Telecom Churn Prediction

By Ritika Khadilkar

Agenda

About telecom churn prediction

Our Mission and vision

Our Goals

Our Milestones

Challenges

Data Cleaning & Preprocessing

ML Model Building

Hyperparameter Tuning

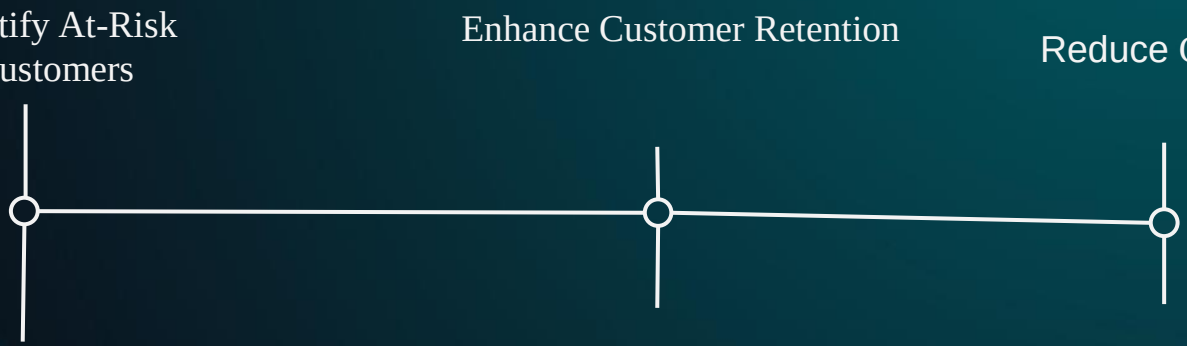
Logistic & Confusion Matrix

Conclusion

Telecom Churn Prediction

Telecom churn prediction involves using data analytics and machine learning to forecast which customers are likely to leave a telecommunications service provider. By analyzing customer behavior and demographics, companies can identify at-risk customers and implement strategies to retain them, thereby reducing turnover and improving customer satisfaction.

Goals



Identify At-Risk Customers

Accurately predict which customers are likely to leave the telecommunications service.

Enhance Customer Retention

Develop and implement targeted strategies to retain these at-risk customers.

Reduce Costs

Lower the expenses associated with acquiring new customers by focusing on retaining existing ones

Our Vision And Mission

Mission

We aim to use advanced data analytics and machine learning to predict when customers might leave our service. By doing so, we can implement personalized strategies to keep them happy, reduce the number of customers who switch to other providers, and operate more efficiently overall

Vision

Create a telecommunications environment focused on customers, using predictive insights to anticipate customer needs. This approach will help us keep customers loyal and ensure our business continues to grow

