**Introduction to the Telco Customer Churn Dashboard**

**Overview**

The Telco Customer Churn Dashboard is an interactive web application designed to analyze and visualize data related to customer churn in a telecommunications company. Customer churn, the phenomenon where customers discontinue their service, is a critical metric for telecom companies as it directly impacts their revenue and growth. This dashboard provides valuable insights into various factors influencing churn, such as customer demographics, service usage, and billing information, allowing stakeholders to make data-driven decisions to reduce churn rates.

The Telco Customer Churn Dashboard is a user-friendly web tool that helps examine and display data about customers who decide to stop using a telecom company's services. Customer churn is a significant concern for telecom companies because when customers leave, it directly affects the company's income and growth. This dashboard makes it easy to understand why customers are leaving by looking at different aspects of their behavior and characteristics. It takes into account various details like customer demographics (age, gender, etc.), how they use the services, and their billing information, to give a clear picture of what might be causing them to churn. This way, the company can pinpoint problem areas and take action to improve customer retention.

**Data Exploration and Analysis (EDA)**

**Dataset Overview:**

* Source: Telco Customer Churn Dataset from Kaggle.
* Records: 7,043 customers.
* Features: 21 attributes, including customer demographics, account information, and service usage details.

**Initial Findings:**

* Churn Rate: The dataset shows that approximately 26.5% of the customers churned.
* Numerical Features: Key numerical features include tenure, MonthlyCharges, and TotalCharges.
* Categorical Features: Include gender, SeniorCitizen, Partner, Dependents, and various service-related features.

Exploratory Data Analysis (EDA) Insights:

* Tenure: Customers with lower tenure are more likely to churn.
* MonthlyCharges: Higher monthly charges are associated with higher churn rates.
* Contract Type: Customers with month-to-month contracts exhibit higher churn rates compared to those with longer contracts.
* Service Usage: Customers who do not use certain add-on services (like online security or tech support) tend to have higher churn rates.

2. Feature Engineering

* Feature engineering was performed to enhance the predictive power of the model:
* Handling Missing Values: The TotalCharges feature had missing values which were removed after converting the column to numeric.
* Categorical Encoding: Categorical features were encoded using one-hot encoding for model compatibility.
* Derived Features: Additional features such as tenure\_per\_month, which represents tenure divided by monthly charges, were considered but ultimately not used due to lack of significant impact.

**Model Development**

The model development process for predicting customer churn focused on selecting and fine-tuning a logistic regression model. This involved optimizing the model’s hyperparameters to achieve the best performance in identifying whether a customer is likely to churn or not.

Logistic Regression Model

Model Choice: Logistic Regression was chosen for this task due to its simplicity and interpretability. It is highly effective for binary classification problems, where the goal is to predict one of two possible outcomes—in this case, whether a customer will churn (leave the service) or not.

Hyperparameters: Hyperparameters are settings that influence the learning process of a machine learning model. Unlike model parameters, which are learned from the training data, hyperparameters are set before the learning process begins. In this project, the hyperparameters of the logistic regression model were optimized using Optuna, a powerful framework for hyperparameter optimization. This helped in systematically searching for the best set of hyperparameters to improve the model’s performance.

Hyperparameters Tuned

- Penalty: This refers to the type of regularization used to prevent overfitting by adding a penalty term to the loss function. The regularization can be:

- l1: Lasso regularization, which can result in sparse models by forcing some coefficient estimates to be exactly zero.

- l2: Ridge regularization, which shrinks the coefficient estimates but keeps them all non-zero.

- None: No regularization.

- C: This is the inverse of the regularization strength. It controls the trade-off between achieving a good fit to the training data and avoiding overfitting. The value of \( C \) was explored in the range from 0.1 to 100. A smaller \( C \) value corresponds to stronger regularization.

- Solver: The algorithm used for optimization. Different solvers are better suited for different types of data and regularization. Options include:

- liblinear: Good for small datasets and supports both l1 and l2 regularization.

- saga: Suitable for large datasets and supports elastic net regularization.

- lbfgs: An optimization algorithm that approximates the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, ideal for relatively smaller datasets and l2 regularization.

- Threshold: This is the decision boundary for classifying a customer as churned or not churned. Logistic regression outputs probabilities, and the threshold determines at what probability level a customer is classified as churned. It was explored in the range from 0 to 1, where a threshold of 0.5 is commonly used as the default for binary classification.

Definition: Hyperparameters are predefined parameters that guide the training process of a machine learning model, affecting its performance and how well it generalizes to new data. They differ from model parameters, which are learned from the data during training.

**Model Evaluation**

After developing and fine-tuning the logistic regression model for predicting customer churn, it was crucial to assess its performance using standard metrics for binary classification. These metrics provide insights into how well the model is performing in terms of both correctly identifying churned customers and minimizing false predictions.

