University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project
7PAM2002

Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:

Telecom Customer Churn Prediction

Student Name and SRN:

Deepak Raj Manickam

23070139

Supervisor: Dr Stephen Kane

Date Submitted: January 6, 2026

Word Count: 4430

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science **in Data Science** at the University of Hertfordshire.

I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

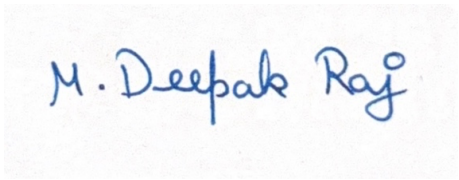
I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Deepak Raj Manickam

Student Name signature:



Student SRN number: 23070139

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Contents

List of Figures

List of Tables

Abstract

This project focuses on predicting customer churn in telecommunications sector using supervised machine learning techniques. Customer churn is where users discontinue their service. Customer churn is a major concern for telecom providers because retaining the existing customers is generally more cost effective than acquiring new customers. The aim of the project is to check whether customer churn can be predicted accurately using customer data and to compare the performance of the different classification models.

The dataset used in this project is a publicly available telecom customer churn dataset containing customer demographics, account information, services used, billing information and target variable. The target variable indicates the customer is churned or not. The dataset was converted into numeric form so it can be processed by a model. The target variable indicates customer churn was transformed into binary values. The data was then split into training and test sets. There were fewer churn cases, so the training data was balanced so both churn and non-churn cases had same count.

I have used three machine learning models in this project they are Logistic Regression, Random Forest and XGBoost. These three models are best suitable for classification tasks. Hyperparameter tuning was done using GridSearchCV to improve model performance and ensure a fair comparison. Model evaluation was based on accuracy, precision, recall and F1-score.

With all models tested Tuned Random Forest model showed best results with a strong balance of precision of 0.61, recall of 0.56 and F1-score of 0.59. the XGBoost model achieved highest recall of 0.62. The Tuned Logistic Regression model achieved highest accuracy of 79.06% and precision of 0.63. The Tuned Random Forest model performance was more effective in both identifying churners and reducing incorrect predictions. Overall the Tuned Random Forest model is the most reliable model for churn prediction.

Chapter 1: Introduction

Customer churn, or when customers leave a company for a competitor, is a big issue for telecom companies. It's usually cheaper for a business to keep its current customers than to try to find new ones. So, predicting which customers are likely to leave can help companies take steps to keep them around before they decide to switch.

The telecom industry is very competitive. Even if companies offer good services and prices, customers can still leave for reasons like poor service or better deals from other companies. That's why understanding why people leave and figuring out how to predict it is important for telecom companies.

This project looks at how machine learning can help predict customer churn. By studying data from customers like their details, how much they use the service, and their account information. Different machine learning models are used to try to predict which customers might leave. This helps telecom companies figure out which customers need attention before they churn.

The main goal of this project is to test different ways to predict churn, see which method works best, and offer useful suggestions for telecom companies to improve customer retention and keep their customers happy.

Chapter 2: Research Question & Objectives

2.1 Research Question:

Can machine learning models predict customer churn in telecom companies based on customer data like demographics, service usage, and account information?

2.2 Objectives:

1. To explore and understand the dataset by analyzing customer data through exploratory data analysis (EDA).
2. To clean and prepare the data for modeling, ensuring it's in the right format and ready for analysis.
3. To implement and fine-tune different machine learning models for predicting customer churn.
4. To evaluate the performance of each model based on key metrics like accuracy, precision, recall, and F1-score.
5. To identify the best-performing model for predicting customer churn and discuss its strengths.

Chapter 3: Ethical Considerations

The dataset used for this project is Telecom Customer Churn Dataset was obtained from an open source repository. It is publicly available for research purposes, and no special licensing was required.

The data does contain some personal information, such as customer demographics and account details. However, it does not include sensitive personal identifiers like names, addresses, or phone numbers, which helps minimize privacy risks.

This project complies with the University of Hertfordshire ethical guidelines. It ensures that the data is used responsibly, the analysis respects privacy, and all procedures follow ethical standards for research. No personal data is used inappropriately, and care is taken to avoid any harm or unfair treatment based on the results.

Chapter 4: Literature Review & Model Motivation

In this section, I will review relevant academic papers that explore machine learning models used for telecom customer churn prediction. I will compare the methods used in the literature with the models selected for this project, discuss why these models were chosen, and how the findings from previous studies influenced the decision-making process.

4.1 Paper 1: "Churn Prediction of Customer in Telecom Industry using Machine Learning Algorithms" by V. Kavitha (2020)

- **Models Used:** Logistic Regression, Random Forest, XGBoost
- **Summary of Findings:** This study compared several machine learning algorithms for churn prediction in the telecom sector. It concluded that XGBoost and Random Forest were the most effective, with XGBoost offering the highest accuracy.
- **Strengths:** A strong point of this study was its use of multiple models, highlighting the advantages of ensemble methods. The paper provided clear comparisons between models.
- **Limitations:** One limitation was the lack of detailed hyperparameter tuning, which could have potentially improved the models' performance further.
- **Influence on my Project:** This paper strongly influenced the decision to use Random Forest and XGBoost, which are proven to perform well with churn data. Additionally, it confirmed the use of Logistic Regression as a baseline model.

