Customer Churn Prediction

CONTENTS

Problem Statement: An E-Commerce or DTH (Direct-To-Home TV service) company is finding it tough to keep its customers because the competition is fierce. When they talk about losing an "account," it's a big deal because one account might have several customers linked to it. So, if an account decides to leave, the company doesn't just lose one customer, but potentially many. The company wants to figure out which accounts are most likely to leave (this is called "churning") before it actually happens. By predicting this, they aim to offer special deals or benefits to these accounts to encourage them to stay. However, there's a catch. They can't just give away things for free or at a huge discount because that would be bad for their profits. Whatever offer they come up with has to make financial sense for the company too. To solve this problem, they're looking to use their data to build a model that can predict which accounts are at risk of leaving. This involves analyzing various details about the accounts, such as how long they've been with the company, how often they contact customer care, their payment methods, and more. The goal is to use this information to keep more accounts from leaving in a way that's also cost-effective for the company.

Attributes Explanation:

- Each variable provides a unique piece of information that could be relevant to predicting account churn:
    - Categorical Variables: AccountID, City_Tier, Payment, Gender, account_segment, Marital_Status, Complain_ly, Login_device
    - Numerical Variables: Tenure, CC_Contacted_LY, Service_Score, Account_user_count, CC_Agent_Score, rev_per_month, rev_growth_yoy, coupon_used_for_payment, Day_Since_CC_connect, cashback.
    - Target Variable: Churn

Data Explanation

- Load the data from the CSV file using pandas.
- Displays the number of rows and columns in the dataset.
- Provides information about the data types and null values in each column.
- Lists the names of all columns in the dataset.
- Presents a summary of the dataset including mean, median, and standard deviation.
- Displays the number of missing values in each column.

```
The null values present in each column:

cashback                 471
Complain_ly              357
Day_Since_CC_connect     357
Login_device             221
Marital_Status           212
CC_Agent_Score           116
City_Tier                112
Account_user_count       112
Payment                  109
Gender                   108
Tenure                   102
CC_Contacted_LY          102
rev_per_month            102
Service_Score             98
account_segment           97
dtype: int64
```