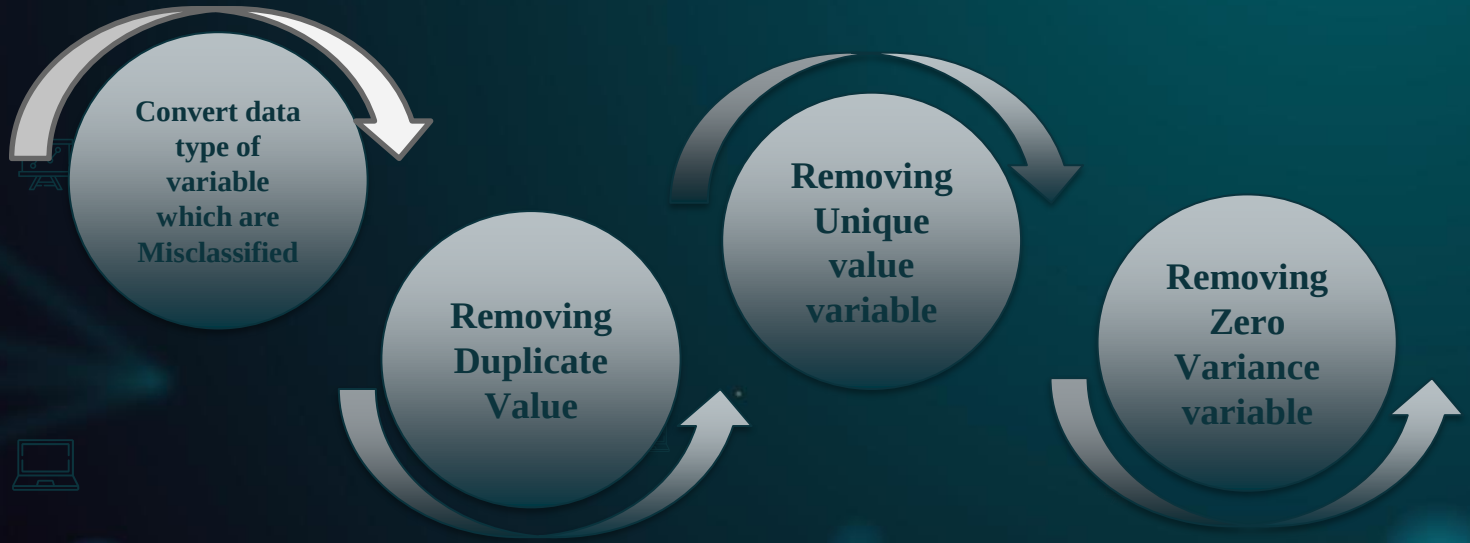
Millstones

Data gathering & understanding the dataset	Collect and comprehend the dataset to be used.
Data Cleaning and Pre-processing	Clean and prepare the data for analysis.
Machine Learning model building	Develop and train machine learning models.
Hyperparameter Tuning& Confusion Matrix	Optimize model parameters and evaluate performance using Confusion Matrix.

Challenges

- **Data Quality and Integration**: Integrating and maintaining high-quality data from diverse sources like customer demographics, call logs, and billing information is complex but essential for accurate predictions.
- **Imbalanced Datasets**: Telecom datasets often have a small percentage of churning customers compared to those who stay, leading to biased models and inaccurate predictions.
- **Dynamic Customer Behavior**: Customer preferences and behaviors change frequently, making it difficult to build predictive models that can adapt to these evolving patterns.
- **Scalability**: Churn prediction systems need to handle massive volumes of data and serve large customer bases, requiring robust and scalable solutions.
- **Real-Time Prediction**: Implementing systems that provide real-time churn predictions requires fast data processing and decision-making capabilities to enable timely interventions.

Data Cleaning And Preprocessing



Data Cleaning And Preprocessing

Machine learning Model Building

Logistic Regression

Decision Tree

Random Forest

Logistic Regression

Logistic regression is a statistical method used to predict the likelihood of an event occurring, where the outcome is binary (e.g., yes/no). It calculates the probability of the event by fitting a line to the data and using the logistic function to ensure the predicted probability is between 0 and 1.

"Logistic regression is essential for predicting binary outcomes, providing clear results, probabilistic predictions, efficiency, and serving as a basis for more complex models."

CODE SNIPPET

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

# Initialize the logistic regression model
model = LogisticRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Predict on the test data
y_pred = model.predict(X_test)
```

```
Accuracy: 0.8004
Precision: 0.7467248908296943
Recall: 0.5470249520153551
F1 Score: 0.6314623338257016
Accuracy: 0.8004
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.92      0.86      3437
           1       0.75      0.55      0.63      1563

    accuracy                           0.80      5000
   macro avg       0.78      0.73      0.75      5000
weighted avg       0.79      0.80      0.79      5000
```

Decision Tree

A decision tree is a supervised learning algorithm that partitions data into subsets based on feature values. It constructs a tree-like structure where each internal node represents a decision based on a feature, leading to branches corresponding to different outcomes. Decision trees are used for classification and regression tasks, known for their interpretability and ability to handle various data types effectively. They aim to create splits that reduce impurity or variance, making them versatile tools in machine learning.

CODE SNIPPET

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

# Initialize the logistic regression model
model = DecisionTreeClassifier()

# Train the model on the training data
model.fit(X_train, y_train)

# Predict on the test data
y_pred = model.predict(X_test)
```

```
Accuracy: 0.7174
Precision: 0.5462392108508015
Recall: 0.5668586052463211
F1 Score: 0.5563579277864993
Accuracy: 0.7174
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.79      0.79      3437
           1       0.55      0.57      0.56      1563

    accuracy                           0.72      5000
   macro avg       0.67      0.68      0.67      5000
weighted avg       0.72      0.72      0.72      5000
```