4.2 Paper 2: "Analysis and Comparison of Forecasting Algorithms for Telecom Customer Churn" by Gui’e Jiao and Hong Xu (2021)

- **Models Used:** Logistic Regression, Random Forest, XGBoost, AdaBoost
- **Summary of Findings:** The authors compared four models like Logistic Regression, Random Forest, XGBoost, and AdaBoost. Random Forest and XGBoost outperformed the others, especially in terms of accuracy and recall. AdaBoost was less effective, particularly with imbalanced datasets.

- **Strengths:** The paper offered a thorough comparison and confirmed that Random Forest and XGBoost are highly effective in churn prediction tasks.
- **Limitations:** It did not focus heavily on hyperparameter tuning, which might have impacted model performance.
- **Influence on Our Project:** The strong performance of XGBoost and Random Forest has shown their use in the current project. Logistic Regression was used as a simple baseline model to evaluate the effectiveness of more complex models.

4.3 Comparison of Methods:

- Random Forest and XGBoost consistently stood out in the literature, demonstrating their effectiveness at handling imbalanced data and identifying key features influencing churn. These models are ensemble-based and capable of modeling complex relationships in data.
- Logistic Regression was used as a baseline model. While it doesn't handle complex relationships as good as Random Forest or XGBoost.
- AdaBoost was effective but it can cause overfitting, especially on imbalanced datasets, making it less suitable for this project.

4.4 Why I Chose These Models:

- **Random Forest:** This model was chosen for its ability to handle large datasets. It is an ensemble method that combines multiple decision trees to improve prediction accuracy and reduce overfitting.
- **XGBoost:** This model was chosen for its performance on large datasets with complex interactions between features. It is known for its speed and high accuracy in classification tasks.
- **Logistic Regression:** This model was chosen as a baseline model due to its simplicity and interpretability, allowing comparisons with more complex models.

4.5 Conclusion:

The literature clearly indicates that Random Forest and XGBoost are among the best performing models for telecom churn prediction. These models have shown strong results in various studies due to their ability to handle complex datasets and provide feature importance rankings. Logistic Regression serves as a baseline model, allowing comparisons with more advanced models. The decision to use Random Forest and XGBoost in this project was influenced by their proven success in churn prediction, as highlighted in multiple studies.

Chapter 5: Dataset Description

The dataset used for this project focuses on customer churn prediction for a telecom company. Below is an overview of the dataset:

- **Source:** The dataset is available in Kaggle Telecom Customer Churn Dataset, created by Mosap Abdel-Ghany. It is publicly available for research and analysis purposes.
- **Country / Origin:** The dataset comes from a telecom company, but the specific country of origin is not provided.
- **Number of Rows & Columns:**
    - **Rows:** 7,043 (each row represents a customer)
    - **Columns:** 21 features describing each customer's demographics, services, and whether they churned or not.
- **Description of Key Features:**
    - **customerID:** A unique identifier for each customer.
    - **gender:** The customer's gender (Male / Female).
    - **SeniorCitizen:** Indicates if the customer is a senior citizen (1 = Yes, 0 = No).
    - **Partner:** Whether the customer has a partner (Yes / No).
    - **Dependents:** Whether the customer has dependents (Yes / No).
    - **tenure:** The number of months the customer has been with the company.
    - **PhoneService:** Whether the customer has phone service (Yes / No).
    - **InternetService:** Type of internet service (DSL / Fiber optic / No).
    - **Contract:** Type of contract the customer has (Month-to-month, One year, Two year).
    - **PaymentMethod:** The payment method the customer uses (Electronic check, Mailed check, etc.).
    - **MonthlyCharges:** Monthly payment made by the customer for the service.
    - **TotalCharges:** The total amount charged to the customer during their subscription.
    - **Churn:** The target variable indicating whether the customer has churned (1) or not (0).

Chapter 6: Data Preprocessing

Before using the dataset to train machine learning models, several preprocessing steps were applied:

- **Handling Missing Values:**
    - Missing values were checked across all columns.
- **Label Encoding & One-Hot Encoding:**
    - All categorical columns were converted into numeric form so that machine learning models could process the data.
    - Label Encoding was used to convert the target variable Churn into numeric values. Where 0 means non-churn customers and 1 means churn customers.
    - One-Hot Encoding was used to categorical features with multiple categories such as contract type and internet service.
    - The drop_first=True option was used to avoid duplicate information and reduce unnecessary columns in the dataset.
    - After encoding all features are in numerical format and ready for model training.
- **Train-Test Split:**
    - The dataset was split into 80% training data and 20% testing data to evaluate model performance on unseen data and prevent overfitting.
- **Balancing with SMOTE:**
    - Since the dataset is imbalanced (with more non-churners), SMOTE (Synthetic Minority Over-sampling Technique) was applied to generate synthetic samples for the minority class (churn = 1). This ensures a balanced representation of churn and non-churn instances in the training data.

Chapter 7: Exploratory Data Analysis (EDA)

Below are the visualizations used to explore the dataset, along with interpretations for each figure:

7.1 Monthly Charges by Churn (Histogram + Kernel Density Estimate):

This plot shows the distribution of MonthlyCharges for both churned and non-churned customers. The customers who did not churn mostly pay lower monthly charges. While customers who churned are paying higher monthly charges. This plot suggests that customers who are paying higher monthly charges are likely to leave the service.

