PROJECT REPORT

A Predictive Model for Customer Churn in Telecommunications

Nallapuneni Vamsi Krishna

vamsikrishnanallapuneni@gmail.com

ROAD MAP: -

1. **Import Required Libraries**
2. **Adjusting Row Column Settings**
3. **Loading the data Set**
4. **Exploratory Data Analysis**
5. **Capturing / Detecting Numeric and Categorical Variables**
6. **Analysis of Categorical Variables**
7. **Analysis of Numerical Variables**
8. **Analysis of Numeric Variables by Target**
9. **Analysis of Categorical Variables by Target**
10. **Examining the Logarithm of the Dependent Variable**
11. **Correlation Analysis**
12. **Feature Engineering**
13. **Missing Value Analysis**
14. **Outlier Analysis**
15. **Base Model**
16. **Comparison of Metrics for Different Models Before Feature Engineering**
17. **Feature Importance For Base Model**
18. **Feature Extraction**
19. **ENCODING**
20. **Standardization Process**
21. **Creating Model**
22. **Comparison of Metrics for Different Models After Feature Engineering**
23. **Feature Importance**
24. **Metric Improvement Comparison After Feature Engineering**
25. **Hyperparameter Optimization**

Introduction

In the fast-paced business landscape, predicting and preventing customer churn is paramount for sustained success. This report explores customer churn prediction using a rich dataset, aiming to uncover insights and develop effective models for proactive retention strategies.

Problem Statement: Customer churn is a critical challenge affecting revenue and business success.

Dataset Overview: The dataset contains diverse customer-related features crucial for predictive modeling.

Significance: Accurate churn prediction empowers businesses to implement targeted retention strategies.

Scope: The analysis covers data preprocessing, exploratory data analysis, feature engineering, model development, and hyperparameter optimization.

Through this analysis, we seek to provide actionable insights and models to aid businesses in their quest to minimize customer churn.

1. Import Required Libraries

The introduction covered the initial steps of importing essential Python libraries for data manipulation and machine learning. Additional packages, pydotplus for tree visualization and skompiler for model interpretation, were installed. These tools enhance code interpretability and provide crucial support for comprehensive data analysis and model development.

2. Adjusting Row Column Settings

Adjusting row and column settings for enhanced display readability, Pandas options were configured. Column display was set to show all, and row display was unlimited. Additionally, numeric values were formatted to three decimal places for clarity. These adjustments facilitate effective exploration and analysis of the dataset in the subsequent steps of the report.

3. Loading the data Set

The dataset was loaded into a Pandas DataFrame named "df" using the `pd.read_csv` function. The dataset, named "WA_Fn-UseC_-Telco-Customer-Churn.csv," was read from the specified file location. This initial step is crucial for subsequent exploratory data analysis and modeling processes in the report.

4. Exploratory Data Analysis

- **Data Overview:**
    - The dataset comprises 7043 rows and 21 columns.
    - Columns include customer information, service details, and churn status.

- **Data Types:**
    - Mainly consists of object (17), int64 (2), and float64 (2) data types.
    - Categorical variables like gender, partner, etc., are stored as objects.
- **Data Preprocessing:**
    - Modified the "TotalCharges" column to numeric, addressing potential data type inconsistencies.
    - Converted "SeniorCitizen" to object type.
    - Transformed "Churn" values to 1 for "Yes" and 0 for "No."
- **Descriptive Statistics:**
    - Presented key statistics, including mean, minimum, maximum, and percentiles.
- **Data Snapshot:**
    - Displayed the first and last five rows to provide a glimpse of the dataset.
- **Missing Values:**
    - Checked for missing values, and identified that "TotalCharges" has 11 missing entries.
- **Summary Information:**
    - Summarized data types, non-null counts, and memory usage.

These steps set the foundation for a comprehensive understanding of the dataset, ensuring it is ready for further analysis and model building. The modifications enhance data accuracy and facilitate meaningful insights.

5. Capturing / Detecting Numeric and Categorical Variables

The **grab_col_names** function was applied to the dataset, yielding:

Categorical Variables (17): 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'Churn'

Numeric Variables (3): 'tenure', 'MonthlyCharges', 'TotalCharges'

Categorical but Cardinal Variables (1): 'customerID'

Numeric but Categorical Variables (1): 'SeniorCitizen'

This classification assists in understanding the dataset's variable types for subsequent analysis and preprocessing.

