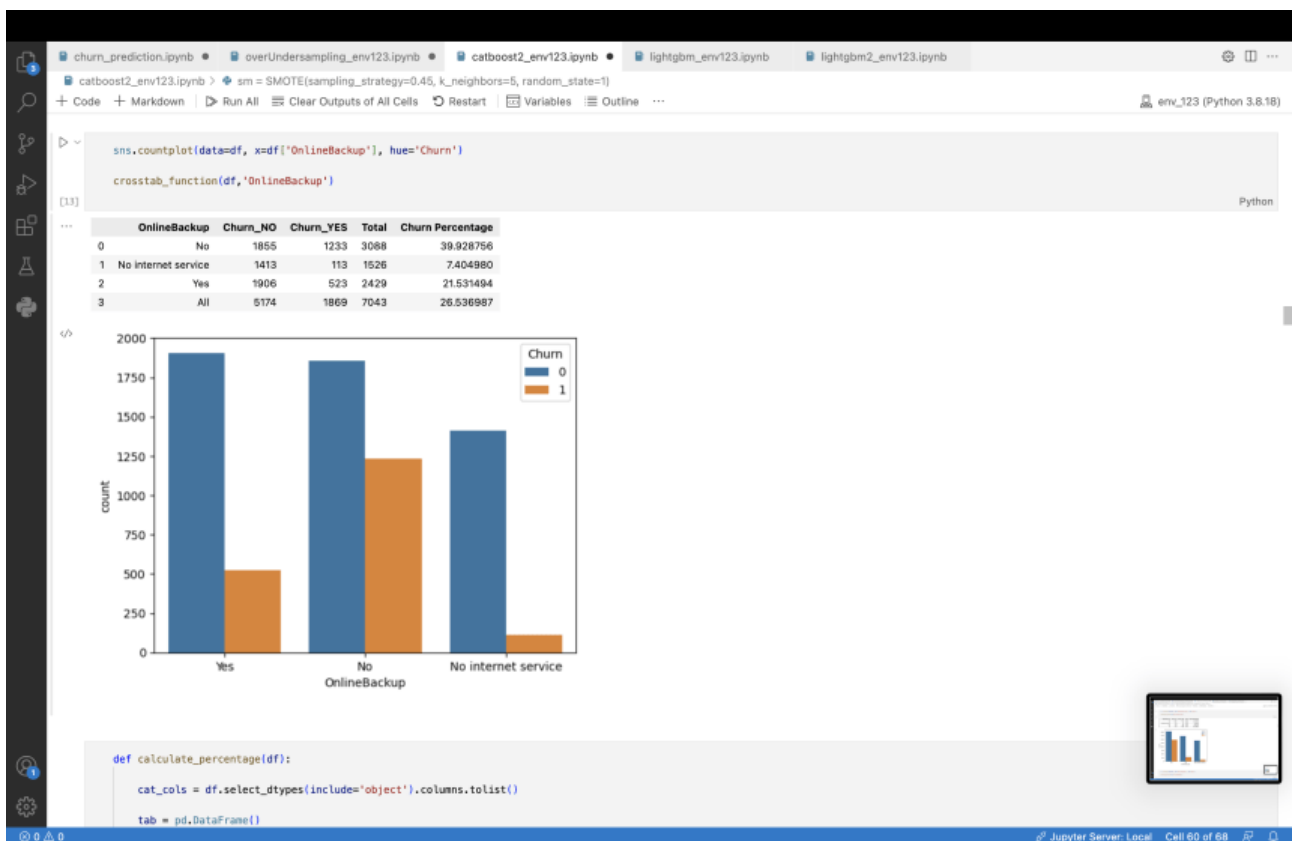
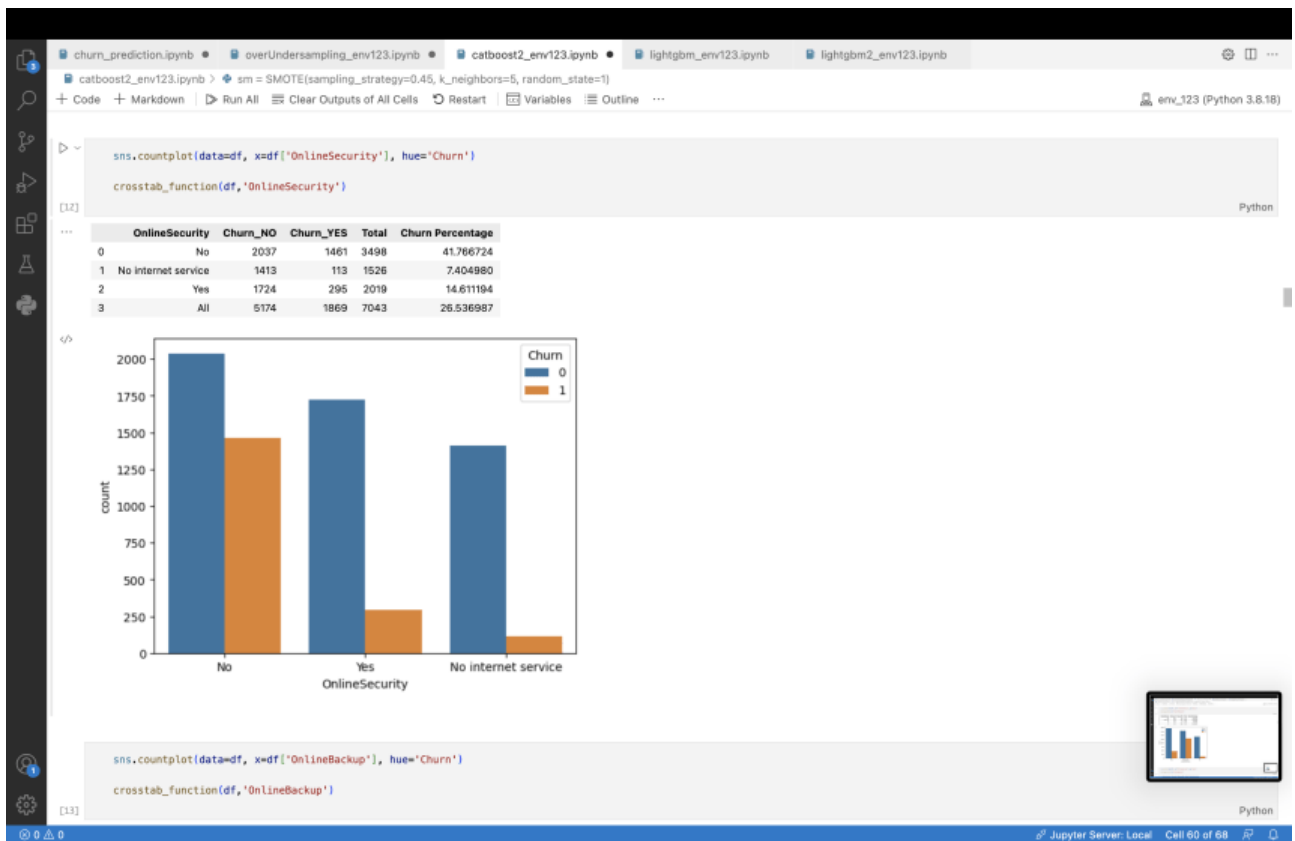
```

The figure displays three bar charts illustrating the distribution of Churn status (0: No, 1: Yes) across different demographic variables. The y-axis represents the count of observations.

- gender:** The chart shows the distribution of Churn status by gender. The y-axis ranges from 0 to 2500. For both Female and Male, the 'No' (Churn=0) group is significantly larger than the 'Yes' (Churn=1) group.
- SeniorCitizen:** The chart shows the distribution of Churn status by SeniorCitizen status. The y-axis ranges from 0 to 4000. For both 'No' and 'Yes' SeniorCitizen groups, the 'No' (Churn=0) group is significantly larger than the 'Yes' (Churn=1) group.
- Partner:** The chart shows the distribution of Churn status by Partner status. The y-axis ranges from 0 to 2500. For both 'Yes' and 'No' Partner groups, the 'No' (Churn=0) group is significantly larger than the 'Yes' (Churn=1) group.







churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

catboost2\_env123.ipynb > sm = SMOTE(sampling\_strategy=0.45, k\_neighbors=5, random\_state=1)

+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ...

env\_123 (Python 3.8.18)