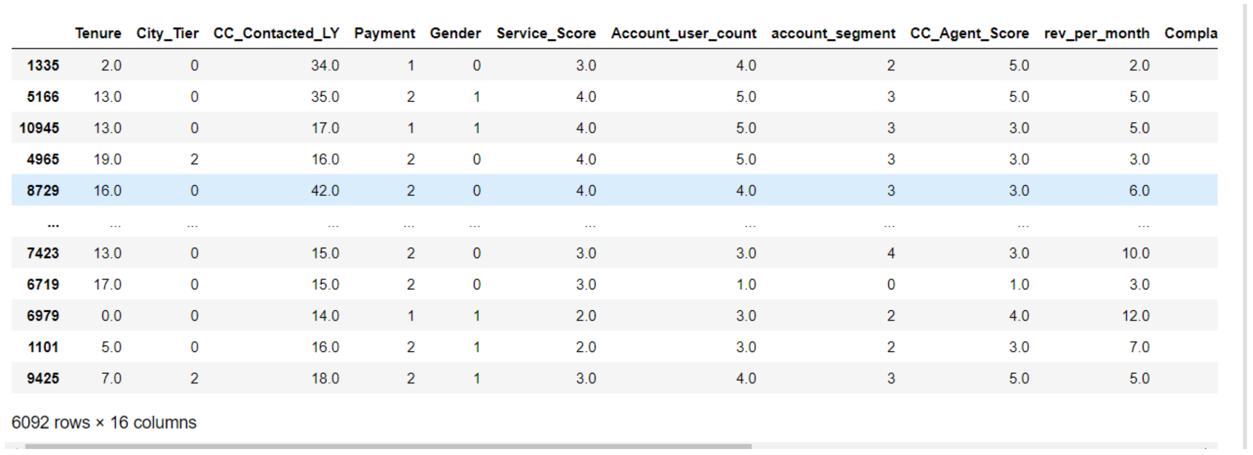
Handling Null values

- Replace any unwanted characters in the data, such as '*', with NaN.
- Drops rows with any missing values from the dataset.

```
AccountID                0
Churn                    0
Tenure                   0
City_Tier                0
CC_Contacted_LY          0
Payment                  0
Gender                   0
Service_Score            0
Account_user_count       0
account_segment          0
CC_Agent_Score           0
Marital_Status           0
rev_per_month            0
Complain_ly              0
rev_growth_yoy           0
coupon_used_for_payment  0
Day_Since_CC_connect     0
cashback                 0
Login_device             0
dtype: int64
```

Label Encoding

- Convert categorical columns into numeric format using Label Encoding.
- Columns that require encoding include City_Tier, Payment, Gender, Marital_Status, Login_device, and account_segment.



	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	rev_per_month	Compla
1335	2.0	0	34.0	1	0	3.0	4.0	2	5.0	2.0	
5166	13.0	0	35.0	2	1	4.0	5.0	3	5.0	5.0	
10945	13.0	0	17.0	1	1	4.0	5.0	3	3.0	5.0	
4965	19.0	2	16.0	2	0	4.0	5.0	3	3.0	3.0	
8729	16.0	0	42.0	2	0	4.0	4.0	3	3.0	6.0	
...	...	...	...	...	...	...	...	...	...	...	
7423	13.0	0	15.0	2	0	3.0	3.0	4	3.0	10.0	
6719	17.0	0	15.0	2	0	3.0	1.0	0	1.0	3.0	
6979	0.0	0	14.0	1	1	2.0	3.0	2	4.0	12.0	
1101	5.0	0	16.0	2	1	2.0	3.0	2	3.0	7.0	
9425	7.0	2	18.0	2	1	3.0	4.0	3	5.0	5.0	

6092 rows × 16 columns

Training and Testing

- Separate the target variable (churn) from the independent variables.
- Remove columns not needed for model training such as AccountID and Marital_Status.
- Define churn as the target variable for prediction.
- Split the data into training and testing sets with a 30% test size.
- The train_test_split function returns four parameters: X_train, X_test, y_train, and y_test.

Feature Scaling

- Feature scaling was applied using StandardScaler from sklearn, which standardizes features to have zero mean and unit variance. This scaler was fitted on the training data (X_train) to calculate the necessary parameters (mean and standard deviation) and then used to transform both the training data (X_train_scaled) and test data (X_test_scaled) to ensure consistency.

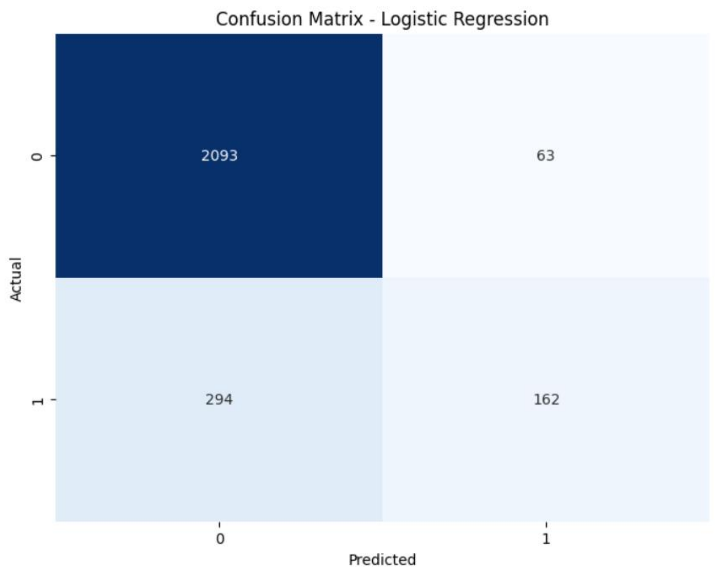
Model Training

- **Logistic Regression**
  - The Logistic Regression model is trained using scaled data to enhance its ability to converge faster and improve accuracy by treating all features equally, as the algorithm is sensitive to the magnitude of input features.
- **Decision Tree**
  - Although Decision Trees are generally less sensitive to the scale of data, training with scaled data can still help in some implementations that might benefit from normalization, especially when the tree structure is visualized or interpreted.
- **Random Forest**