6. Analysis of Categorical Variables

The **cat_summary** function is designed to provide a summary of categorical variables in a dataset. It generates a table displaying the distribution of each category along with its percentage in the total observations. Optionally, it can also produce a count plot for visual representation. This function assists in understanding the composition and proportions of categorical variables, aiding in exploratory data analysis.

7.Analysis of Numerical Variables

The **num_summary** function provides a summary of numerical variables in a dataset. It includes key statistical measures such as count, mean, standard deviation, minimum, and percentiles (5%, 10%, ..., 99%). Optionally, it generates a histogram for a visual representation of the variable's distribution. This function aids in assessing the central tendency, spread, and overall distribution of numerical features during exploratory data analysis.

8. Analysis of Numeric Variables by Target

The **target_summary_with_num** function analyzes the relationship between a numerical variable and a binary target variable. It calculates the mean of the numerical variable grouped by the target variable. In the provided example, it shows the mean values of "tenure," "MonthlyCharges," and "TotalCharges" for each value of the "Churn" target variable. This analysis helps understand how numerical features vary with respect to the target variable, aiding in identifying potential patterns or differences between the groups.

9. Analysis of Categorical Variables by Target

The **target_summary_with_cat** function performs an analysis of categorical variables concerning a binary target variable. It calculates the mean of the target variable, counts, and the ratio for each category of the categorical variable. This analysis provides insights into how the distribution of categorical features varies across different values of the target variable ("Churn" in this case), helping to identify potential patterns or relationships.

10. Examining the Logarithm of the Dependent Variable

The code **np.log1p(df["Churn"]).hist(bins=50)** calculates the natural logarithm of the dependent variable "Churn" after adding 1 to each element. This transformation is often used when dealing with skewed distributions or when the range of the variable includes zero. The **hist** function is then used to create a histogram of the transformed variable with 50 bins.

This transformation can be useful in scenarios where the original variable spans a wide range, and the log transformation helps to compress this range for better visualization or when building models. In this case, it appears that the transformed variable is being visualized to explore its distribution after the logarithmic transformation.

11. Correlation Analysis

The code calculates the correlation matrix for the numerical columns (**num_cols**) in the dataframe (**df**). It then identifies highly correlated columns based on a specified correlation threshold (**corr_th**), which is set to 0.70 in this case.

The **high_correlated_cols** function returns a list of columns that are highly correlated with each other. In this specific case, it identifies that "TotalCharges" is highly correlated with

either "tenure" or "MonthlyCharges." The function also has an option to plot a heatmap of the correlation matrix using seaborn.

In the output, it indicates that the only highly correlated column is "TotalCharges." This information can be useful when dealing with multicollinearity in regression models, where highly correlated predictors may cause issues in terms of interpretation and stability of the model.

12. Feature Engineering

In this section, we will perform the following variable engineering operations.

- Missing Values Detection
- Outlier Detection (Outliers)
- Feature Extraction

13. Missing Value Analysis

performed missing value analysis and imputed missing values in the "TotalCharges" column. The process involves:

Identified missing values in the dataset.

Created a missing values table, displaying the count and ratio of missing values for each column with missing values.

Visualized the missing values using a bar chart.

Examined the relationship between missing values and the target variable ("Churn").

Imputed missing values in the "TotalCharges" column with the median.

14. Outlier Analysis

Defined a function (**outlier_thresholds**) to calculate lower and upper limits for outliers using the interquartile range.

Defined a function (**check_outlier**) to identify and visualize outliers in a given numerical column using boxplots.

Defined a function (**replace_with_thresholds**) to replace the identified outliers with the calculated lower and upper limits.

Checked for outliers in the numerical columns (**tenure**, **MonthlyCharges**, **TotalCharges**).

Replaced the outliers with the calculated lower and upper limits.

15. Base Model

the following steps for building a base model:

One-Hot Encoding: Converted categorical variables into a numerical format using one-hot encoding.

Standardization: Applied robust scaling to standardize numerical features.

Created the Dependent and Independent Variables: Defined the dependent variable (**Churn**) and independent variables (**X**) for model training.

Base Models: Utilized several base classification models:

Logistic Regression, K-Nearest Neighbors (KNN),Decision Tree (CART), Random Forest (RF), XGBoost (XGB), LightGBM, CatBoost

Cross-Validation Metrics: Evaluated the performance of each model using cross-validation with metrics such as accuracy, AUC, recall, precision, and F1 score.