**Metrics Used**

1. Accuracy: Accuracy measures the proportion of correctly classified instances out of the total instances. It provides a general idea of the model's correctness but may not be sufficient when classes are imbalanced.

2. Precision: Precision measures the accuracy of positive predictions made by the model. It calculates the proportion of true positive predictions (correctly identified churned customers) among all positive predictions (instances predicted as churned). Precision is essential when the cost of false positives is high.

3. Recall: Recall, also known as sensitivity or true positive rate, measures the model's ability to capture all positive instances. It calculates the proportion of true positive predictions among all actual positive instances (churned customers). Recall is crucial when missing actual positives (churned customers) is costly.

4. F1-Score: The F1-Score is the harmonic means of precision and recall. It provides a balance between precision and recall and is particularly useful when there is an uneven class distribution or when false positives and false negatives have different costs.

Detailed Explanation

- Accuracy:

- Formula: \( \text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \)

- Accuracy provides an overall measure of the model's correctness. However, it might not be reliable when classes are imbalanced, as high accuracy could be achieved by simply predicting the majority class.

- Precision:

- Formula: \( \text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}} \)

- Precision focuses on the accuracy of positive predictions. It answers the question: "Of all the instances predicted as churned, how many are actually churned?" High precision indicates that the model has a low rate of false positives.

- Recall:

- Formula: \( \text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}} \)

- Recall measures the model's ability to capture all positive instances. It answers the question: "Of all the actual churned customers, how many did the model correctly identify?" High recall indicates that the model effectively identifies churned customers.

- F1-Score:

- Formula: \( \text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}} \)

- The F1-Score is the harmonic mean of precision and recall. It provides a balanced measure that considers both false positives and false negatives. A high F1-Score indicates that the model has both good precision and recall.

Results:

Accuracy: The model achieved an accuracy of approximately 79%, indicating good overall performance.

Precision: A precision score of 71% indicates a reasonable balance between true and false positives.

Recall: A recall score of 62% suggests that the model effectively identifies most churn cases.

F1-Score: An F1-score of 66% reflects a balanced performance between precision and recall

**Dashboard Implementation**

The dashboard was developed using Dash and Plotly Express to provide an interactive and user-friendly interface for exploring churn data.

**Key Components:**

* Feature Selection Dropdowns: Allow users to select numerical and categorical features to visualize distributions and relationships with churn.
* Histograms and Box Plots: Visualize distributions of key features and their association with churn.
* Facet Histograms: Show how contract type and tenure affect churn rates, highlighting the impact of service agreements on customer retention.
* Functionalities:
* Users can explore how different factors contribute to churn by selecting features from dropdowns.
* Visualizations update dynamically, providing insights into the behavior and characteristics of churned customers.

6. Challenges Faced

* Data Quality Issues:
* Missing Values: The TotalCharges column had missing values that needed to be addressed.
* Class Imbalance: The dataset had an imbalance between churned and non-churned customers, which could bias the model.
* Feature Engineering:
* Identifying and creating meaningful derived features was challenging, as many did not significantly improve model performance.
* Hyperparameter tuning was time-consuming, requiring extensive experimentation to identify optimal settings for the logistic regression model.
* Interactivity and Performance:
* Ensuring the dashboard remained responsive while handling large datasets and multiple visualizations was a challenge, necessitating performance optimization.

**MY CODE :-**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import shap

import optuna

optuna.logging.disable\_default\_handler()

# Load the dataset

df = pd.read\_csv('WA\_Fn-UseC\_-Telco-Customer-Churn.csv')

# Data Overview

print(df.head(10))

print(f"Shape: {df.shape}")

print(f"Size: {df.size}")

print(f"Columns: {df.columns.tolist()}")

print(f"Null Values: \n{df.isnull().sum()}")

# Check if all customer IDs are unique

print(f"All customer IDs are unique: {df['customerID'].nunique() == df.shape[0]}")

# Drop the 'customerID' column as it's not needed for analysis

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

# Convert 'TotalCharges' to numeric, coerce errors

df['TotalCharges'] = pd.to\_numeric(df['TotalCharges'], errors='coerce')

# Check data types and summary statistics

num\_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']

cat\_cols = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',

            'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',

            'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod']

print(df[num\_cols].describe())

# Plotting

plt.figure(figsize=(24, 8))

sns.set(style='whitegrid', palette='deep')

# Tenure Histogram

plt.subplot(1, 3, 1)

sns.histplot(data=df, x='tenure', bins=30, kde=True, label='Total')

sns.histplot(data=df[df['Churn'] == 'Yes'], x='tenure', bins=30, kde=True, label='Churn')

plt.legend()

plt.title('Tenure Distribution')