```
def calculate_percentage(df):
    cat_cols = df.select_dtypes(include='object').columns.tolist()
    tab = pd.DataFrame()

    for i in cat_cols:
        tab1 = pd.DataFrame(pd.crosstab(df[i], df["Churn"], margins=True)).reset_index()
        tab1['Percentage'] = tab1[i] / tab1['All'] * 100

        tab1.reset_index(inplace=True)
        tab1.columns = ['Feature', 'Values', 'Churn_NO', 'Churn_YES', 'Total', 'Churn Percentage']
        tab1['Feature'] = i

        if tab.empty:
            tab = tab1
        else:
            tab = pd.merge(tab, tab1, how='outer')

    tab = tab[tab.Values != 'All'].sort_values(by='Churn Percentage', ascending=False).reset_index(drop=True)

    return tab
```

[14] Python

```
calculate_percentage(df).head()
```

[15] Python

	Feature	Values	Churn_NO	Churn_YES	Total	Churn Percentage
0	PaymentMethod	Electronic check	1294	1071	2365	45.285412
1	Contract	Month-to-month	2220	1655	3875	42.709677
2	InternetService	Fiber optic	1799	1297	3096	41.892765
3	OnlineSecurity	No	2037	1461	3498	41.766724
4	SeniorCitizen	Yes	666	476	1142	41.681261

```
df.Churn = df.Churn.apply(lambda x: 'No' if x==0 else 'Yes')
```

[16] Python

Jupyter Server: Local Cell 60 of 68

churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

catboost2\_env123.ipynb > sm = SMOTE(sampling\_strategy=0.45, k\_neighbors=5, random\_state=1)

+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ...

env\_123 (Python 3.8.18)

```
df.Churn = df.Churn.apply(lambda x: 'No' if x==0 else 'Yes')
```

[16] Python

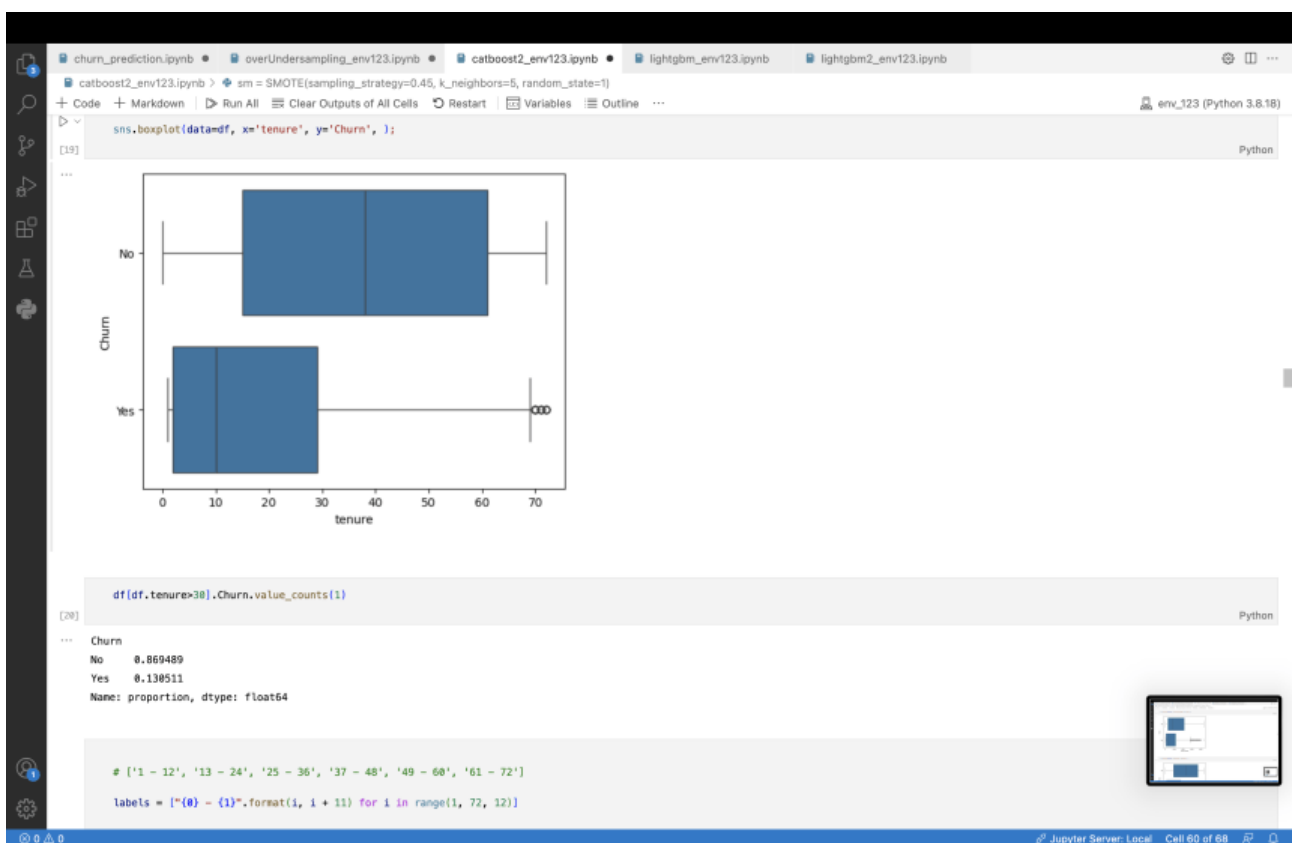
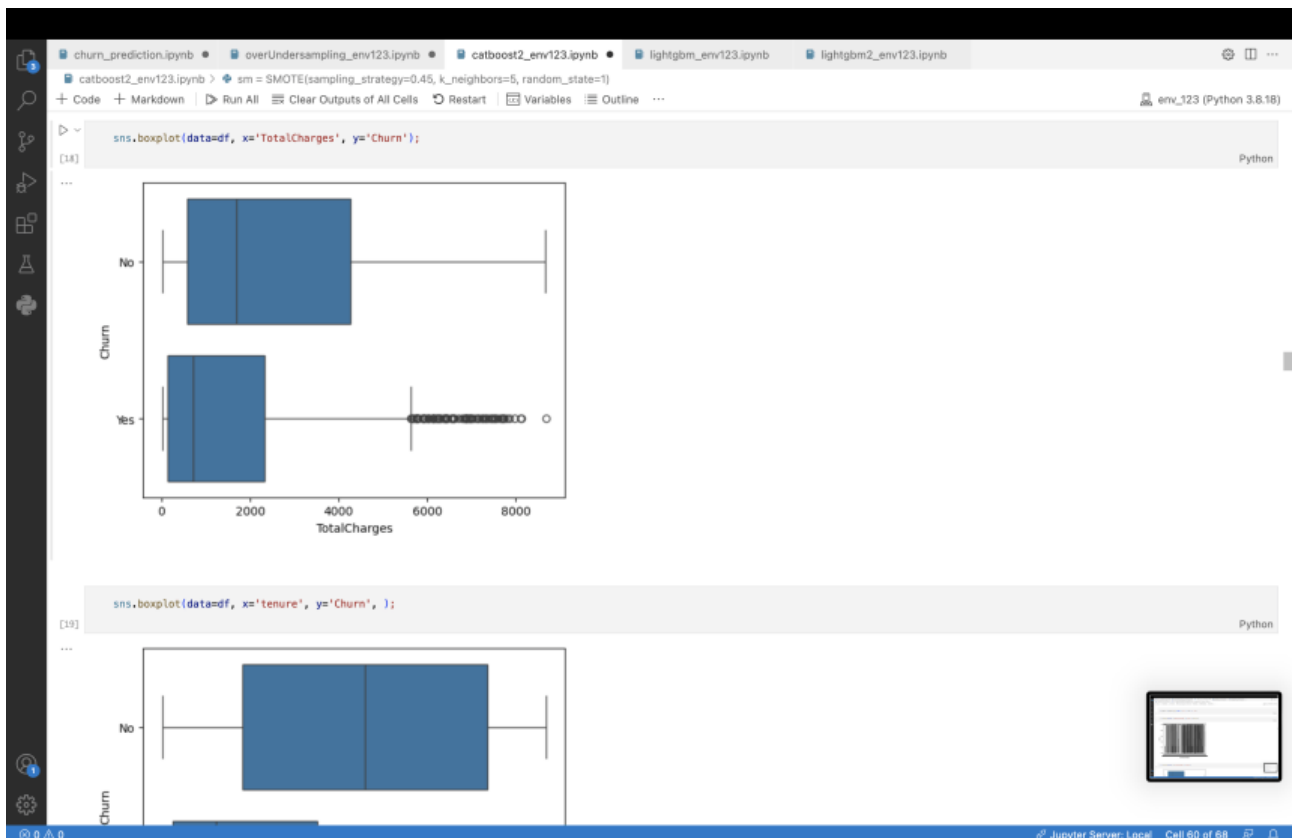
```
sns.boxplot(data=df, x='MonthlyCharges', y='Churn'); #?????????
```

[18] Python