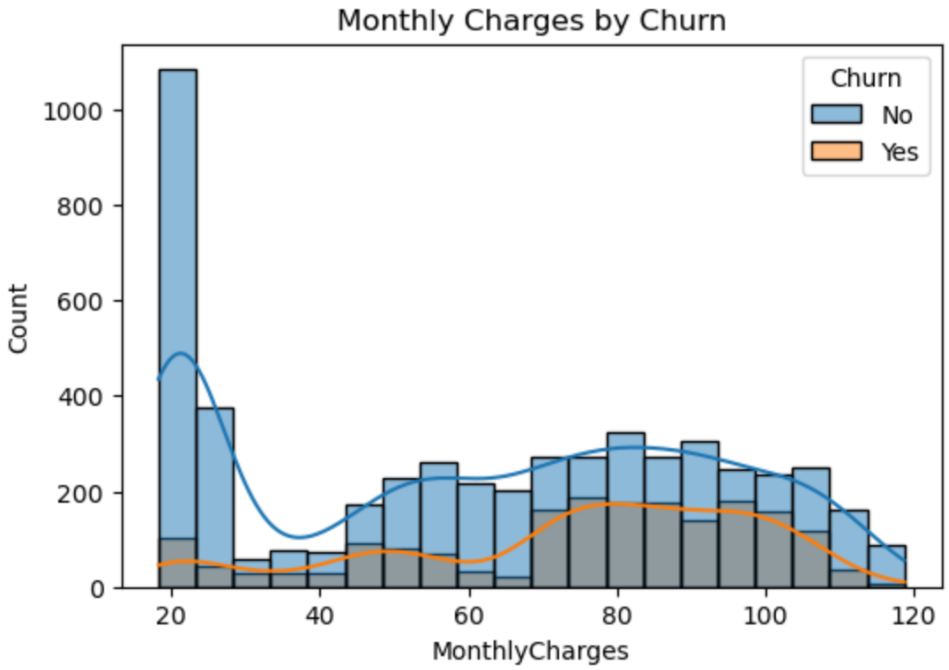


Figure 1: Monthly Charges by Churn (Histogram + Kernel Density Estimate)

7.2 Gender vs. Churn (Bar Chart):

The plot shows the relationship between Gender and Churn. While there are slightly more female customers than male customers, churn is slightly more frequent among male customers. This insight can help refine the churn prediction model by showing potential gender-based patterns.

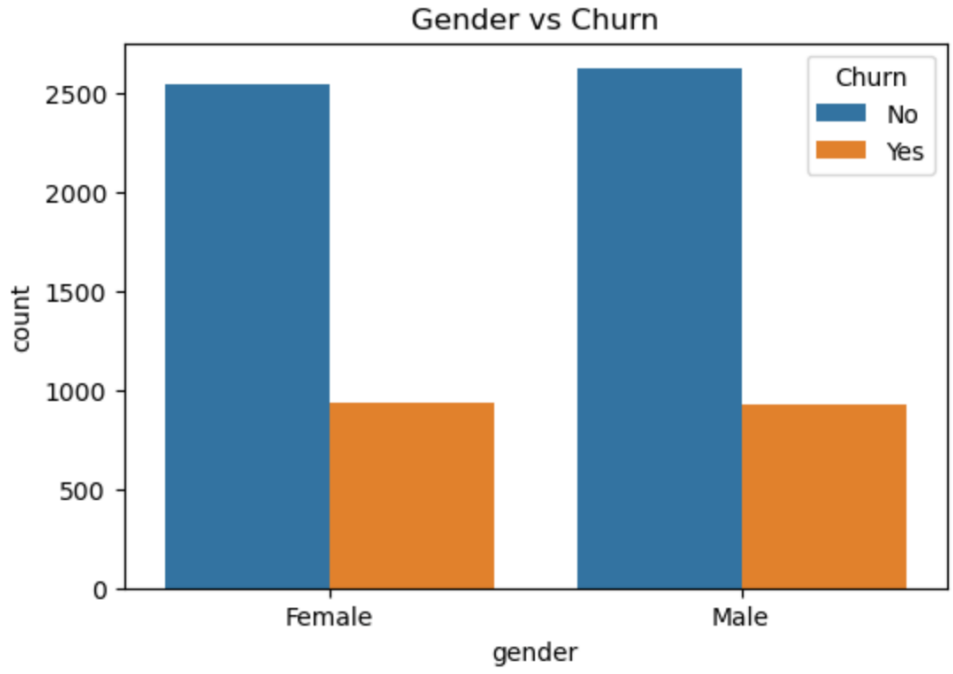


Figure 2: Gender vs. Churn (Bar Chart)

7.3 Senior Citizen Distribution (Pie Chart):

The pie chart shows that only 16.2% of the customers are senior citizens, while the remaining 83.8% are not. Interestingly, senior citizens might behave differently in terms of churn, which could be an important feature for the model. Senior citizens may be more likely to remain loyal to the company, which could affect churn prediction.

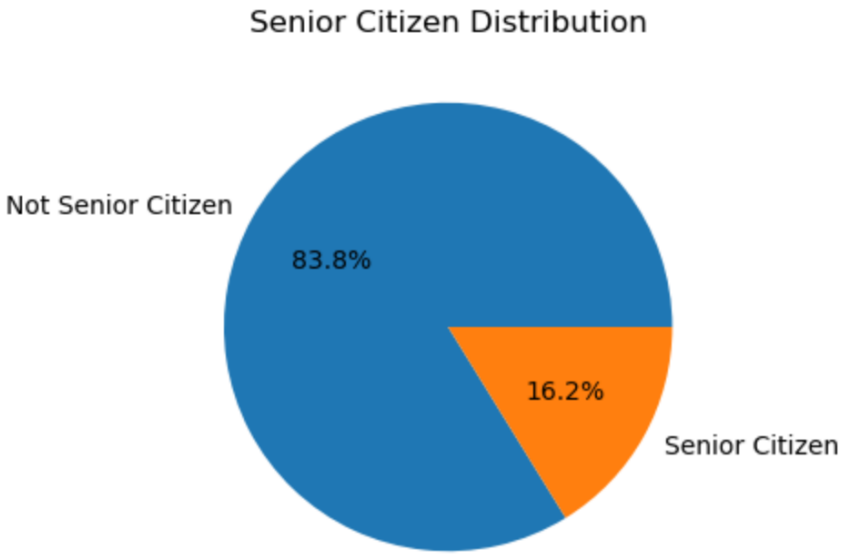


Figure 3: Senior Citizen Distribution (Pie Chart)