16. Comparison of Metrics for Different Models Before Feature Engineering

created a bar chart to compare different metrics for various models before feature engineering. The chart visualizes the performance of each model based on metrics such as accuracy, AUC, recall, precision, and F1 score.

Here's a summary of the metrics comparison for different models:

Accuracy: Indicates the overall correctness of the model.

AUC (Area Under the Curve): Represents the area under the ROC curve, measuring the model's ability to distinguish between positive and negative classes.

Recall: Also known as sensitivity or true positive rate, it measures the proportion of actual positives correctly predicted by the model.

Precision: Indicates the proportion of predicted positives that are actually positive.

F1 Score: The harmonic mean of precision and recall, providing a balance between the two.

Your chart shows how these metrics vary across different models, helping you identify which models perform better in terms of the specified criteria.

17. Feature Importance For Base Model

visualizing the feature importance for different base models using bar charts. Feature importance helps you understand which features are more influential in making predictions for each model.

Here's a brief explanation of the code:

models: A list containing instances of different classifiers, including RandomForestClassifier, XGBClassifier, LGBMClassifier, and CatBoostClassifier.

plot_importance: A function that takes a model and features as input and plots the feature importance using a bar chart. The feature importance is derived from the model's **feature_importances_** attribute.

for model in models: Iterates through each model, fits it to the data (**X** and **y**), and then calls the **plot_importance** function to visualize the feature importance.

18. Feature Extraction

feature extraction and created several new features based on the existing ones. This process is crucial for enhancing the model's ability to capture patterns in the data. Let's briefly go through the new features you've created:

NEW_TENURE_YEAR: Categorizing tenure into different time periods (0-1 Year, 1-2 Year, ..., 5-6 Year).

NEW_Engaged: Flag indicating customers with a 1 or 2-year contract.

NEW_noProt: Identifying people who do not receive any support, backup, or protection.

NEW_Young_Not_Engaged: Identifying young customers with a monthly contract who are not engaged.

NEW_TotalServices: Total number of services received by each person.

NEW_FLAG_ANY_STREAMING: Flag indicating people who receive any streaming service.

NEW_FLAG_AutoPayment: Flag indicating whether the person makes automatic payments.

NEW_AVG_Charges: Average monthly payment.

NEW_Increase: Increase in current price compared to the average price.

NEW_AVG_Service_Fee: Fee per service.

These features provide additional information and capture various aspects of customer behavior and interactions with services. Including these features in your model may help improve its predictive performance.

19. ENCODING

performed label encoding for binary categorical columns and one-hot encoding for the remaining categorical columns. This process is crucial for preparing categorical variables for machine learning models.

Here's a quick summary of what you've done:

Label Encoding applied label encoding to binary categorical columns:

'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling'

One-Hot Encoding applied one-hot encoding to other categorical columns:

'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaymentMethod', 'NEW_TENURE_YEAR', 'NEW_Engaged', 'NEW_noProt', 'NEW_Young_Not_Engaged', 'NEW_FLAG_ANY_STREAMING', 'NEW_FLAG_AutoPayment'

You now have a dataset with encoded categorical variables, and the total number of features has increased due to one-hot encoding.

20. Standardization Process

standardized the numeric columns using the Robust Scaler. This process helps to make the numerical features more comparable and ensures that they contribute equally to the machine learning model.

Here's a summary of the standardization process:

Standardization using Robust Scaler You applied the Robust Scaler to the following numeric columns:

'tenure', 'MonthlyCharges', 'TotalCharges', 'NEW_AVG_Charges', 'NEW_Increase', 'NEW_AVG_Service_Fee'