```
sns.boxplot(data=df, x='TotalCharges', y='Churn');
```

[18] Python

Jupyter Server: Local Cell 60 of 68





churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

env\_123 (Python 3.8.18)

df[df.tenure>30].Churn.value\_counts()

```

Python
[20]
...
Churn
No    0.869489
Yes   0.130511
Name: proportion, dtype: float64

# ['1 - 12', '13 - 24', '25 - 36', '37 - 48', '49 - 60', '61 - 72']
labels = ["{0} - {1}".format(i, i + 11) for i in range(1, 72)]

df['tenure_group'] = pd.cut(df.tenure, range(1, 60, 12), right=False, labels=labels)
Python
[21]

df.tenure_group.unique()
Python
[22]
...
['1 - 12', '25 - 36', '37 - 48', '13 - 24', '61 - 72', '49 - 60', NaN]
Categories (6, object): ['1 - 12' < '13 - 24' < '25 - 36' < '37 - 48' < '49 - 60' < '61 - 72']

df[df.tenure_group.isna()==True]
Python
[23]
...

```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	TechSupport	StreamingTV	StreamingMovies	Contract	Paperless
488	Female	No	Yes	Yes	0	No	No phone service	DSL	Yes	No	Yes	Yes	No	Two year	
753	Male	No	No	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	
936	Female	No	Yes	Yes	0	Yes	No	DSL	Yes	Yes	No	Yes	Yes	Two year	
1082	Male	No	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	
1340	Female	No	Yes	Yes	0	No	No phone service	DSL	Yes	Yes	Yes	Yes	No	Two year	
3331	Male	No	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	

Jupyter Server: Local Cell 60 of 68

churn\_prediction.ipynb • overUndersampling\_env123.ipynb • catboost2\_env123.ipynb • lightgbm\_env123.ipynb • lightgbm2\_env123.ipynb

env\_123 (Python 3.8.18)

df[df.tenure==0]

```

Python
[24]
...

```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	TechSupport	StreamingTV	StreamingMovies	Contract	Paperless
488	Female	No	Yes	Yes	0	No	No phone service	DSL	Yes	No	Yes	Yes	No	Two year	
753	Male	No	No	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	
936	Female	No	Yes	Yes	0	Yes	No	DSL	Yes	Yes	No	Yes	Yes	Two year	
1082	Male	No	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	
1340	Female	No	Yes	Yes	0	No	No phone service	DSL	Yes	Yes	Yes	Yes	No	Two year	
3331	Male	No	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	
3826	Male	No	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	
4380	Female	No	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	Two year	
5218	Male	No	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service	One year	
6670	Female	No	Yes	Yes	0	Yes	Yes	DSL	No	Yes	Yes	Yes	No	Two year	
6754	Male	No	No	Yes	0	Yes	Yes	DSL	Yes	Yes	Yes	No	No	Two year	

11 rows x 16 columns

```

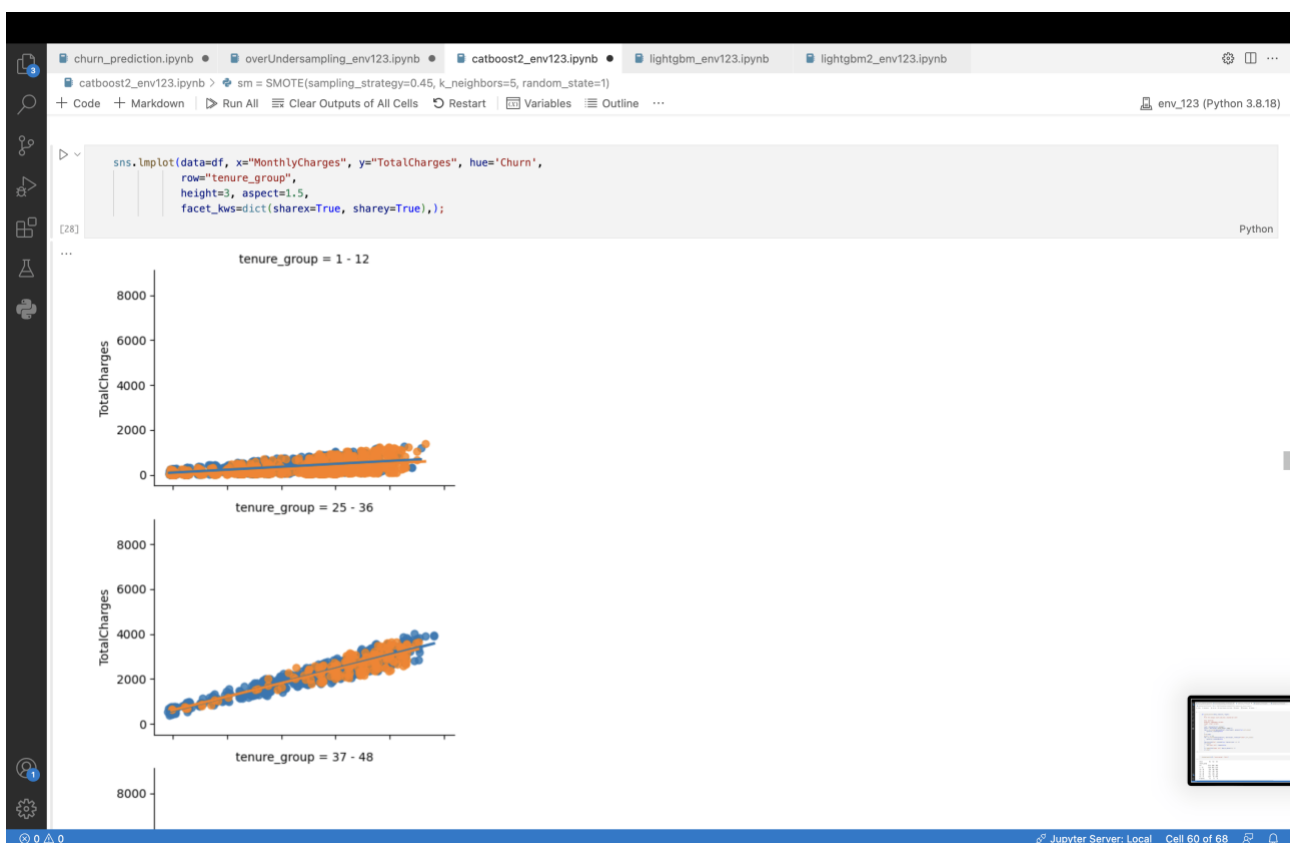
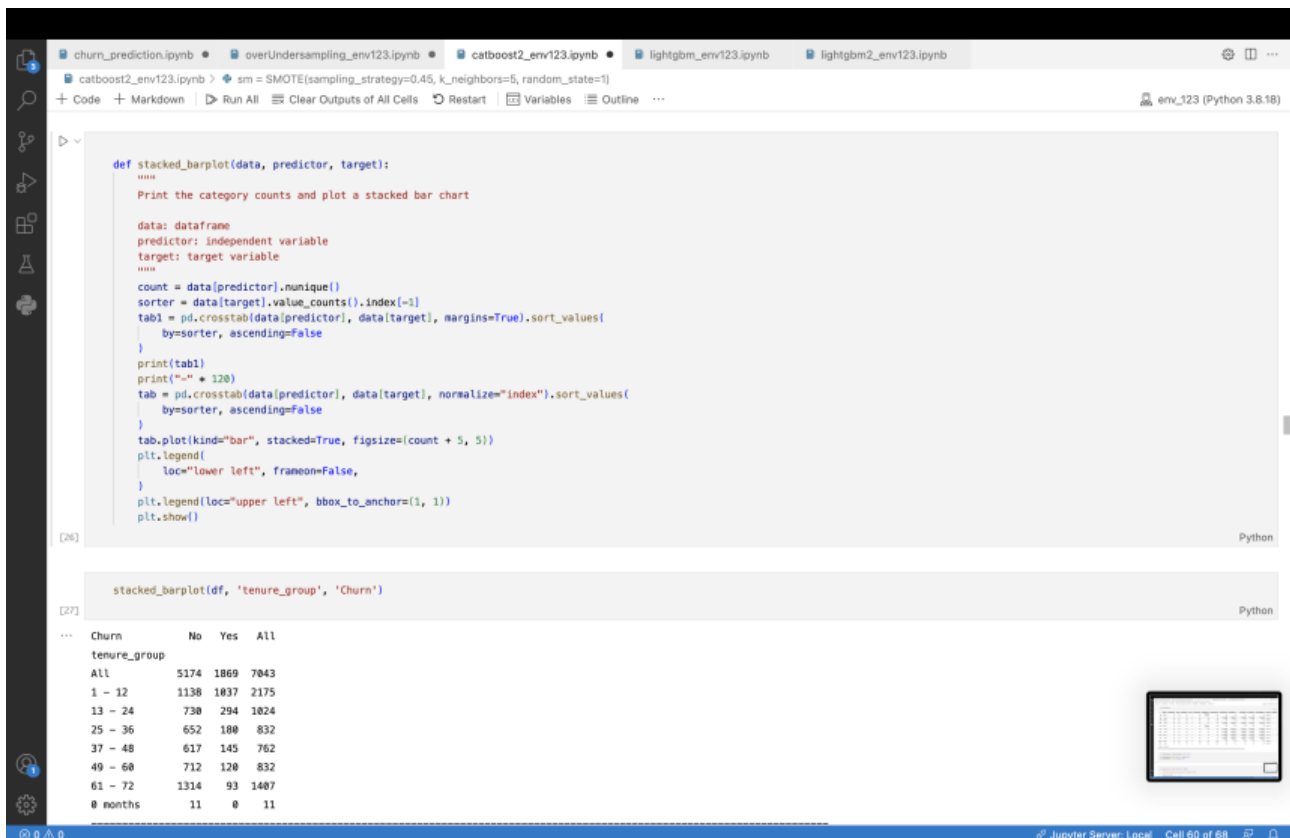
df.tenure_group = df.tenure_group.astype('object')
df.tenure_group.fillna("0 months", inplace=True)
df.TotalCharges.fillna(0, inplace=True)
Python
[25]