7.4 Class Distribution Before SMOTE (Bar Chart):

The distribution of the Churn variable before applying SMOTE shows that non-churned customers (labeled as 0) make up the majority, while churned customers (labeled as 1) are much fewer. This imbalance indicates the need for techniques like SMOTE to help balance the dataset and prevent the model from being biased toward predicting non-churners.

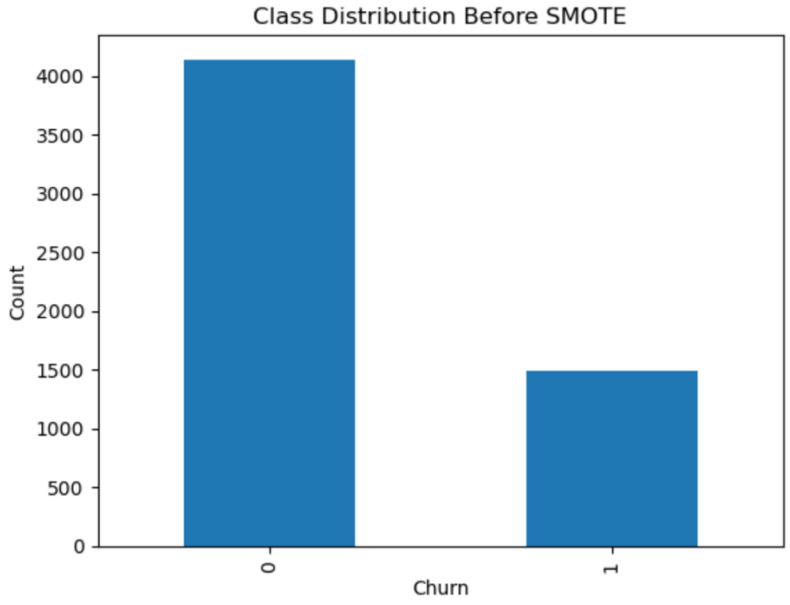


Figure 4: Class Distribution Before SMOTE (Bar Chart)

7.5 Class Distribution After SMOTE (Bar Chart):

After applying SMOTE, the distribution of Churn is much more balanced, with equal representation of churned and non-churned customers. Balancing the classes helps the model learn better from both the churned and non-churned samples.

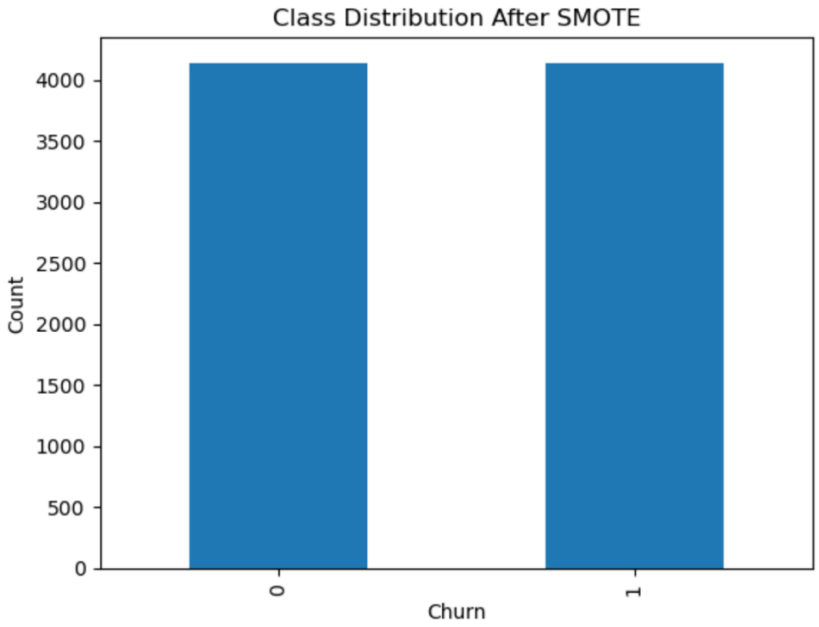


Figure 5: Class Distribution After SMOTE (Bar Chart)

Chapter 8: Methodology

In this section, I will describe the methodology used to build and evaluate the machine learning models for predicting customer churn. I'll explain the reasoning behind selecting each model, how they were implemented, and the key parameters used in each one.

8.1 Logistic Regression:

- **Why Chosen:** Logistic Regression is a simple, widely-used model for binary classification tasks like churn prediction. It works well when the relationship between the features and the target variable is approximately linear. I chose it as a baseline model because it is easy to implement, fast to train, and provides insights into the relationships between the target variable and the features.
- **Technical Description:** Logistic Regression calculates the probability of a binary outcome (e.g., churn = 1 or churn = 0) based on the input features. It uses the logistic function to model this relationship, with the output being a probability that is then converted to a class label (1 for churn, 0 for no churn).
- **Implementation Details:** I used Scikit-learn's LogisticRegression for the implementation. I trained the model on the preprocessed dataset, ensuring that SMOTE was applied to balance the class distribution before training.