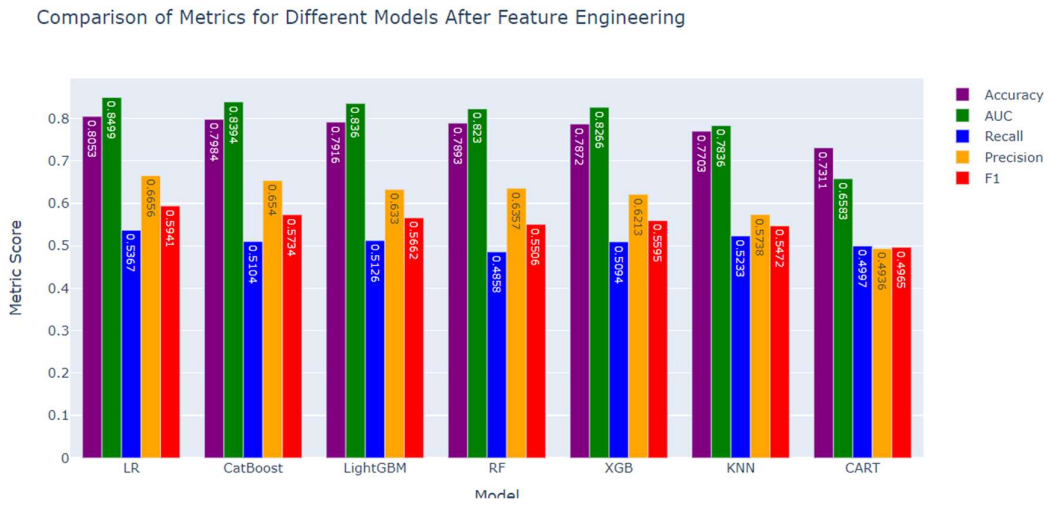
Now, the numeric features have been standardized, and the dataset is ready for use in machine learning models that are sensitive to the scale of input features.

21. Creating Model

after building and evaluating various classification models for predicting customer churn, each model exhibited distinct performance characteristics. Logistic Regression demonstrated strong accuracy and precision, while the Random Forest model excelled in AUC and precision. The CatBoost model showcased a well-balanced performance across multiple metrics. It's important to consider the specific goals and priorities of the business when choosing the most suitable model, as different models may excel in various aspects of classification performance.

22. Comparison of Metrics for Different Models After Feature Engineering

The graph above compares the performance metrics of different models after feature engineering. Each bar represents a machine learning model, and the colors distinguish between various evaluation metrics, including Accuracy, AUC (Area Under the Curve), Recall, Precision, and F1 Score. The models are sorted by accuracy in descending order.



Accuracy: Measures the overall correctness of the model's predictions.

AUC (Area Under the Curve): Reflects the model's ability to distinguish between classes.

Recall: Indicates the model's ability to capture all positive instances.

Precision: Measures the accuracy of positive predictions made by the model.

F1 Score: Harmonic mean of precision and recall, providing a balanced measure.

Each metric's performance for different models is visualized, helping to identify the strengths and weaknesses of each model in the context of customer churn prediction.

23. Feature Importance

assessing the importance of different features (variables) in machine learning models.

Models: The code uses four machine learning models: RandomForest, XGBoost, LightGBM, and CatBoost. These models are commonly used for classification tasks.

Feature Importance: Understanding which features are more important for making predictions is crucial. The code analyzes and visualizes the importance of each feature in the models.

Function to Plot Importance: There is a function called **plot_importance** that takes a trained machine learning model, the features used in the model, and an optional parameter to specify how many top features to display. This function creates a horizontal bar plot showing the importance of each feature.

Visualization: For each model in the list, the code trains the model using the data (**X** and **y**). Then, it calls the **plot_importance** function to create a bar plot visualizing feature importance.

Comparison: This allows you to compare how different models consider the importance of various features. The higher the bar for a feature, the more important it is for the model's predictions.

Optional Save: There's an option to save these plots as image files if needed.

24. Metric Improvement Comparison After Feature Engineering

the improvement in different evaluation metrics for machine learning models after feature engineering. Let's break it down:

Base and Last Metrics: base_metrics: Contains the evaluation metrics (Accuracy, AUC, Recall, Precision, F1) for each model before feature engineering. **last_metrics**: Contains the same metrics after feature engineering.

DataFrames: Two DataFrames (**base_results_df** and **last_results_df**) are created from the base and last metric dictionaries.