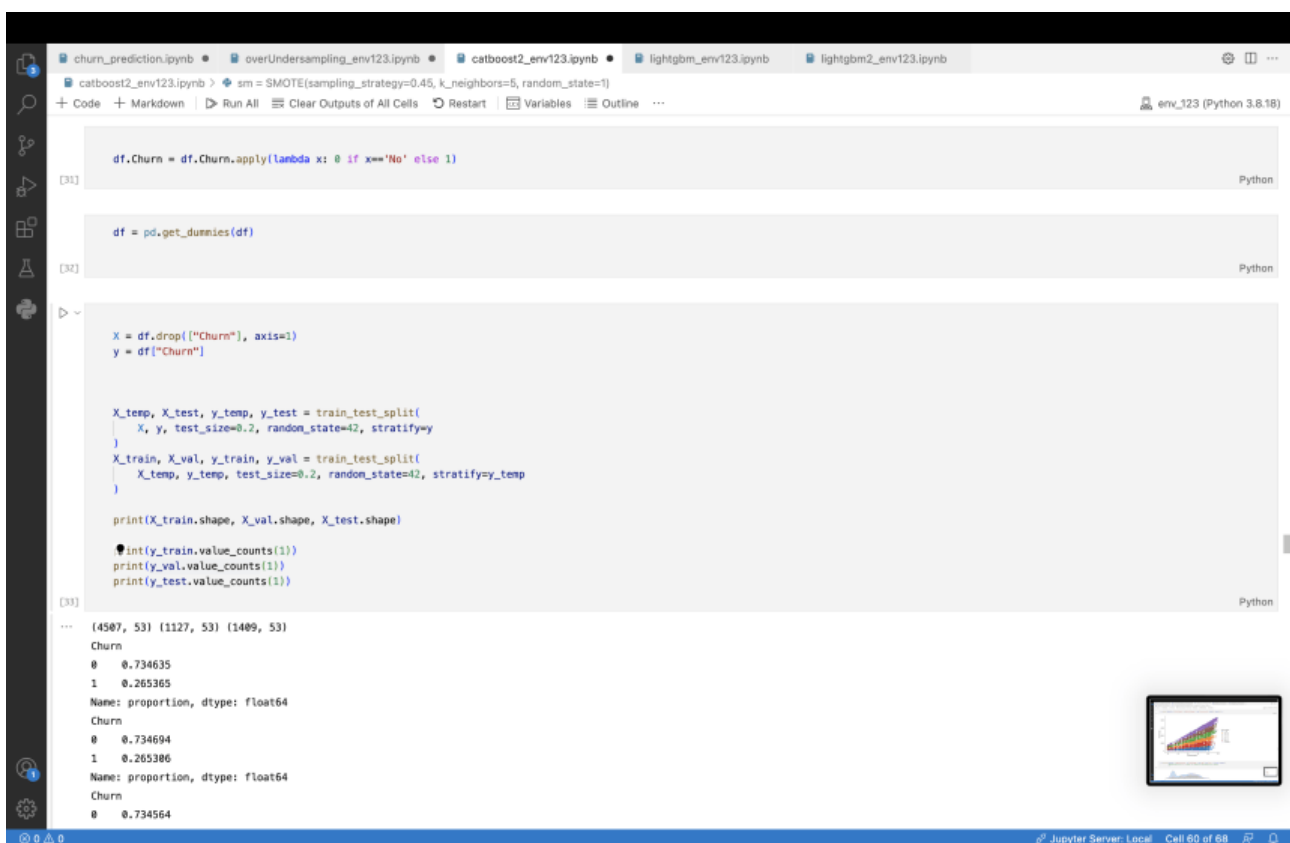
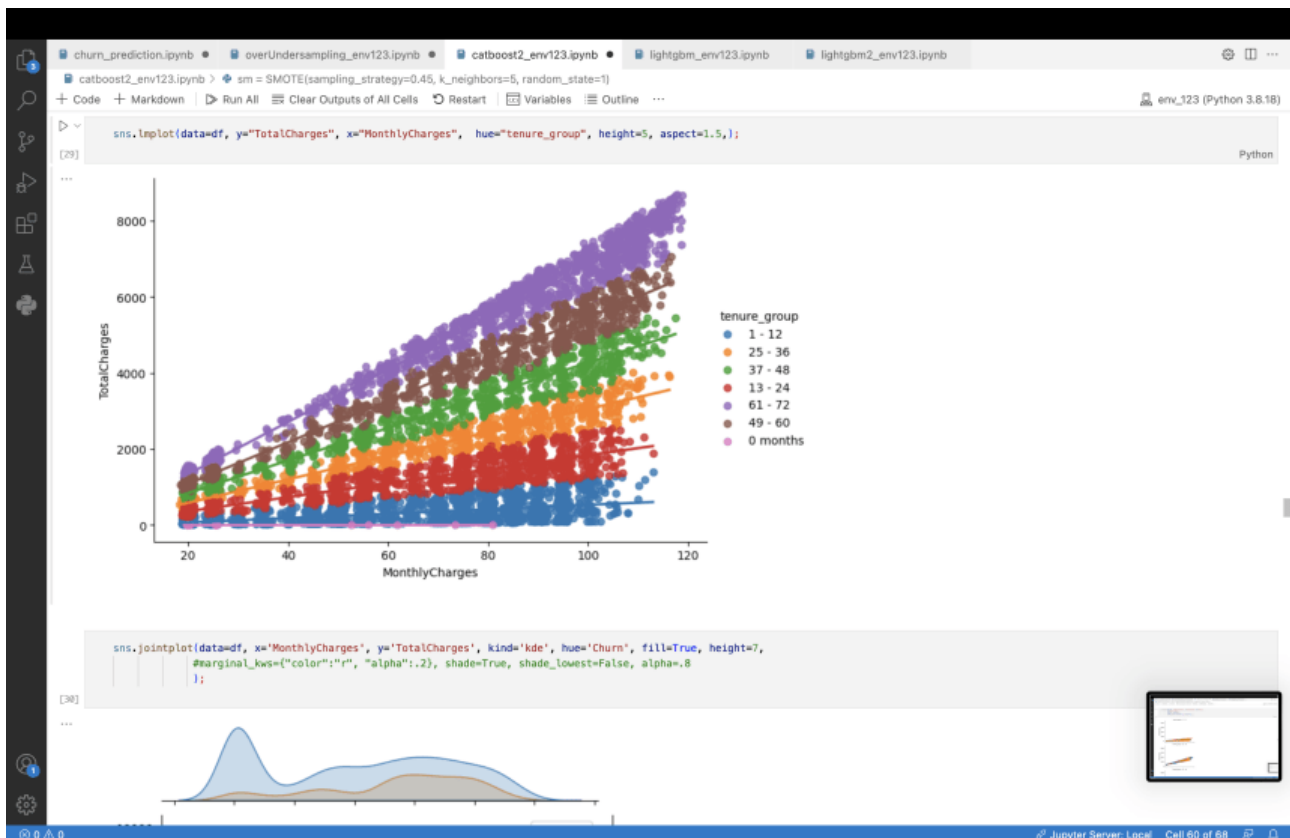
def stacked_barplot(data, predictor, target):
    """
    Print the category counts and plot a stacked bar chart

