Introduction

This project aims to analyze customer churn data from a telecommunications company to understand the factors contributing to customer churn and to build a predictive model. Customer churn refers to the phenomenon where customers stop using a company's services over a given period. Understanding and predicting churn is crucial for businesses as retaining existing customers is often more cost-effective than acquiring new ones.

Dataset Overview

The dataset used in this project is the "Telco Customer Churn" dataset, which includes the following information:

- Customers who left within the last month, indicated by the column "Churn".
- Services that each customer has signed up for, such as phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies.
- Customer account information, including how long they've been a customer, their contract type, payment method, paperless billing status, monthly charges, and total charges.
- Demographic information about customers, including gender, age range, and whether they have partners and dependents.

Preprocessing

The preprocessing steps are crucial to ensure that the data is clean and suitable for modeling

- Using the info function, we can observe the number of columns and rows, the data types of each column, and detect missing values. The dataset comprises 2 columns of integer type, 1 column of float type, and 18 columns of object type.
- Specifically, there are 7043 rows and 21 columns in the dataset, with 2 columns being of integer type, 1 column of float type, and 18 columns of object type.
- Interestingly, the TotalCharges column is categorized as object type, despite its actual data type being float.

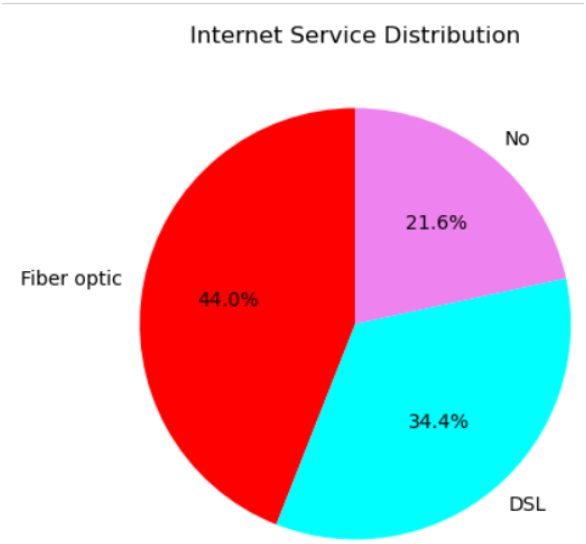
The TotalCharges column originally had 11 missing values. These missing values were addressed by dropping them from the dataset. Since the number of missing values was small (less than 1% of the total dataset), this approach is generally acceptable.

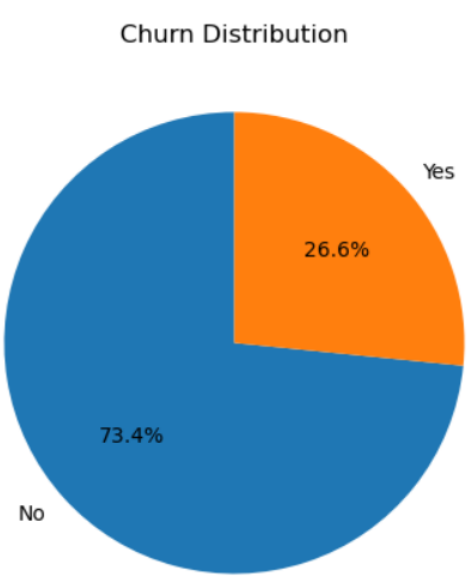
Data Visualization

- The churn distribution indicates that 73.4% of customers did not churn, while 26.6% of customers did churn.

- The distribution of internet services shows that 44% of customers have Fiber Optics, 21.6% do not have internet service, and 34.4% have DSL.