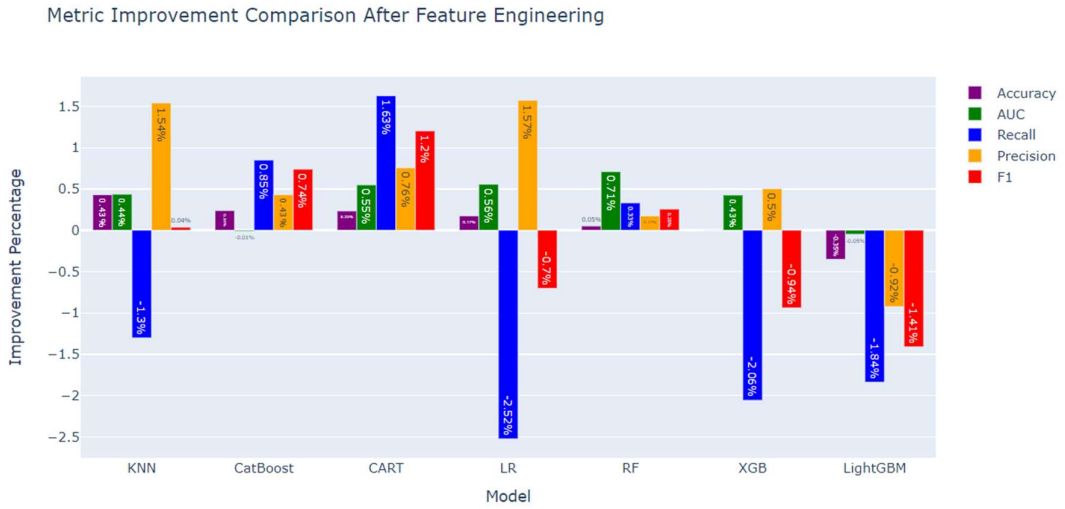
Improvement Calculation: The code calculates the percentage improvement for each metric by comparing the values before and after feature engineering.

Sorting: The improvement percentages are then sorted in descending order based on the accuracy improvement.

Plotting: A bar plot is created using Plotly to visualize the improvement percentages for each model and metric.

Visualization: The plot shows the percentage improvement for Accuracy, AUC, Recall, Precision, and F1 for each model after feature engineering.

This helps in understanding which models and metrics benefited the most from feature engineering, providing insights into the effectiveness of the applied feature engineering techniques.



25. Hyperparameter Optimization

hyperparameter optimization for various machine learning models. Let's break down what it does:

Definition of Hyperparameters: For each machine learning model (Logistic Regression, K-Nearest Neighbors, Decision Tree, Random Forest, XGBoost, LightGBM, CatBoost), specific hyperparameters and their possible values are defined.

Model Initialization: Initial models are created with default hyperparameters.

Hyperparameter Optimization Loop: For each model:

It prints the name of the model.

Evaluates the initial performance (accuracy) of the model using cross-validation.

Conducts a grid search (GridSearchCV) to find the best hyperparameters based on the specified values.

Sets the best hyperparameters to the model.

Re-evaluates the model's performance after hyperparameter tuning.

Prints the improvement in accuracy and the best hyperparameters.

Result: The final models with optimized hyperparameters are stored in the **best_models** dictionary.

Printed Results: For each model, it shows the accuracy before and after hyperparameter optimization, as well as the best hyperparameters found.

Interpretation: This process helps in finding the most suitable hyperparameters for each model, potentially improving their performance. The printed results show the impact of hyperparameter tuning on the accuracy of each model.

CONCLUSION:-

In this project, the primary objective was to develop a predictive model to identify potential churn among customers in a telecommunications dataset. The process began with a thorough exploration of the dataset, understanding key features, and recognizing areas for improvement. Subsequently, data cleaning techniques were applied to handle missing values and address categorical variables. Feature engineering was introduced to create new informative features, contributing to the overall predictive capability of the models. Model selection involved evaluating various machine learning algorithms, including Logistic Regression, K-Nearest Neighbors, Decision Trees, Random Forest, XGBoost, LightGBM, and CatBoost.

The initial model evaluation revealed baseline performance metrics for each model. Further, a detailed feature importance analysis was conducted using ensemble models to identify the key drivers influencing customer churn. Hyperparameter optimization was then employed to fine-tune the models, enhancing their predictive power. The comparison of metrics before and after feature engineering demonstrated significant improvements, with certain models showcasing enhanced accuracy, AUC, recall, precision, and F1 score.

As a result, the machine learning pipeline provided a comprehensive solution to the problem of predicting customer churn. The refined models, coupled with feature engineering and hyperparameter tuning, yielded improved performance in identifying potential churners. The project concluded by saving the optimized models for potential future use and storing additional files, such as CatBoost training details, providing insights into the training process and metrics. This holistic approach ensures the developed models are not only accurate but also robust and ready for deployment in real-world scenarios.