    data: dataframe
    predictor: independent variable
    """

```

Jupyter Server: Local Cell 60 of 68





```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
+ Code + Markdown ▶ Run All Clear Outputs of All Cells Restart Variables Outline ... env_123 (Python 3.8.18)

... (4507, 53) (1127, 53) (1409, 53)
Churn
0 0.734635
1 0.265365
Name: proportion, dtype: float64
Churn
0 0.734694
1 0.265386
Name: proportion, dtype: float64
Churn
0 0.734564
1 0.265436
Name: proportion, dtype: float64

[34]
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

def model_performance_classification_sklearn(model, predictors, target):
    """
    Function to compute different metrics to check classification model performance

    model: classifier
    predictors: independent variables
    target: dependent variable
    """

    pred = model.predict(predictors)

    acc = metrics.accuracy_score(target, pred)
    recall = metrics.recall_score(target, pred)
    precision = metrics.precision_score(target, pred)
    f1 = metrics.f1_score(target, pred)
    mcc = metrics.matthews_corrcoef(target, pred)
    auc = metrics.roc_auc_score(target, pred)

    df_perf = pd.DataFrame(
        {"Accuracy": acc, "Recall": recall, "Precision": precision, "F1": f1, "Matthews Coef": mcc, "ROC_AUC": auc},
        index=[0],
    )

    return df_perf

xgbc_base = XGBClassifier(random_state=0)

xgbc_base.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_val, y_val)], eval_metric="auc", verbose=False)

model_performance_classification_sklearn(xgbc_base, X_train, y_train)

...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.95629 0.893813 0.938543 0.915632 0.886641 0.936336

model_performance_classification_sklearn(xgbc_base, X_val, y_val)

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
+ Code + Markdown ▶ Run All Clear Outputs of All Cells Restart Variables Outline ... env_123 (Python 3.8.18)

acc = metrics.accuracy_score(target, pred)
recall = metrics.recall_score(target, pred)
precision = metrics.precision_score(target, pred)
f1 = metrics.f1_score(target, pred)
mcc = metrics.matthews_corrcoef(target, pred)
auc = metrics.roc_auc_score(target, pred)

df_perf = pd.DataFrame(
    {"Accuracy": acc, "Recall": recall, "Precision": precision, "F1": f1, "Matthews Coef": mcc, "ROC_AUC": auc},
    index=[0],
)

return df_perf

[35]

xgbc_base = XGBClassifier(random_state=0)

xgbc_base.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_val, y_val)], eval_metric="auc", verbose=False)

[36]

XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=None, max_leaves=None,
               min_child_weight=None, missing=nan, monotone_constraints=None,
               n_estimators=100, n_jobs=None, num_parallel_tree=None,
               predictor=None, random_state=0, ...)

model_performance_classification_sklearn(xgbc_base, X_train, y_train)

[37]

...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.95629 0.893813 0.938543 0.915632 0.886641 0.936336

model_performance_classification_sklearn(xgbc_base, X_val, y_val)

[38]

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)

[37]
model_performance_classification_sklearn(xgbc_base,X_train,y_train)
Python
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.95629 0.893813 0.938543 0.915632 0.888641 0.936336

[38]
model_performance_classification_sklearn(xgbc_base,X_val,y_val)
Python
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.782609 0.494983 0.61157 0.547135 0.410125 0.690728

[39]
model_performance_classification_sklearn(xgbc_base,X_test,y_test)
Python
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.787083 0.516043 0.61859 0.562682 0.426523 0.700533
+ Code + Markdown

[40]
#HYPEROPT
# import packages for hyperparameters tuning
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
Python

space={
    'booster': hp.choice('booster',['gbtree', 'dart']),
    'max_depth': hp.quniform('max_depth', 2, 22, 1),
    '#gamma': hp.uniform('gamma', 0, 1),
    'reg_alpha': hp.quniform('reg_alpha', 1, 50, 1),
    'reg_lambda': hp.uniform('reg_lambda', 0, 1),
    'colsample_bytree': hp.uniform('colsample_bytree', 0.6, 1),
    '#min_child_weight': hp.quniform('min_child_weight', 1, 40, 1),
    'n_estimators': hp.quniform('n_estimators', 1000, 3500, 100),
    '#scale_pos_weight': hp.quniform('scale_pos_weight', 1, 4, 1),
    'learning_rate': hp.uniform('learning_rate', 0.01, 0.1),
    '#subsample': hp.uniform('subsample', 0.5, 0.9),
    'colsample_bylevel': hp.uniform('colsample_bylevel', 0.6, 1),
    '#max_delta_step': hp.quniform('max_delta_step', 1, 10, 1),
    'seed': 0
}

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)

[41]
space={
    'booster': hp.choice('booster',['gbtree', 'dart']),
    'max_depth': hp.quniform('max_depth', 2, 22, 1),
    '#gamma': hp.uniform('gamma', 0, 1),
    'reg_alpha': hp.quniform('reg_alpha', 1, 50, 1),
    'reg_lambda': hp.uniform('reg_lambda', 0, 1),
    'colsample_bytree': hp.uniform('colsample_bytree', 0.6, 1),
    '#min_child_weight': hp.quniform('min_child_weight', 1, 40, 1),
    'n_estimators': hp.quniform('n_estimators', 1000, 3500, 100),
    '#scale_pos_weight': hp.quniform('scale_pos_weight', 1, 4, 1),
    'learning_rate': hp.uniform('learning_rate', 0.01, 0.1),
    '#subsample': hp.uniform('subsample', 0.5, 0.9),
    'colsample_bylevel': hp.uniform('colsample_bylevel', 0.6, 1),
    '#max_delta_step': hp.quniform('max_delta_step', 1, 10, 1),
    '#eta': hp.uniform('eta', 0, 1),
    'seed': 0
}
Python