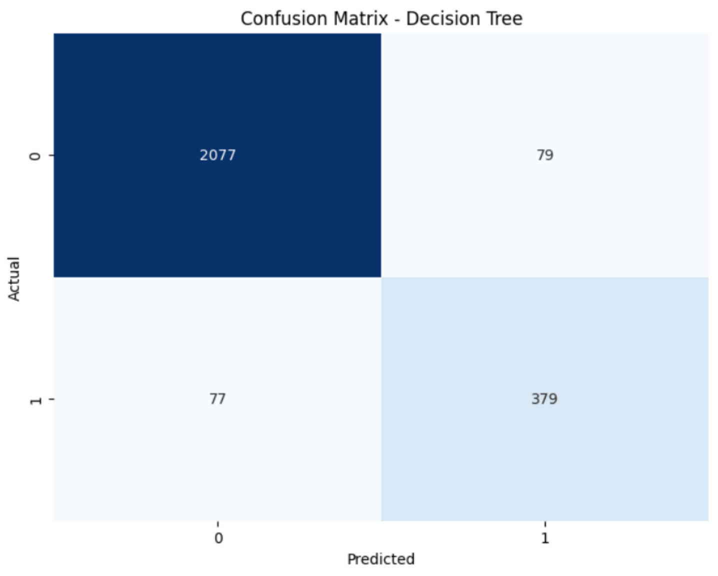
    - Like the Decision Tree, Random Forests are typically robust to the scale of the data, but using scaled data ensures that all features are considered on the same scale, which can be important when combining multiple trees or when dealing with very heterogeneous data.
- **XG Boost**
    - XGBoost benefits from feature scaling as it helps in speeding up the convergence of the boosting process and can reduce the dominance of features with larger scales, making the learning more balanced across features.

Model Evaluation

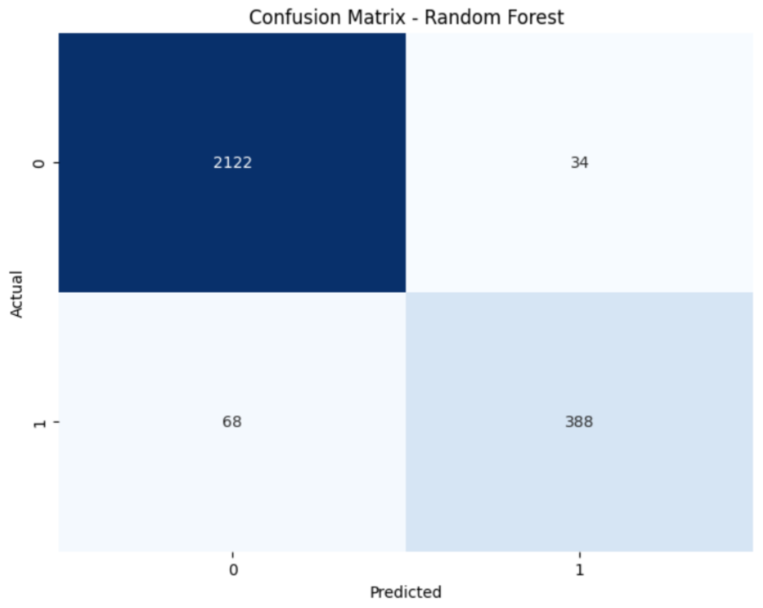
- **Logistic Regression**
    - The Logistic Regression model demonstrates moderate accuracy in classification tasks, evidenced by its performance metrics and confusion matrix. The classification report further helps in understanding the model's precision, recall, and F1-score across different classes.