# Monthly Charges Histogram

plt.subplot(1, 3, 2)

sns.histplot(data=df, x='MonthlyCharges', bins=30, kde=True, label='Total')

sns.histplot(data=df[df['Churn'] == 'Yes'], x='MonthlyCharges', bins=30, kde=True, label='Churn')

plt.legend()

plt.title('Monthly Charges Distribution')

# Total Charges Histogram

plt.subplot(1, 3, 3)

sns.histplot(data=df, x='TotalCharges', bins=30, kde=True, label='Total')

sns.histplot(data=df[df['Churn'] == 'Yes'], x='TotalCharges', bins=30, kde=True, label='Churn')

plt.legend()

plt.title('Total Charges Distribution')

plt.show()

# Boxplots for Churn vs Numerical Variables

plt.figure(figsize=(24, 8))

sns.set(style='whitegrid')

plt.subplot(1, 3, 1)

sns.boxplot(data=df, x='Churn', y='tenure')

plt.title('Churn vs Tenure')

plt.subplot(1, 3, 2)

sns.boxplot(data=df, x='Churn', y='MonthlyCharges')

plt.title('Churn vs Monthly Charges')

plt.subplot(1, 3, 3)

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

plt.title('Churn vs Total Charges')

plt.show()

# Count Plot with Percentages

def annotate\_percent(ax, total):

    for p in ax.patches:

        percentage = '{:.1f}%'.format(100 \* p.get\_height() / total)

        x = p.get\_x() + p.get\_width() / 2

        y = p.get\_height() / 2

        ax.annotate(percentage, (x, y), ha='center', fontsize=15, weight='bold')

plt.figure(figsize=(8, 8))

sns.set(style='whitegrid')

ax = sns.countplot(data=df, x='Churn')

annotate\_percent(ax, df.shape[0])

plt.title('Churn Distribution')

plt.show()

# Count plots for categorical variables

plt.figure(figsize=(32, 8))

sns.set(style='whitegrid')

plt.subplot(1, 4, 1)

ax1 = sns.countplot(data=df, x='gender')

annotate\_percent(ax1, df.shape[0])

plt.title('Gender Distribution')

plt.subplot(1, 4, 2)

ax2 = sns.countplot(data=df, x='SeniorCitizen')

annotate\_percent(ax2, df.shape[0])

plt.title('Senior Citizen Distribution')

plt.subplot(1, 4, 3)

ax3 = sns.countplot(data=df, x='Partner')

annotate\_percent(ax3, df.shape[0])

plt.title('Partner Distribution')

plt.subplot(1, 4, 4)

ax4 = sns.countplot(data=df, x='Dependents')

annotate\_percent(ax4, df.shape[0])

plt.title('Dependents Distribution')

plt.show()

# Count plots for Contract, Paperless Billing, Payment Method

plt.figure(figsize=(24, 8))

sns.set(style='whitegrid')

plt.subplot(1, 3, 1)

ax1 = sns.countplot(data=df, x='Contract')

annotate\_percent(ax1, df.shape[0])

plt.title('Contract Type Distribution')

plt.subplot(1, 3, 2)

ax2 = sns.countplot(data=df, x='PaperlessBilling')

annotate\_percent(ax2, df.shape[0])

plt.title('Paperless Billing Distribution')

plt.subplot(1, 3, 3)

ax3 = sns.countplot(data=df, x='PaymentMethod')

annotate\_percent(ax3, df.shape[0])

plt.title('Payment Method Distribution')

plt.show()

# Count plots for Internet Services

plt.figure(figsize=(32, 16))

sns.set(style='whitegrid')

internet\_services = ['InternetService', 'OnlineSecurity', 'OnlineBackup',

                     'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

for idx, service in enumerate(internet\_services):

    plt.subplot(2, 4, idx + 1)

    ax = sns.countplot(data=df, x=service)

    annotate\_percent(ax, df.shape[0])

    plt.title(f'{service} Distribution')

plt.show()

# Count plots for Phone Services

plt.figure(figsize=(16, 8))

sns.set(style='whitegrid')

phone\_services = ['PhoneService', 'MultipleLines']

for idx, service in enumerate(phone\_services):

    plt.subplot(1, 2, idx + 1)

    ax = sns.countplot(data=df, x=service)

    annotate\_percent(ax, df.shape[0])

    plt.title(f'{service} Distribution')

plt.show()

# Histogram for Contract Types vs Tenure

plt.figure(figsize=(24, 8))

sns.set(style='whitegrid')

plt.subplot(1, 3, 1)

ax1 = sns.histplot(data=df[df['Contract'] == 'Month-to-month'], x='tenure', hue='Churn', bins=30, kde=True)

plt.title('Tenure for Month-to-Month Contracts')

plt.subplot(1, 3, 2)

ax2 = sns.histplot(data=df[df['Contract'] == 'One year'], x='tenure', hue='Churn', bins=30, kde=True)

plt.title('Tenure for One-Year Contracts')

plt.subplot(1, 3, 3)

ax3 = sns.histplot(data=df[df['Contract'] == 'Two year'], x='tenure', hue='Churn', bins=30, kde=True)

plt.title('Tenure for Two-Year Contracts')

plt.show()