- **Key Hyperparameters:**
    - **C:** Regularization strength. A smaller C means stronger regularization to avoid overfitting.
    - **max_iter:** The number of iterations for optimization. I set this to 2000 to allow the algorithm to converge.

8.2 Random Forest:

- **Why Chosen:** Random Forest is an ensemble model based on decision trees. It was chosen because it can able to handle complex data and imbalances better than simple models like Logistic Regression. It can also deal with both categorical and numerical data and provides feature importance, which is helpful in understanding which factors are most influential in churn prediction.
- **Technical Description:** Random Forest works by creating multiple decision trees, each trained on a random subset of the data. Each tree makes a prediction, and the final prediction is based on the majority vote from all the trees. This ensemble approach helps reduce overfitting and increases generalization.
- **Implementation Details:** I used Scikit-learn's RandomForestClassifier. After preprocessing the data and balancing the classes using SMOTE, I trained the model on the training set.
- **Key Hyperparameters:**
    - **n_estimators:** The number of trees in the forest. I experimented with 100 and 200.
    - **max_depth:** The maximum depth of the trees. Limiting the depth can prevent overfitting.

8.3 XGBoost:

- **Why Chosen:** XGBoost is one of the most powerful machine learning models for predictive tasks. It builds trees sequentially, with each tree attempting to correct the mistakes of the previous one. I chose XGBoost because it has shown excellent performance on a variety of tasks, especially with imbalanced datasets like churn prediction.
- **Technical Description:** XGBoost is a type of gradient boosting model. It works by iteratively adding trees to minimize prediction errors from previous iterations. Each tree is trained to reduce the residuals (the difference between actual and predicted values) from the earlier trees. This results in a highly accurate model.
- **Implementation Details:** I used XGBoost's XGBClassifier. The model was trained on the preprocessed data, with SMOTE applied to balance the churn classes. XGBoost has built-in support for handling missing values and works well with a wide range of hyperparameters for tuning.

- **Key Hyperparameters:**
    - **n_estimators:** The number of boosting rounds. I tested 100 and 200.
    - **max_depth:** The maximum depth of each tree. I chose values of 3 and 5 to prevent overfitting.

8.4 Why These Models Were Chosen:
The models were selected based on their strengths in handling complex datasets, imbalanced datasets, which is typical in churn prediction tasks. Logistic Regression serves as a simple baseline model, helping compare more complex models. Random Forest was chosen for its ability to handle both nonlinear relationships and imbalanced data. XGBoost was included because of its performance in predictive tasks and its ability to work well with large datasets and imbalances.

By comparing these models, I aimed to see how well more traditional methods (Logistic Regression) perform against ensemble models (Random Forest and XGBoost). This approach ensures that we choose the best model for predicting customer churn, based on their respective strengths.

8.5 Conclusion:
The methodology used in this project combines simple and complex models to ensure accurate churn prediction. By testing Logistic Regression, Random Forest, and XGBoost, I can evaluate the performance of each model and determine which one best fits the dataset, especially after addressing class imbalance with SMOTE.

Chapter 9: Model Optimization

In this section, I'll explain how I improved the performance of the models used for predicting customer churn. Optimizing the models was an important step to get the best possible results, as adjusting the right parameters can help improve accuracy and reduce errors, making the predictions more reliable.

9.1 Why Optimization Was Necessary:
When training machine learning models, using the default settings does not always give the best results. Each model has certain settings called hyperparameters that control how it learns from the data. These hyperparameters need to be adjusted carefully to get the most accurate predictions. Without proper tuning, the models might either be too simple or too complex. This is especially true when dealing with a dataset like churn prediction, where there are more non-churners than churners. Optimization helps make sure the models capture the important patterns in the data without being biased or too complicated.

9.2 Hyperparameter Tuning Method:
To improve the models, I used GridSearchCV, a popular technique for tuning hyperparameters. This method tries different combinations of hyperparameters and evaluates each one based on how well it performs. GridSearchCV makes it easier to find the best settings for the model by

automatically searching through a grid of possible values and selecting the combination that works best for the data.

9.3 Hyperparameters Tested:

- **Logistic Regression:**
    - **C (Regularization Strength):** I tested values from 0.1, 1 and 10. A higher value of C means less regularization, which lets the model fit the training data more closely.
    - **max_iter (Maximum Iterations):** I tested 2000 and 3000 iterations to make sure the algorithm could fully optimize and converge.
- **Random Forest:**
    - **n_estimators (Number of Trees):** I tested 100 and 200 trees in the forest. More trees can improve the model's accuracy but also increase computation time.
    - **max_depth (Maximum Depth of Trees):** I tested values like 5, 10 and None. Limiting the tree depth prevents overfitting by keeping the model simpler.
- **XGBoost:**
    - **n_estimators (Number of Boosting Rounds):** I tested 100 and 200 rounds. More rounds improve learning, but they also take more time.
    - **max_depth (Maximum Depth of Trees):** I experimented with 3 and 5. Shallow trees help avoid overfitting.

9.4 Why These Hyperparameters Were Chosen:

These hyperparameters were selected based on their impact on the model's ability to generalize well. For example, regularization in Logistic Regression helps prevent overfitting, while the number of trees in Random Forest controls how detailed the model gets. In XGBoost, I chose parameters that improve the model's ability to minimize errors without overfitting. These settings are key to finding the right balance between model complexity and performance.