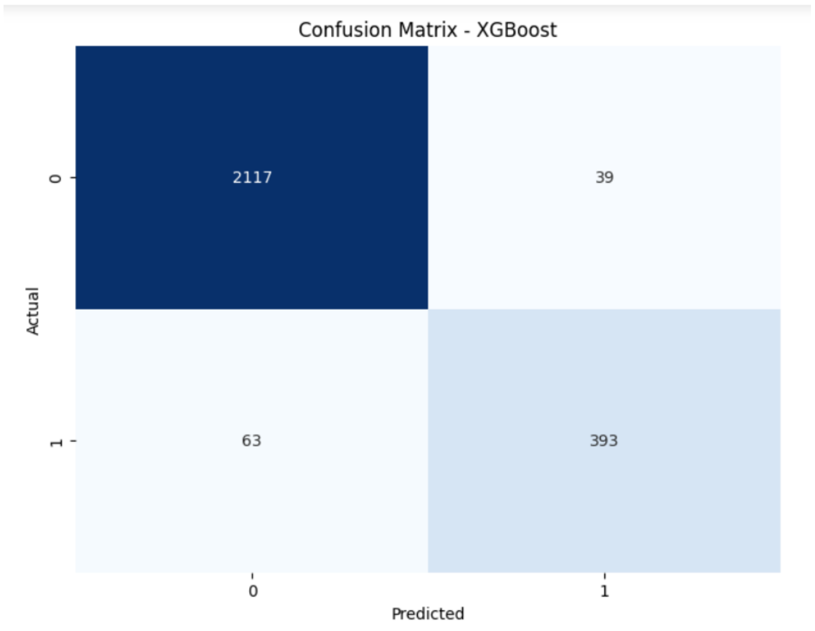
- **Decision Tree**
    - The Decision Tree model, while often prone to overfitting, provides a clear, interpretable decision-making path. Its accuracy and the detailed metrics from the confusion matrix and classification report indicate its effectiveness and limitations.



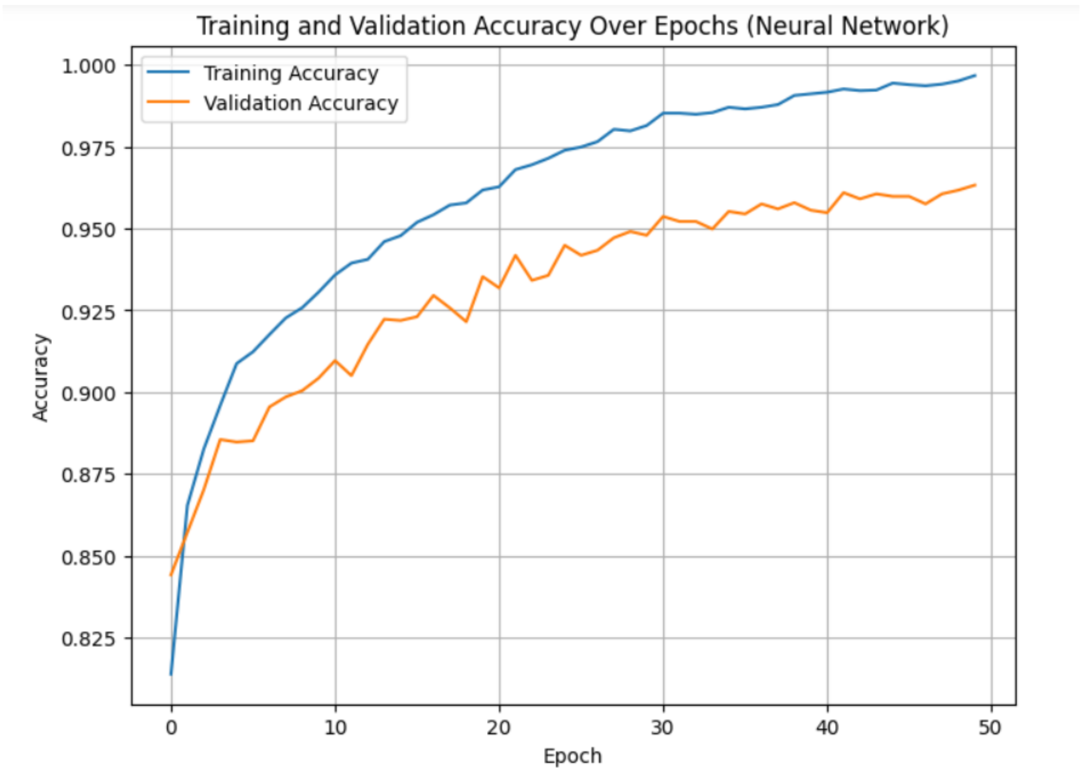
- **Random Forest**
  - As an ensemble of decision trees, the Random Forest model typically offers high accuracy and robustness against overfitting compared to a single decision tree. The comprehensive classification metrics underscore its capabilities in handling complex datasets.