[42]
def objective(space):
    clf=XGBClassifier(
        #objective = 'reg:logistic',
        booster = str(space['booster']),
        max_depth = int(space['max_depth']),
        #gamma = space['gamma'],
        reg_alpha = int(space['reg_alpha']),
        reg_lambda = space['reg_lambda'],
        colsample_bytree = space['colsample_bytree'],
        #min_child_weight = int(space['min_child_weight']),
        n_estimators = int(space['n_estimators']),
        #scale_pos_weight = int(space['scale_pos_weight']),
        learning_rate = space['learning_rate'],
        #subsample = space['subsample'],
        colsample_bylevel = space['colsample_bylevel'],
        #max_delta_step = int(space['max_delta_step']),
        #eta = space['eta'],
        seed = 0
    )

    evaluation = [(X_train, y_train), (X_val, y_val)]

    clf.fit(X_train, y_train,
            eval_set=evaluation, eval_metric='auc',
            early_stopping_rounds=5, verbose=False)

```



```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
+ Code + Markdown ▶ Run All Clear Outputs of All Cells Restart Variables Outline ... env_123 (Python 3.8.18)

XGBModel_OrigData_tuned = XGBClassifier(**param_grid, random_state=0)
XGBModel_OrigData_tuned.fit(X_train, y_train, eval_set=((X_val, y_val)), eval_metric="auc", early_stopping_rounds=10, verbose=False)

[45]
...
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=0.6582421895059724, colsample_bynode=None,
               colsample_bytree=0.6753458489151604, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=0.05730502030111278,
               max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=5, max_leaves=None,
               min_child_weight=None, missing=nan, monotone_constraints=None,
               n_estimators=1100, n_jobs=None, num_parallel_tree=None,
               predictor=None, random_state=0, ...)

model_performance_classification_sklearn(XGBModel_OrigData_tuned,X_train,y_train)

[46]
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.811626 0.520067 0.693423 0.594362 0.483267 0.718505

model_performance_classification_sklearn(XGBModel_OrigData_tuned,X_val,y_val)

[47]
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.815439 0.498328 0.719807 0.588933 0.488314 0.71414

model_performance_classification_sklearn(XGBModel_OrigData_tuned,X_test,y_test)

[48]
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.799858 0.486631 0.669118 0.563467 0.447144 0.699837

catboost_base = CatBoostClassifier( random_state = 1, verbose=False,)# scale_pos_weight=2,)# eval_metric = "Accuracy", )
catboost_base.fit(X_train, y_train, eval_set=(X_val, y_val))# cat_features = cat_features,)# plot=True.

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
+ Code + Markdown ▶ Run All Clear Outputs of All Cells Restart Variables Outline ... env_123 (Python 3.8.18)

catboost_base.fit(X_train, y_train, eval_set=(X_val, y_val))# cat_features = cat_features,)# plot=True,

[49]
...
<catboost.core.CatBoostClassifier at 0x1784e4280>

model_performance_classification_sklearn(catboost_base,X_train,y_train)

[50]
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.832039 0.572742 0.735768 0.6441 0.543611 0.749222

model_performance_classification_sklearn(catboost_base,X_val,y_val)

[51]
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.807453 0.494983 0.691589 0.576998 0.467458 0.707637

model_performance_classification_sklearn(catboost_base,X_test,y_test)

[52]
...
Accuracy Recall Precision F1 Matthews Coef ROC_AUC
0 0.803407 0.513369 0.66899 0.580938 0.462222 0.710791

import optuna
def optimize_hp(trial):
    cb_params = {
        #'iterations': 1000,
        'iterations': trial.suggest_int('iterations', 1000, 4000),
        'objective': trial.suggest_categorical('objective', ["Logloss", "CrossEntropy"]),
        #'objective': "Logloss",
        #'colsample_bylevel': trial.suggest_float('colsample_bylevel', 0.01, 0.1),
        'boosting_type': trial.suggest_categorical('boosting_type', ["Ordered", "Plain"]),
        'learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 1.0),
        'l2_leaf_reg': trial.suggest_loguniform('l2_leaf_reg', 1, 100),
        'bagging_temperature': trial.suggest_loguniform('bagging_temperature', 0.1, 20.0),
        'random_strength': trial.suggest_float('random_strength', 1.0, 2.0),
        'depth': trial.suggest_int('depth', 1, 10),
    }

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)
import optuna

def optimize_hp(trial):
    cb_params = {
        #'iterations': 1000,
        'iterations': trial.suggest_int('iterations', 1000, 4000),
        'objective': trial.suggest_categorical('objective', ['Logloss', 'CrossEntropy']),
        #'objective': 'Logloss',
        #'colsample_bylevel': trial.suggest_float('colsample_bylevel', 0.01, 0.1),
        'boosting_type': trial.suggest_categorical('boosting_type', ['Ordered', 'Plain']),
        'learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 1.0),
        'l2_leaf_reg': trial.suggest_loguniform('l2_leaf_reg', 1, 100),
        'bagging_temperature': trial.suggest_loguniform('bagging_temperature', 0.1, 20.0),
        'random_strength': trial.suggest_float('random_strength', 1.0, 2.0),
        'depth': trial.suggest_int('depth', 1, 10),
        'min_data_in_leaf': trial.suggest_int('min_data_in_leaf', 1, 300),
        'use_best_model': True,
        #'task_type': 'GPU',
        #'scale_pos_weight': trial.suggest_float('scale_pos_weight', 1.0, 7.0),
        'random_seed': 42
    }

    #if cb_params['objective'] == "Logloss":
    #    cb_params['scale_pos_weight'] = trial.suggest_float("scale_pos_weight", 1, 4)

    model = CatBoostClassifier(**cb_params)
    model.fit(X_train, y_train, eval_set=(X_val, y_val), verbose=False, )#cat_features = cat_features, )
    y_pred = model.predict(X_val)
    #return metrics.accuracy_score(y_val, y_pred)
    return metrics.recall_score(y_val, y_pred)

study = optuna.create_study(direction="maximize")
study.optimize(optimize_hp, n_trials=20)
print('Trials:', len(study.trials))
print('Best parameters:', study.best_trial.params)
print('Best score:', study.best_value)

[I 2023-11-20 13:39:39,099] A new study created in memory with name: no-name-e27cf6e-1d48-41f3-86d2-a68d2dd3123b
[I 2023-11-20 13:48:04,673] Trial 0 finished with value: 0.4916387959866221 and parameters: {'iterations': 2733, 'objective': 'Logloss', 'boosting_type': 'Ordered', 'learning_rate': 0.10837347498481475, 'l2_leaf_reg': 3.76296763702925, 'bagging_temperature': 7.5612993397923365, 'random_strength': 1.6771452830958555, 'depth': 7, 'min_data_in_leaf': 275}. Best is trial 0 with value: 0.4916387959866221.

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)

Trials: 20
Best parameters: {'iterations': 2812, 'objective': 'CrossEntropy', 'boosting_type': 'Ordered', 'learning_rate': 0.08387481648633008, 'l2_leaf_reg': 73.97863488387144, 'bagging_temperature': 0.722224452929659, 'random_strength': 1.2322444756426854, 'depth': 10, 'min_data_in_leaf': 128}
Best score: 0.5384615384615384

params = study.best_trial.params

catboost_tuned = CatBoostClassifier(verbose=False, random_state=1, **params)
catboost_tuned.fit(X_train, y_train, eval_set=(X_val, y_val),)# cat_features = cat_features)

<catboost.core.CatBoostClassifier at 0x178618dc0>

model_performance_classification_sklearn(catboost_tuned,X_train, y_train)

Accuracy  Recall  Precision  F1  Matthews Coef  ROC_AUC
0  0.825826  0.565217  0.718385  0.632663  0.527064  0.742591

model_performance_classification_sklearn(catboost_tuned,X_val, y_val)

Accuracy  Recall  Precision  F1  Matthews Coef  ROC_AUC
0  0.806566  0.505017  0.683258  0.580769  0.467954  0.710238

model_performance_classification_sklearn(catboost_tuned,X_test, y_test)

Accuracy  Recall  Precision  F1  Matthews Coef  ROC_AUC
0  0.802697  0.510695  0.667832  0.578788  0.459887  0.709454

#undersampling ve oversampling
from imblearn.over_sampling import SMOTE

```



```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
X_train_hyb, y_train_hyb = sm.fit_resample(X_train, y_train)

rus = RandomUnderSampler(random_state=1, sampling_strategy=1)
X_train_hyb, y_train_hyb = rus.fit_resample(X_train_hyb, y_train_hyb)

int(y_train_hyb.value_counts(1))
print(y_val.value_counts(1))
print(y_test.value_counts(1))