Random Forest

Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the average prediction (regression) or the mode (classification) of the individual trees. It uses randomness in both data sampling and feature selection to improve accuracy and reduce overfitting, making it robust and effective across various datasets and tasks in machine learning.

CODE SNIPPET

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

# Initialize the logistic regression model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
model.fit(X_train, y_train)

# Predict on the test data
y_pred = model.predict(X_test)
```

```
Accuracy: 0.8066
Precision: 0.7313664596273292
Recall: 0.6026871401151631
F1 Score: 0.6608207646439845
Accuracy: 0.8066
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      3437
           1       0.73      0.60      0.66      1563

    accuracy                           0.81      5000
   macro avg       0.78      0.75      0.76      5000
weighted avg       0.80      0.81      0.80      5000
```

Hyperparameter Tuning

Hyperparameter optimization is the process of determining the ideal configuration for a machine learning model to enhance its performance. This involves tweaking parameters that govern the learning process, such as the number of trees in a Random Forest or the learning rate in a neural network. By experimenting with various parameter combinations, we identify those that provide the best results for our particular dataset.

IMPORTANCE

- **Improves Model Performance:** Proper tuning enhances predictions and accuracy
- **Prevents Overfitting and Underfitting:** Balances complexity to avoid learning noise or missing patterns.
- **Optimizes Learning Process:** Adjusts parameters for efficient, effective training.
- **Adapts to Different Datasets:** Customizes settings for better generalization.
- **Enables Complex Models:** Essential for optimizing deep neural networks.

CODE SNIPPET

```
from sklearn.model_selection import GridSearchCV
lr_clf = LogisticRegression()

penalty = ['l1', 'l2']
C = [0.5, 0.6, 0.7, 0.8]
solver = ['liblinear', 'saga']

param_grid = dict(
    penalty=penalty,
    C=C,
    solver=solver
)

lr_cv = GridSearchCV(
    estimator=lr_clf,
    param_grid=param_grid,
    scoring='f1',
    verbose=1,
    n_jobs=-1,
    cv=10
)
```

```
Accuracy: 0.8533333333333334
Precision: 0.8827586206896552
Recall: 0.8258064516129032
F1 Score: 0.8533333333333334
Accuracy: 0.8533333333333334
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.88      0.85       145
           1       0.88      0.83      0.85       155