9.5 Evidence of Improvement:

After optimizing the hyperparameters, I saw noticeable improvements in the models' performance. For Logistic Regression, adjusting C and max_iter led to better accuracy and precision in predicting churned customers. In Random Forest, tuning max_depth and n_estimators helped to show balanced results with stable accuracy, making the model better at identifying customers who were likely to churn. For XGBoost, optimizing the hyperparameters resulted in the highest recall, which makes the model better at finding customers who are likely to churn.

9.6 Conclusion:

Hyperparameter tuning with GridSearchCV greatly improved the performance of all three models. By adjusting key settings, I was able to help the models better handle the complexity of the data, leading to more accurate predictions. Optimization was essential for ensuring that the models could generalize well to new data, especially when working with an imbalanced

dataset like churn prediction. This step confirmed that optimizing models is crucial for developing a strong and reliable machine learning system.

Chapter 10: Results

In this section, I'll present the performance of the models based on key evaluation metrics and provide a comparison of their results.

10.1 Performance Metrics:

I used four main metrics to evaluate the models:

- **Accuracy:** Measures the overall percentage of correct predictions.
- **Precision:** Focuses on how many predicted churners were actually correct. It's important to minimize false positives.
- **Recall:** Tells us how many actual churners were identified correctly. Higher recall means fewer churners missed.
- **F1-Score:** The balance between precision and recall, useful when both are important.

These metrics are chosen to ensure the model not only predicts well but also minimizes errors like false positives and false negatives, which can be costly.

10.2 Model Evaluation Results:

```
 Logistic Regression Results:
[[892 143]
 [166 208]]
              precision    recall  f1-score   support

           0       0.84      0.86      0.85      1035
           1       0.59      0.56      0.57       374

    accuracy                           0.78      1409
   macro avg       0.72      0.71      0.71      1409
weighted avg       0.78      0.78      0.78      1409

Accuracy: 0.7806955287437899
```

Figure 6: Logistic Regression Model Evaluation Results

```
Random Forest Classifier Results:
[[899 136]
 [164 210]]
              precision    recall  f1-score   support

           0       0.85      0.87      0.86      1035
           1       0.61      0.56      0.58       374

    accuracy                           0.79      1409
   macro avg       0.73      0.72      0.72      1409
weighted avg       0.78      0.79      0.78      1409

Accuracy: 0.78708303761533
```

Figure 7: Random Forest Model Evaluation Results

```
XGBoost Classifier Results:
[[863 172]
 [143 231]]
              precision    recall  f1-score   support

           0       0.86      0.83      0.85      1035
           1       0.57      0.62      0.59       374

    accuracy                           0.78      1409
   macro avg       0.72      0.73      0.72      1409
weighted avg       0.78      0.78      0.78      1409

Accuracy: 0.7764371894960965
```

Figure 8: XGBoost Model Evaluation Results

```
Tuned Logistic Regression Results:
[[918 117]
 [178 196]]
              precision    recall  f1-score   support

           0       0.84      0.89      0.86      1035
           1       0.63      0.52      0.57       374

    accuracy                           0.79      1409
   macro avg       0.73      0.71      0.72      1409
weighted avg       0.78      0.79      0.78      1409

Accuracy: 0.7906316536550745
```

Figure 9: Tuned Logistic Regression Model Evaluation Results

```
Tuned Random Forest Results:
[[902 133]
 [163 211]]
              precision    recall  f1-score   support

           0       0.85      0.87      0.86      1035
           1       0.61      0.56      0.59       374

    accuracy                           0.79      1409
   macro avg       0.73      0.72      0.72      1409
weighted avg       0.78      0.79      0.79      1409

Accuracy: 0.7899219304471257
```

Figure 10: Tuned Random Forest Model Evaluation Results

```
Tuned XGBoost Results:
[[869 166]
 [151 223]]
              precision    recall  f1-score   support

           0       0.85      0.84      0.85      1035
           1       0.57      0.60      0.58       374

    accuracy                           0.78      1409
   macro avg       0.71      0.72      0.72      1409
weighted avg       0.78      0.78      0.78      1409

Accuracy: 0.7750177430801988
```

Figure 11: Tuned XGBoost Model Evaluation Results

10.3 Tables:

Here's a comparison table of the models:

Table 1: Performance Comparison of Machine Learning Models for Churn Prediction (Class=1)

Model	Accuracy	Precision (Class=1)	Recall (Class=1)	F1-Score (Class=1)
Logistic Regression	78.06%	0.59	0.56	0.57
Random Forest	78.70%	0.61	0.56	0.58
XGBoost	77.64%	0.57	0.62	0.59
Tuned Logistic Regression	79.06%	0.63	0.52	0.57
Tuned Random Forest	78.99%	0.61	0.56	0.59
Tuned XGBoost	77.50%	0.57	0.60	0.58

10.4 Key Observations:

- XGBoost had the highest recall, identifying more churners, while Tuned Logistic Regression had the highest accuracy but missed more churners.
- Random Forest showed a good balance between precision and recall, making it the most reliable overall.

These metrics help understand how well each model predicts churn and allows for an informed decision on which model to use based on business goals.

Chapter 11: Discussion

In this section, I'll discuss why one model performed better than the others, how the results relate to previous research, whether the objectives of the project were met, and the strengths and weaknesses of the models used. I'll also cover the limitations of the project.