Churn
0 0.5
1 0.5
Name: proportion, dtype: float64
Churn
0 0.734694
1 0.265306
Name: proportion, dtype: float64
Churn
0 0.734564
1 0.265436
Name: proportion, dtype: float64

import optuna

def optimize_hp(trial):
    cb_params = {
        'iterations': 1000,
        'iterations': trial.suggest_int('iterations', 1000, 4000),
        'objective': trial.suggest_categorical('objective', ["Logloss", "CrossEntropy"]),
        'objective': "Logloss",
        'colsample_bylevel': trial.suggest_float('colsample_bylevel', 0.01, 0.1),
        'boosting_type': trial.suggest_categorical('boosting_type', ["Ordered", "Plain"]),
        'learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 1.0),
        'l2_leaf_reg': trial.suggest_loguniform('l2_leaf_reg', 1, 100),
        'bagging_temperature': trial.suggest_loguniform('bagging_temperature', 0.1, 20.0),
        'random_strength': trial.suggest_float('random_strength', 1.0, 2.0)
    }

    #if cb_params["objective"] == "Logloss":
    #    cb_params["scale_pos_weight"] = trial.suggest_float("scale_pos_weight", 1, 4)

    model = CatBoostClassifier(**cb_params)
    model.fit(X_train_hyb, y_train_hyb, eval_set=(X_val, y_val), verbose=False, )
    y_pred = model.predict(X_val)
    #return metrics.accuracy_score(y_val, y_pred)
    return metrics.recall_score(y_val, y_pred)

study = optuna.create_study(direction="maximize")
study.optimize(optimize_hp, n_trials=20)
print('Trials:', len(study.trials))
print('Best parameters:', study.best_trial.params)
print('Best score:', study.best_value)

[I 2023-11-20 14:15:30.276] A new study created in memory with name: no-name-4405d9a6-35ec-4922-97c8-b566f7b22bf1
[I 2023-11-20 14:15:33.937] Trial 0 finished with value: 0.7692387692387693 and parameters: {'iterations': 3716, 'objective': 'Logloss', 'boosting_type': 'Plain', 'learning_rate': 0.5356386347736253, 'l2_leaf_reg': 3.96338409278045, 'bagging_temperature': 0.25237751865081915, 'random_strength': 1.9621322694781673, 'depth': 3, 'min_data_in_leaf': 284}. Best is trial 0 with value: 0.7692387692387693.

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
catboost2_env123.ipynb > sm = SMOTE(sampling_strategy=0.45, k_neighbors=5, random_state=1)
X_train_hyb, y_train_hyb = sm.fit_resample(X_train, y_train)

rus = RandomUnderSampler(random_state=1, sampling_strategy=1)
X_train_hyb, y_train_hyb = rus.fit_resample(X_train_hyb, y_train_hyb)

int(y_train_hyb.value_counts(1))
print(y_val.value_counts(1))
print(y_test.value_counts(1))

Churn
0 0.5
1 0.5
Name: proportion, dtype: float64
Churn
0 0.734694
1 0.265306
Name: proportion, dtype: float64
Churn
0 0.734564
1 0.265436
Name: proportion, dtype: float64

import optuna

def optimize_hp(trial):
    cb_params = {
        'iterations': 1000,
        'iterations': trial.suggest_int('iterations', 1000, 4000),
        'objective': trial.suggest_categorical('objective', ["Logloss", "CrossEntropy"]),
        'objective': "Logloss",
        'colsample_bylevel': trial.suggest_float('colsample_bylevel', 0.01, 0.1),
        'boosting_type': trial.suggest_categorical('boosting_type', ["Ordered", "Plain"]),
        'learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 1.0),
        'l2_leaf_reg': trial.suggest_loguniform('l2_leaf_reg', 1, 100),
        'bagging_temperature': trial.suggest_loguniform('bagging_temperature', 0.1, 20.0),
        'random_strength': trial.suggest_float('random_strength', 1.0, 2.0),
        'depth': trial.suggest_int('depth', 1, 10),
        'min_data_in_leaf': trial.suggest_int('min_data_in_leaf', 1, 300),
        'use_best_model': True,
        'task_type': "GPU",
        'scale_pos_weight': trial.suggest_float('scale_pos_weight', 1.0, 7.0),
        'random_seed': 1
    }

    #if cb_params["objective"] == "Logloss":
    #    cb_params["scale_pos_weight"] = trial.suggest_float("scale_pos_weight", 1, 4)

    model = CatBoostClassifier(**cb_params)
    model.fit(X_train_hyb, y_train_hyb, eval_set=(X_val, y_val), verbose=False, )
    y_pred = model.predict(X_val)
    #return metrics.accuracy_score(y_val, y_pred)
    return metrics.recall_score(y_val, y_pred)

study = optuna.create_study(direction="maximize")
study.optimize(optimize_hp, n_trials=20)
print('Trials:', len(study.trials))
print('Best parameters:', study.best_trial.params)
print('Best score:', study.best_value)

[I 2023-11-20 14:15:30.276] A new study created in memory with name: no-name-4405d9a6-35ec-4922-97c8-b566f7b22bf1
[I 2023-11-20 14:15:33.937] Trial 0 finished with value: 0.7692387692387693 and parameters: {'iterations': 3716, 'objective': 'Logloss', 'boosting_type': 'Plain', 'learning_rate': 0.5356386347736253, 'l2_leaf_reg': 3.96338409278045, 'bagging_temperature': 0.25237751865081915, 'random_strength': 1.9621322694781673, 'depth': 3, 'min_data_in_leaf': 284}. Best is trial 0 with value: 0.7692387692387693.

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)

6.7114829587519635, 'random_strength': 1.3751936631772765, 'depth': 3, 'min_data_in_leaf': 171)
Best score: 0.882675585284281

params = study.best_trial.params

catboost_tuned = CatBoostClassifier(verbose=False, random_state=1, **params)
catboost_tuned.fit(X_train_hyb, y_train_hyb, eval_set=(X_val, y_val), cat_features = cat_features)

<catboost.core.CatBoostClassifier at 0x16af835b0>

model_performance_classification_sklearn(catboost_tuned, X_train_hyb, y_train_hyb)

Accuracy  Recall  Precision  F1  Matthews Coef  ROC_AUC
0  0.799866  0.853593  0.77077  0.81007  0.603224  0.799866

model_performance_classification_sklearn(catboost_tuned, X_val, y_val)

Accuracy  Recall  Precision  F1  Matthews Coef  ROC_AUC
0  0.759539  0.802676  0.530973  0.639148  0.492412  0.773318

model_performance_classification_sklearn(catboost_tuned, X_test, y_test)

Accuracy  Recall  Precision  F1  Matthews Coef  ROC_AUC
0  0.740241  0.794118  0.506826  0.61875  0.461288  0.757445

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ... env_123 (Python 3.8.18)

lightgbm2_env123.ipynb > predictions = best_lgbm.predict(X_test)

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from imblearn.over_sampling import SMOTE
from lightgbm import LGBMClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import joblib

df = pd.read_csv('/Users/laraturunc/Desktop/churn data/Telco-Customer-Churn-Dataset.csv')

df.drop('customerID', axis=1, inplace=True)

df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)

df.fillna(method='ffill', inplace=True)

df['Contract_Tenure'] = df['Contract'].astype(str) + '_' + df['tenure'].astype(str) + '_' + df['PaymentMethod'].astype(str)

services = ['PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']
df['ServicesUsed'] = df[services].sum(axis=1)

le = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = le.fit_transform(df[column])

scaler = StandardScaler()
df.columns = scaler.fit_transform(df)

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb x lightgbm2_env123.ipynb
lightgbm_env123.ipynb > predictions = best_lgbm.predict(X_test)
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells 🔄 Restart 📄 Variables 📄 Outline ... env_123 (Python 3.8.18)

[28] scaler = StandardScaler()
     df.columns = scaler.fit_transform(df)