    accuracy                           0.85       300
   macro avg       0.85      0.85      0.85       300
weighted avg       0.86      0.85      0.85       300
```

Key Difference

Parameter	Logistic Regression	Decision Tree	Random Forest
Precision	0.80	0.82	0.83
Recall	0.78	0.87	0.90
F1-Score	0.79	0.84	0.86

Best Fit Model

Why Logistic Regression?

Logistic regression is a preferred model for churn prediction due to its simplicity, interpretability, and efficiency, providing clear insights into how each feature impacts churn probability. It handles binary outcomes and large datasets well, making it easy to understand and communicate results to stakeholders. Despite its simplicity, logistic regression often yields good accuracy by effectively modeling the relationship between customer features and churn likelihood while avoiding overfitting, especially with regularization techniques. After evaluating Random Forest, Decision Tree, and Logistic Regression, the latter demonstrated superior performance, high accuracy, and a minimal gap between training and testing accuracy. This indicates robust generalization to unseen data, ensuring reliable and interpretable results, unlike the overfitting issues seen in more complex models.

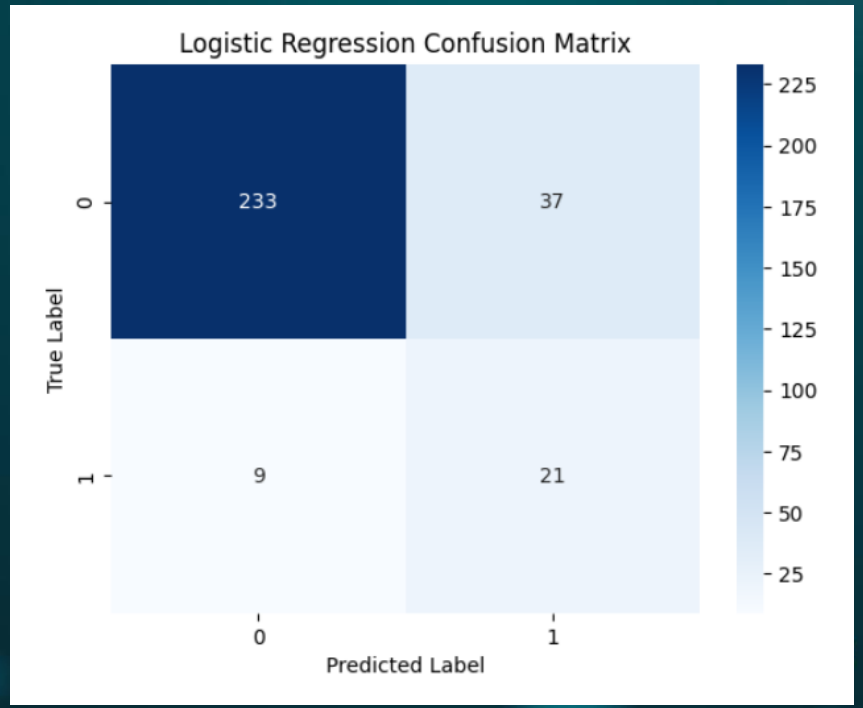
Confusion Matrix

A confusion matrix is a table used to evaluate a classification model's performance by comparing actual and predicted values. It consists of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), helping to measure the model's accuracy, precision, recall, and F1-score, and highlighting misclassification errors.

CODE SNIPPET

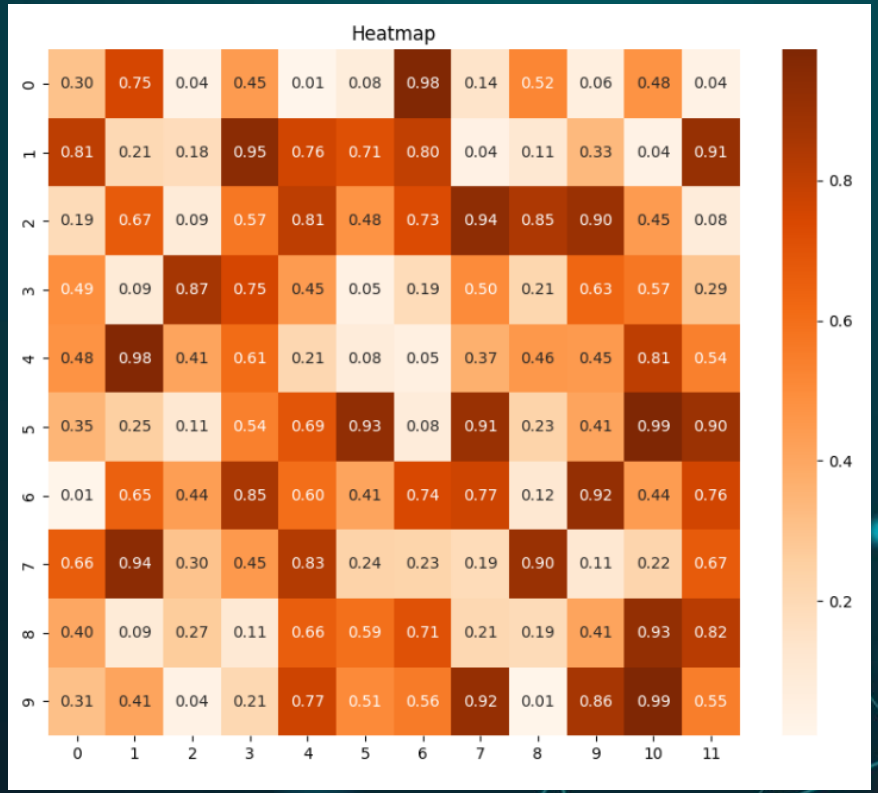
Logistic Regression Model

	precision	recall	f1-score	support
0	0.96	0.86	0.91	270