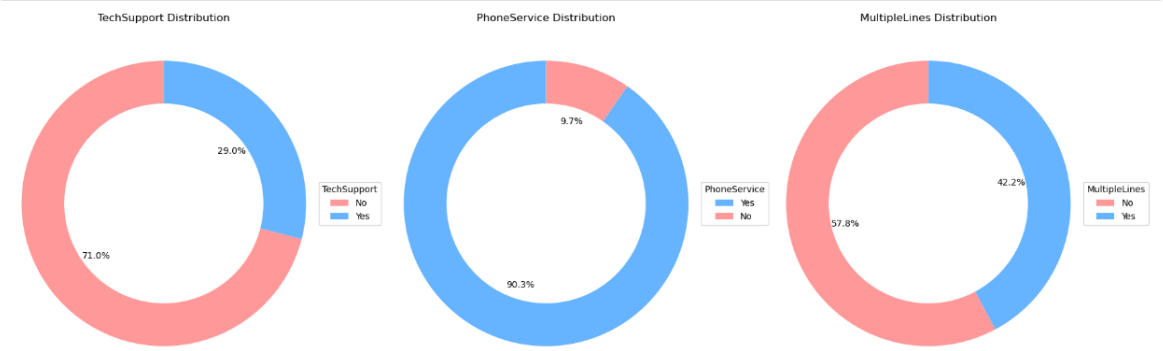
- The distribution of tech support service indicates that 71% of customers do not





have tech support, while 29% of customers have tech support.

- The distribution of phone service shows that 90.3% of customers have phone service, while 9.7% do not have phone service.
- The distribution of multiple lines indicates that 57.8% of customers do not have multiple lines, while 42.2% of customers have multiple lines.



Label Encoding: Converts each category into a different integer. Useful for ordinal data where there is a meaningful order

Label encoding is suitable for categorical variables that have an inherent ordinal relationship.

Data Splitting for Training and Testing

In preparation for model development, the dataset was split into training and testing sets using a test size of 20% and a random state of 0. This procedure ensures that the model's performance can be accurately evaluated on unseen data.

The dataset was divided into two sets:

- **Training Set**: Used to train the machine learning model. It comprises 80% of the original data.
- **Testing Set**: Reserved for evaluating the model's performance. It consists of the remaining 20% of the original data.

The splitting process was conducted using the train_test_split function from the sklearn.model_selection module in Python. The random state parameter was set to 0 to ensure the reproducibility of the split.

This methodical approach to data splitting helps in maintaining the integrity and generalizability of the model, allowing for robust evaluation and validation of its predictive capabilities.

Data Standardization

To ensure consistent and effective modeling, the data underwent standardization to scale all features onto a single scale. This process is crucial for models that are sensitive to the scale of the input data, such as support vector machines, k-nearest neighbors, and neural networks.

Data Standardization: Standardization transforms the data such that it has a mean of 0 and a standard deviation of 1. This transformation is applied independently to each feature.

Implementation: The data was standardized after splitting it into training and testing sets. This ensures that the training and testing data are transformed in a consistent manner.

Improved Model Performance: Standardization ensures that all features contribute equally to the modeling process, preventing any particular feature from dominating due to its larger scale.

Model Training

I conducted model training using logistic regression and random forest classifiers. Initially, both models achieved an accuracy of 74% on the testing data, suggesting they performed similarly on unseen data. To improve their performance, I employed GridSearchCV and RandomizedSearchCV for fine-tuning.

After fine-tuning, the logistic regression model's accuracy significantly increased to more than 92%. This improvement demonstrates the effectiveness of the hyperparameter tuning in enhancing the model's predictive capability. On the other hand, the random forest classifier also showed notable enhancement, achieving an accuracy of almost 95%. These results

highlight the successful refinement of both models, allowing them to better generalize to unseen data and improving their overall performance.

Before applying fine-tuning techniques, I employed the SMOTEENN method to balance the dataset. SMOTEENN combines the Synthetic Minority Over-sampling Technique (SMOTE) for oversampling the minority class with Edited Nearest Neighbors (ENN) for undersampling the majority class. This approach generates synthetic data points and cleans noisy samples, making it particularly useful for preparing the dataset for model training.

SMOTEENN was chosen because it effectively addresses class imbalances by creating synthetic examples of the minority class and removing examples from the majority class that are considered noise or outliers. By doing so, it helps to create a more balanced and cleaner dataset, which is crucial for training accurate machine learning models.

I also applied other algorithms, namely Xtreme Gradient Boosting (XGBoost) and Gradient Boosting, with fine-tuning. Both algorithms showed similar performance on the testing data, achieving an accuracy of almost 95%. This demonstrates that these models are robust and effective in predicting outcomes on unseen data, further validating the success of the fine-tuning process in optimizing their performance.