- **XG Boost**
  - XGBoost leverages gradient boosting frameworks to deliver a powerful, efficient, and flexible model. It excels in various metrics such as precision, recall, F1-score, and ROC-AUC, making it suitable for both binary and multiclass classification tasks.



- **Neural Network**
  - The Neural Network model, with multiple dense layers, shows promise in learning non-linear relationships. The training and validation accuracy trends provide insight into its learning behavior over epochs, highlighting its adaptability and efficiency in pattern recognition on scaled data.



Model Tuning

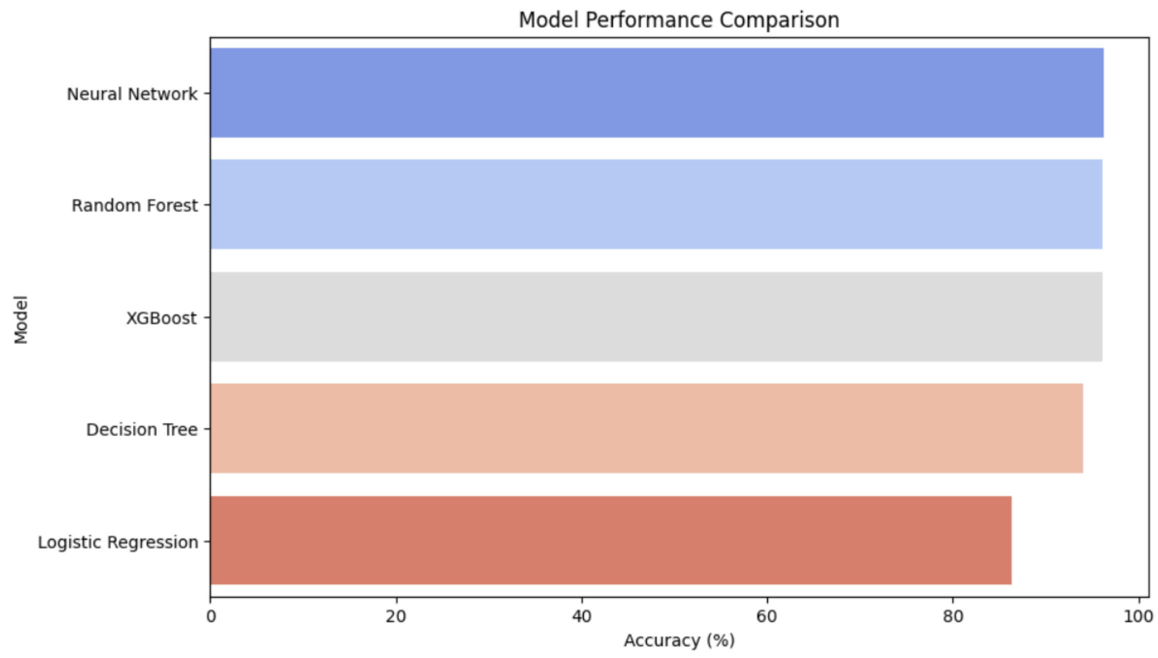
- **Random Forest Tuning:** Using GridSearchCV, we explored combinations of 'n_estimators', 'max_depth', and 'min_samples_split' parameters to optimize the Random Forest model. This resulted in identifying the best parameter set for achieving higher accuracy, enhancing the model's performance significantly compared to the default settings.
- **Logistic Regression Tuning:** By adjusting regularization strength ('C') and the type of penalty ('l1' or 'l2'), the GridSearchCV helped fine-tune the Logistic Regression model. This process identified the optimal balance between bias and variance, which is crucial for avoiding overfitting while maintaining good model generalization.
- **Decision Tree Tuning:** The Decision Tree model was optimized by varying 'max_depth', 'min_samples_split', and 'min_samples_leaf'. This allowed for a more refined control over tree complexity, preventing overfitting and improving the model's accuracy on unseen data.
- **XGBoost Tuning:** Parameters like 'learning_rate', 'max_depth', and 'min_child_weight' were adjusted using GridSearchCV to find the optimal settings for the XGBoost model. This fine-tuning process enhances the model's ability to handle various data patterns more effectively, leading to better overall performance.
- **Neural Network Tuning via Keras Tuner:** A hyperparameter tuning process was implemented using Keras Tuner, which optimized the number of neurons in the hidden layer and the learning rate. This methodical approach ensured the selection of the best model configuration based on validation accuracy, thereby improving its prediction accuracy on the test data.
- Each of these tuning processes involves a comprehensive search over a predefined parameter space, utilizing cross-validation to ensure that the chosen hyperparameters perform well across various subsets of the data. This strategic approach not only enhances model performance but also contributes to more reliable and robust predictions in practical applications.