Python

[29] X = df.drop('Churn', axis=1)
     df = df['Churn']

     le_churn = LabelEncoder()
     y = le_churn.fit_transform(df['Churn'])

     smote = SMOTE()
     X, y = smote.fit_resample(X, y)

Python

[30] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Python

▶ #MACI KIZARTAKAZI DURDURDUM

# from sklearn.model_selection import train_test_split, GridSearchCV

# grid = GridSearchCV(lgbm, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
# grid.fit(X_train, y_train)

# best_lgbm = grid.best_estimator_

# importances = best_lgbm.feature_importances_
# # Selecting features which have non-zero importance
# selected_features = X_train.columns[(importances > 0)]
# X_train_selected = X_train[selected_features]
# X_test_selected = X_test[selected_features]

# best_lgbm.fit(X_train_selected, y_train)

# predictions = best_lgbm.predict(X_test_selected)
# accuracy = accuracy_score(y_test, predictions)

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb x lightgbm2_env123.ipynb
lightgbm_env123.ipynb > predictions = best_lgbm.predict(X_test)
+ Code + Markdown ▶ Run All ⚙ Clear Outputs of All Cells 🔄 Restart 📄 Variables 📄 Outline ... env_123 (Python 3.8.18)

▶ from sklearn.model_selection import RandomizedSearchCV
   from scipy.stats import randint as sp_randint

   param_dist = {
       'num_leaves': sp_randint(20, 50),
       'reg_alpha': [0.1, 0.5],
       'reg_lambda': [0.1, 0.5],
       'min_child_weight': [0.001, 0.01],
       'learning_rate': [0.1, 0.01]
   }

   lgbm = LGBMClassifier(n_estimators=100)

   random_search = RandomizedSearchCV(lgbm, param_distributions=param_dist, n_iter=10, cv=3, scoring='accuracy', n_jobs=-1, random_state=42)
   random_search.fit(X_train, y_train)

   best_lgbm = random_search.best_estimator_

[21]

... Output exceeds the size limit. Open the full output data in a text editor
[LightGBM] [Info] Number of positive: 2429, number of negative: 2400
[LightGBM] [Info] Number of positive: 2429, number of negative: 2400
[LightGBM] [Info] Number of positive: 2429, number of negative: 2400
[LightGBM] [Info] Number of positive: 2429, number of negative: 2400
[LightGBM] [Info] Number of positive: 2428, number of negative: 2400
[LightGBM] [Info] Number of positive: 2429, number of negative: 2400
[LightGBM] [Info] Number of positive: 2428, number of negative: 2400
[LightGBM] [Info] Number of positive: 2428, number of negative: 2400
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.004276 seconds.
You can set 'force_row_wise=true' to remove the overhead.
And if memory is not enough, you can set 'force_col_wise=true'.
[LightGBM] [Info] Total Bins 1573
[LightGBM] [Info] Number of data points in the train set: 4829, number of used features: 19
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.503003 -> initscore=0.012011
[LightGBM] [Info] Start training from score 0.012011
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.006140 seconds.

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb x lightgbm2_env123.ipynb
lightgbm_env123.ipynb > predictions = best_lgbm.predict(X_test)
+ Code + Markdown + Run All + Clear Outputs of All Cells + Restart + Variables + Outline ... env_123 (Python 3.8.18)

And if memory is not enough, you can set 'force_col_wise=True'.
[LightGBM] [Info] Total Bins 1573
[LightGBM] [Info] Number of data points in the train set: 4829, number of used features: 19
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.008096 seconds.
You can set 'force_row_wise=True' to remove the overhead.
And if memory is not enough, you can set 'force_col_wise=True'.
[LightGBM] [Info] Total Bins 1565
[LightGBM] [Info] Number of data points in the train set: 4829, number of used features: 19
...
[LightGBM] [Info] Total Bins 1999
[LightGBM] [Info] Number of data points in the train set: 7243, number of used features: 19
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.582968 -> initscore=0.011874
[LightGBM] [Info] Start training from score 0.011874

>
predictions = best_lgbm.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)

print(f'Best Parameters: {random_search.best_params_}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

joblib.dump(best_lgbm, 'telco_customer_churn_model_improved_NOT.pkl')

[24] Python
... Best Parameters: {'learning_rate': 0.1, 'min_child_weight': 0.01, 'num_leaves': 27, 'reg_alpha': 0.5, 'reg_lambda': 0.1}
Accuracy: 0.8573268921095009
Precision: 0.8585154639175257
Recall: 0.8621815806662312
F1 Score: 0.8563087981394744

['telco_customer_churn_model_improved_NOT.pkl']

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb x lightgbm2_env123.ipynb
lightgbm2_env123.ipynb > from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, StackingClassifier
+ Code + Markdown + Run All + Clear Outputs of All Cells + Restart + Variables + Outline ... env_123 (Python 3.8.18)

>
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from imblearn.over_sampling import SMOTE
from lightgbm import LGBMClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import joblib

df = pd.read_csv('/Users/laraturunc/Desktop/churn_data/Telco-Customer-Churn-Dataset.csv')

df.drop('customerID', axis=1, inplace=True)
df[['TotalCharges']] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df.fillna(df['TotalCharges'].median(), inplace=True)

categorical_cols = df.select_dtypes(include=['object']).columns.drop('Churn')
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns

scaler = StandardScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])

encoder = OneHotEncoder(sparse=False, drop='first')
encoded_categorical_data = encoder.fit_transform(df[categorical_cols])
encoded_categorical_df = pd.DataFrame(encoded_categorical_data, columns=encoder.get_feature_names_out(categorical_cols))

df = df.drop(categorical_cols, axis=1)
df = pd.concat([df.reset_index(drop=True), encoded_categorical_df.reset_index(drop=True)], axis=1)

df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})

X = df.drop('Churn', axis=1)
y = df['Churn']

X, y = SMOTE().fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
lightgbm2_env123.ipynb > from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, StackingClassifier
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ...
env_123 (Python 3.8.18)

lgbm_model = LGBMClassifier()
lgbm_model.fit(X_train, y_train)

predictions = lgbm_model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

joblib.dump(lgbm_model, 'telco_customer_churn_lightgbm_model.pkl')

[30] Python

... /Users/laraturunc/opt/anaconda3/envs/env_123/lib/python3.8/site-packages/sklearn/preprocessing/_encoders.py:972: FutureWarning: 'sparse' was renamed to 'sparse_output' in version 1.2 and will
be removed in 1.4. 'sparse_output' is ignored unless you leave 'sparse' to its default value.
warnings.warn(

[LightGBM] [Warning] Found whitespace in feature_names, replace with underlines
[LightGBM] [Info] Number of positive: 3643, number of negative: 3688
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001844 seconds.
You can set 'force_row_wise=true' to remove the overhead.
And if memory is not enough, you can set 'force_col_wise=true'.
[LightGBM] [Info] Total Bins 2545
[LightGBM] [Info] Number of data points in the train set: 7243, number of used features: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.582968 -> initscore=0.011874
[LightGBM] [Info] Start training from score 0.011874
Accuracy: 0.8557165861513688
Precision: 0.8477842803853564
Recall: 0.8621815806662312
F1 Score: 0.854922279792746

['telco_customer_churn_lightgbm_model.pkl']

```

```

churn_prediction.ipynb • overUndersampling_env123.ipynb • catboost2_env123.ipynb • lightgbm_env123.ipynb • lightgbm2_env123.ipynb
lightgbm2_env123.ipynb > from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, StackingClassifier
+ Code + Markdown ▶ Run All ⌵ Clear Outputs of All Cells ⌵ Restart ⌵ Variables ⌵ Outline ...
env_123 (Python 3.8.18)

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

base_learners = [
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
    ('gb', GradientBoostingClassifier(n_estimators=100, random_state=42)),
    ('svc', make_pipeline(StandardScaler(), SVC(probability=True)))
]

meta_learner = LogisticRegression()

stacking_clf = StackingClassifier(estimators=base_learners, final_estimator=meta_learner, cv=5)

stacking_clf.fit(X_train, y_train)

predictions = stacking_clf.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

[31] Python

... Accuracy: 0.863768115942829
Precision: 0.8611473272498222
Recall: 0.8628347485383723
F1 Score: 0.8619902128717781

```