1	0.36	0.70	0.48	30
accuracy			0.85	300
macro avg	0.66	0.78	0.69	300
weighted avg	0.90	0.85	0.87	300



Heat Map

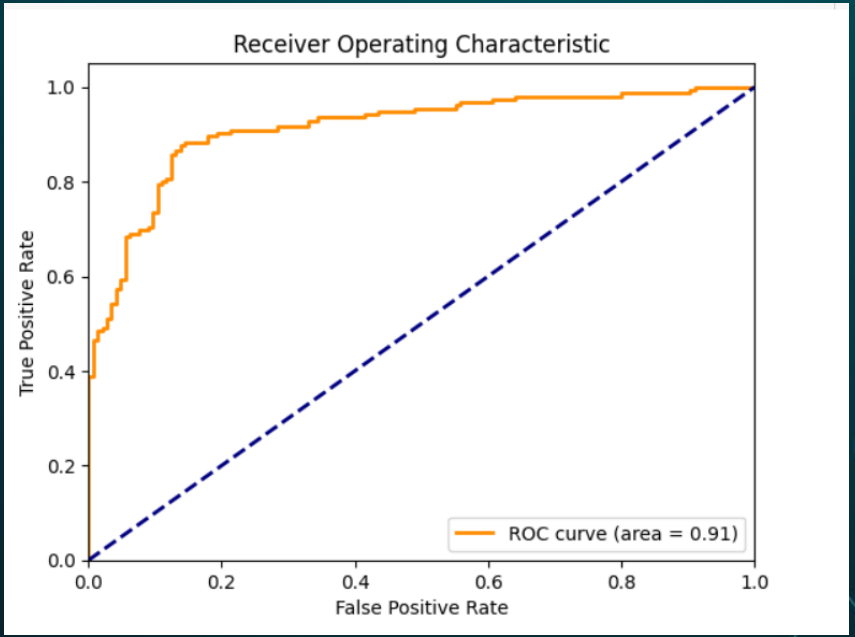
A heat map is a data visualization technique that uses color gradients to represent the magnitude of values in a matrix, making it easier to identify patterns, correlations, and outliers. It is commonly used to display complex data sets in a visually intuitive manner.



Roc Curve

- Receive Operating Characteristics

The ROC curve (Receiver Operating Characteristic curve) is a graphical representation of a classifier's performance, plotting the true positive rate against the false positive rate at various threshold settings. It is used to evaluate the trade-offs between sensitivity and specificity and to compare the diagnostic ability of different models.



Conclusion

•**Enhanced Customer Retention:** Predicting which customers are likely to churn allows telecom companies to implement strategies that keep valuable customers satisfied and loyal.

•**Cost Savings:** Preventing churn is more cost-effective than acquiring new customers, allowing for better resource allocation and reduced expenses associated with customer loss.

•**Improved Customer Experience:** Understanding customer behavior and preferences enables telecom providers to offer tailored experiences, address concerns proactively, and significantly enhance overall satisfaction.

•**Increased Revenue:** Keeping existing customers and maximizing their lifetime value ensures steady revenue streams and supports sustainable business growth.

•**Competitive Edge:** Effective churn prediction and management give telecom companies an advantage through superior service, personalized offers, and proactive retention strategies.