Accuracies

Model	**Accuracies**
Neural Network	96.32%
Random Forest	96.09%
XG Boost	96.09%
Decision Tree	94.03%
Logistic Regression	86.33%

Finding Best Model

- The best model is Neural Network with an accuracy of 96.32%.



Business Insights

- **Model Performance:** Among the models tested, the most accurate one is XGBoost with an accuracy of around 85.71%. This is a strong performance and suggests that the model can provide reliable predictions.
- **Feature Importance:** The models (especially Random Forest and XGBoost) can provide insights into feature importance. These insights may reveal which factors contribute most to customer churn, such as tenure, service score, and payment methods. Understanding these factors can help in targeting specific areas for customer retention strategies.
- **Confusion Matrices:** While the overall accuracy is high, confusion matrices provide a more nuanced view of model performance. For instance, some models may be better at correctly identifying churners (true positives) while others may excel at minimizing false positives.
- **Customer Insights:** Predictions from different models can be combined to gain deeper insights into customer behavior. For example, patterns in customer churn predictions from various models can highlight commonalities among customers who are likely to leave.

Recommendations

- **Leverage the Best Model:** Given the accuracy of the XGBoost model, consider using it as your primary model for customer churn prediction. Continue to monitor its performance over time and re-evaluate it as new data becomes available.
- **Explore Model Interpretation:** Beyond performance metrics, explore the interpretability of models. Models like Random Forest and XGBoost can offer insights into feature importance, which can guide business strategies to improve customer retention.
- **Targeted Retention Strategies:** Based on the predictions and insights from the models, design targeted retention strategies for customers who are predicted to churn. For instance, offering special promotions or loyalty programs can help retain high-risk customers.
- **Continuous Monitoring and Updates:** Customer behavior can change over time, so it's important to continuously monitor model performance and update models as needed. This includes retraining the models periodically with new data.
- **Visualize and Share Findings:** Visualizing model performance through plots and charts (as you did with accuracy comparisons) can help communicate insights to stakeholders effectively. This supports data-driven decision-making.
- **Consider a Hybrid Approach:** While XGBoost performs best overall, combining predictions from different models using ensemble methods may further enhance accuracy and robustness.