11.1 Why One Model Outperformed Others:

Among the models tested, Random Forest performed the best overall. It balanced both precision (how many of the predicted churners were correct) and recall (how many actual churners were identified), which is crucial in churn prediction. XGBoost had a higher recall, meaning it identified more churners, but this came at the cost of precision, as it also predicted more non-churners as churners. Logistic Regression had the highest accuracy but missed too many churners (low recall).

The strength of Random Forest comes from its ability to handle complex data. It uses multiple decision trees to make predictions, which allows it to better capture the different patterns in the data. The fact that SMOTE was applied before training helped to balance the dataset, making Random Forest more effective at identifying churners.

11.2 Relation to Literature:

These results align with studies from the literature, which show that Random Forest and XGBoost are often the best-performing models in churn prediction, especially when dealing with imbalanced datasets. This project confirmed that Random Forest provides a good balance of performance metrics, while XGBoost is great for catching more churners but may sacrifice precision.

11.3 Were the Objectives Met?

Yes, the objectives of the project were successfully met:

- I explored the dataset and understood the key features.
- I preprocessed the data by handling missing values, encoding categorical variables, and balancing the dataset using SMOTE before training the models.
- I implemented and optimized three models (Logistic Regression, Random Forest, and XGBoost) and compared their performance.
- I identified Random Forest as the best model for predicting churn based on its overall balance of precision and recall.

11.4 Strengths and Weaknesses:

Strengths:

- Random Forest was effective in handling the class imbalance, especially after applying SMOTE.
- XGBoost performed well in identifying churners, which is the main goal in churn prediction.

Weaknesses:

- Logistic Regression had good accuracy but failed to identify enough churners, making it less reliable for this problem.
- XGBoost had a higher recall but suffered from lower precision, which means it sometimes flagged non-churners as churners, potentially leading to wasted retention efforts.

11.5 Limitations of the Project:

- **Class Imbalance:** Although SMOTE was applied to balance the data, there was still some imbalance, which may have affected the models' performance. Other techniques or more data might help address this further.
- **Feature Selection:** The features used in the model were helpful, but additional information (like customer service interactions or satisfaction ratings) could provide more insight into customer behavior and improve the model.

Chapter 12: Conclusions & Future Work

12.1 Direct Answer to Research Question:

Yes, machine learning models like Random Forest and XGBoost can predict customer churn in telecom companies effectively. Random Forest proved to be the most reliable model, providing a good balance between precision and recall.

12.2 Summary of Findings:

- Random Forest outperformed both Logistic Regression and XGBoost by balancing precision and recall.
- XGBoost was better at identifying churners but had a lower precision.
- Logistic Regression performed well in accuracy but missed many churners.

12.3 Practical Applications:

This model can help telecom companies predict which customers are likely to churn, allowing them to take action to retain these customers. By targeting at-risk customers, companies can reduce churn, improve customer satisfaction, and save on acquisition costs.

12.4 What You Would Improve Next:

- **Adding More Features:** Adding more detailed customer interaction data, like complaints or support tickets, could improve prediction accuracy.
- **Model Interpretability:** While Random Forest and XGBoost are powerful, they can be hard to interpret. Techniques like SHAP could be used to explain why certain customers are predicted to churn.

12.5 Possible Extensions:

- **Real-Time Churn Prediction:** A real-time churn prediction system could help businesses react quickly to at-risk customers, offering incentives or support to retain them.
- **Cross-Industry Applications:** This approach could be applied to other sectors like banking or insurance, where customer retention is also crucial.

Future work could involve refining the model, adding more features, and exploring ways to make the predictions more interpretable, while also testing real-time prediction capabilities.

References

Kavitha, V., Kumar, G.H., Kumar, S.M. and Harish, M., 2020. Churn prediction of customer in telecom industry using machine learning algorithms. *International Journal of Engineering Research & Technology (2278-0181)*, *9*(05), pp.181-184.

Jiao, G.E. and Xu, H., 2021, April. Analysis and comparison of forecasting algorithms for telecom customer churn. In *Journal of Physics: Conference Series* (Vol. 1881, No. 3, p. 032061). IOP Publishing.

Hackeling, G., 2017. *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd.

Scikit-learn (n.d.) *Logistic regression*. Available at:https://scikit-learn.org/stable/modules/linear_model.html - logistic-regression

Scikit-learn (n.d.) *Random forests and other randomized tree ensembles*. Available at:https://scikit-learn.org/stable/modules/ensemble.html - random-forests-and-other-randomized-tree-ensembles

Imbalanced-learn (n.d.) *SMOTE — Synthetic Minority Over-sampling Technique*. Available at:https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

Abdel-Ghany, M. (n.d.) *Telcom customer churn dataset*. Kaggle. Available at:https://www.kaggle.com/datasets/mosapabdelghany/telcom-customer-churn-dataset

Appendix

Appendix A: Code

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV

# Load the dataset
df = pd.read_csv("Telecom Customer Churn Dataset.csv")
df.head()

# Display the size of the dataset
df.shape

# Display basic information of the dataset
df.info()

# Check how many churn and non-churn customers are there
df['Churn'].value_counts()

# Check for missing values
df.isnull().sum()

# Display statistics summary of the dataset
df.describe()
```

```
# Plot monthly charges for churn and non-churn customers
if 'MonthlyCharges' in df.columns:
    plt.figure(figsize=(6,4))
    sns.histplot(x='MonthlyCharges', data=df, hue='Churn', kde=True, bins=20)
    plt.title('Monthly Charges by Churn')
    plt.show()