# Encoding 'InternetService' for further analysis

label\_encode = {'No': 0, 'DSL': 1, 'Fiber optic': 2}

df['InternetService'] = df['InternetService'].map(label\_encode)

print(df['InternetService'].value\_counts())

**MY DASBOARD:-**

import pandas as pd

import dash

from dash import dcc, html, Input, Output

import dash\_bootstrap\_components as dbc

import plotly.express as px

df = pd.read\_csv('WA\_Fn-UseC\_-Telco-Customer-Churn.csv')

df['TotalCharges'] = pd.to\_numeric(df['TotalCharges'], errors='coerce')

df = df.dropna(subset=['TotalCharges'])

app = dash.Dash(\_\_name\_\_, external\_stylesheets=[dbc.themes.BOOTSTRAP])

app.layout = dbc.Container([

    dbc.Row([

        dbc.Col(html.H1("Telco Customer Churn Dashboard", className="text-center mb-4"), width=12)

    ]),

    dbc.Row([

        dbc.Col([

            dcc.Dropdown(

                id='feature-dropdown',

                options=[{'label': col, 'value': col} for col in ['tenure', 'MonthlyCharges', 'TotalCharges']],

                value='tenure',

                multi=False,

                style={'width': "100%"}

            ),

            dcc.Graph(id='histogram')

        ], width=6),

        dbc.Col([

            dcc.Dropdown(

                id='cat-feature-dropdown',

                options=[{'label': col, 'value': col} for col in ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService',

                                                                  'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',

                                                                  'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',

                                                                  'Contract', 'PaperlessBilling', 'PaymentMethod']],

                value='gender',

                multi=False,

                style={'width': "100%"}

            ),

            dcc.Graph(id='countplot')

        ], width=6)

    ]),

    dbc.Row([

        dbc.Col([

            dcc.Graph(id='boxplot-tenure', config={'displayModeBar': False})

        ], width=4),

        dbc.Col([

            dcc.Graph(id='boxplot-monthly', config={'displayModeBar': False})

        ], width=4),

        dbc.Col([

            dcc.Graph(id='boxplot-total', config={'displayModeBar': False})

        ], width=4)

    ]),

    dbc.Row([

        dbc.Col([

            dcc.Graph(id='contract-tenure-churn', config={'displayModeBar': False})

        ], width=12)

    ])

], fluid=True)

@app.callback(

    Output('histogram', 'figure'),

    Input('feature-dropdown', 'value')

)

def update\_histogram(feature):

    fig = px.histogram(df, x=feature, color='Churn', barmode='overlay', nbins=30, title=f'Distribution of {feature}')

    fig.update\_layout(xaxis\_title=feature, yaxis\_title='Count')

    return fig

@app.callback(

    Output('countplot', 'figure'),

    Input('cat-feature-dropdown', 'value')

)

def update\_countplot(feature):

    fig = px.histogram(df, x=feature, color='Churn', barmode='group', title=f'Churn Rate by {feature}')

    fig.update\_layout(xaxis\_title=feature, yaxis\_title='Count')

    return fig

@app.callback(

    [Output('boxplot-tenure', 'figure'),

     Output('boxplot-monthly', 'figure'),

     Output('boxplot-total', 'figure')],

    Input('cat-feature-dropdown', 'value')

)

def update\_boxplots(\_):

    fig\_tenure = px.box(df, x='Churn', y='tenure', title='Tenure Distribution by Churn')

    fig\_tenure.update\_layout(xaxis\_title='Churn', yaxis\_title='Tenure')

    fig\_monthly = px.box(df, x='Churn', y='MonthlyCharges', title='Monthly Charges by Churn')

    fig\_monthly.update\_layout(xaxis\_title='Churn', yaxis\_title='Monthly Charges')

    fig\_total = px.box(df, x='Churn', y='TotalCharges', title='Total Charges by Churn')

    fig\_total.update\_layout(xaxis\_title='Churn', yaxis\_title='Total Charges')

    return fig\_tenure, fig\_monthly, fig\_total

@app.callback(

    Output('contract-tenure-churn', 'figure'),

    Input('cat-feature-dropdown', 'value')

)

def update\_contract\_tenure\_churn(\_):

    fig = px.histogram(df, x='tenure', color='Churn', facet\_col='Contract', barmode='overlay', nbins=30,

                       title='Tenure Distribution by Contract Type and Churn')

    fig.update\_layout(xaxis\_title='Tenure', yaxis\_title='Count')

    return fig

if \_\_name\_\_ == '\_\_main\_\_':

    app.run\_server(debug=True)

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**