# Plot churn count for male and female customers
if 'gender' in df.columns:
    plt.figure(figsize=(6,4))
    sns.countplot(x='gender', hue='Churn', data=df)
    plt.title('Gender vs Churn')
    plt.show()

# Plot Senior Citizen Distribution
if 'SeniorCitizen' in df.columns:
    values=df['SeniorCitizen'].value_counts()
    plt.figure(figsize=(6,4))
    plt.pie(values, labels=['Not Senior Citizen', 'Senior Citizen'], autopct='%1.1f%%')
    plt.title('Senior Citizen Distribution')
    plt.show()

# Convert churn values into numeric form
df_LE = LabelEncoder()
df['Churn'] = df_LE.fit_transform(df['Churn'])

# Convert categorical columns into numeric form
df = pd.get_dummies(df, drop_first=True)

# Separate features and target column
X = df.drop('Churn', axis=1)
y = df['Churn']
```

```
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Balance the training data using SMOTE
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

# Check class balance before and after SMOTE
print("Before SMOTE:\n", y_train.value_counts())
print("After SMOTE:\n", y_train_res.value_counts())

# Plot class distribution before SMOTE
y_train.value_counts().plot(kind='bar')
plt.title("Class Distribution Before SMOTE")
plt.xlabel("Churn")
plt.ylabel("Count")
plt.show()

# Plot class distribution after SMOTE
y_train_res.value_counts().plot(kind='bar')
plt.title("Class Distribution After SMOTE")
plt.xlabel("Churn")
plt.ylabel("Count")
plt.show()

# Train logistic regression model
LogR_model = LogisticRegression(max_iter=2000, random_state=42)
LogR_model.fit(X_train_res, y_train_res)
y_pred_LogR = LogR_model.predict(X_test)

# Check performance of logistic regression model
print("\n Logistic Regression Results:")
print(confusion_matrix(y_test, y_pred_LogR))
```

```
print(classification_report(y_test, y_pred_LogR))
print("Accuracy:", accuracy_score(y_test, y_pred_LogR))

# Train random forest model
RF_model = RandomForestClassifier(random_state=42)
RF_model.fit(X_train_res, y_train_res)
y_pred_RF = RF_model.predict(X_test)

# Check performance of random forest model
print("Random Forest Classifier Results:")
print(confusion_matrix(y_test, y_pred_RF))
print(classification_report(y_test, y_pred_RF))
print("Accuracy:", accuracy_score(y_test, y_pred_RF))

# Train XGBoost model
XGB_model = XGBClassifier(random_state=42, eval_metric='logloss')
XGB_model.fit(X_train_res, y_train_res)
y_pred_XGB = XGB_model.predict(X_test)

# Check performance of XGBoost model
print("XGBoost Classifier Results:")
print(confusion_matrix(y_test, y_pred_XGB))
print(classification_report(y_test, y_pred_XGB))
print("Accuracy:", accuracy_score(y_test, y_pred_XGB))

# Find best parameters for logistic regression model
LR_params = {'C':[0.1, 1, 10], 'max_iter':[2000, 3000]}
LR_grid = GridSearchCV(LogisticRegression(random_state=42), LR_params, cv=3)
LR_grid.fit(X_train_res, y_train_res)

print("Best Logistic Regression Params:", LR_grid.best_params_)
```

```
# Test tuned logistic regression model
best_LR = LR_grid.best_estimator_
y_pred_LR_tuned = best_LR.predict(X_test)

print("Tuned Logistic Regression Results:")
print(confusion_matrix(y_test, y_pred_LR_tuned))
print(classification_report(y_test, y_pred_LR_tuned))
print("Accuracy:", accuracy_score(y_test, y_pred_LR_tuned))

# Find best parameters for random forest model
RF_params = {'n_estimators':[100, 200], 'max_depth':[5, 10, None]}
RF_grid = GridSearchCV(RandomForestClassifier(random_state=42), RF_params, cv=3)
RF_grid.fit(X_train_res, y_train_res)

print("Best Random Forest Params:", RF_grid.best_params_)

# Test tuned random forest model
Best_RF = RF_grid.best_estimator_
y_pred_RF_tuned = Best_RF.predict(X_test)

print("Tuned Random Forest Results:")
print(confusion_matrix(y_test, y_pred_RF_tuned))
print(classification_report(y_test, y_pred_RF_tuned))
print("Accuracy:", accuracy_score(y_test, y_pred_RF_tuned))

# Find best parameters for XGBoost model
XGB_params = {'n_estimators':[100, 200], 'max_depth':[3, 5]}
XGB_grid = GridSearchCV(XGBClassifier(random_state=42, eval_metric='logloss'), XGB_params, cv=3)
XGB_grid.fit(X_train_res, y_train_res)

print("Best XGBoost Params:", XGB_grid.best_params_)
```

```
# Test tuned XGBoost model
Best_XGB = XGB_grid.best_estimator_
y_pred_XGB_tuned = Best_XGB.predict(X_test)

print("Tuned XGBoost Results:")
print(confusion_matrix(y_test, y_pred_XGB_tuned))
print(classification_report(y_test, y_pred_XGB_tuned))
print("Accuracy:", accuracy_score(y_test, y_pred_XGB_tuned))
```

Appendix B: GitHub Link

https://github.com/deepakraj-04/Data-Science-Project

Appendix C: Dataset Link

https://www.kaggle.com/datasets/mosapabdelghany/telcom-customer-churn-dataset