Telecom Customer Churn Analysis and Prediction

Predictive Analytics Using Machine Learning

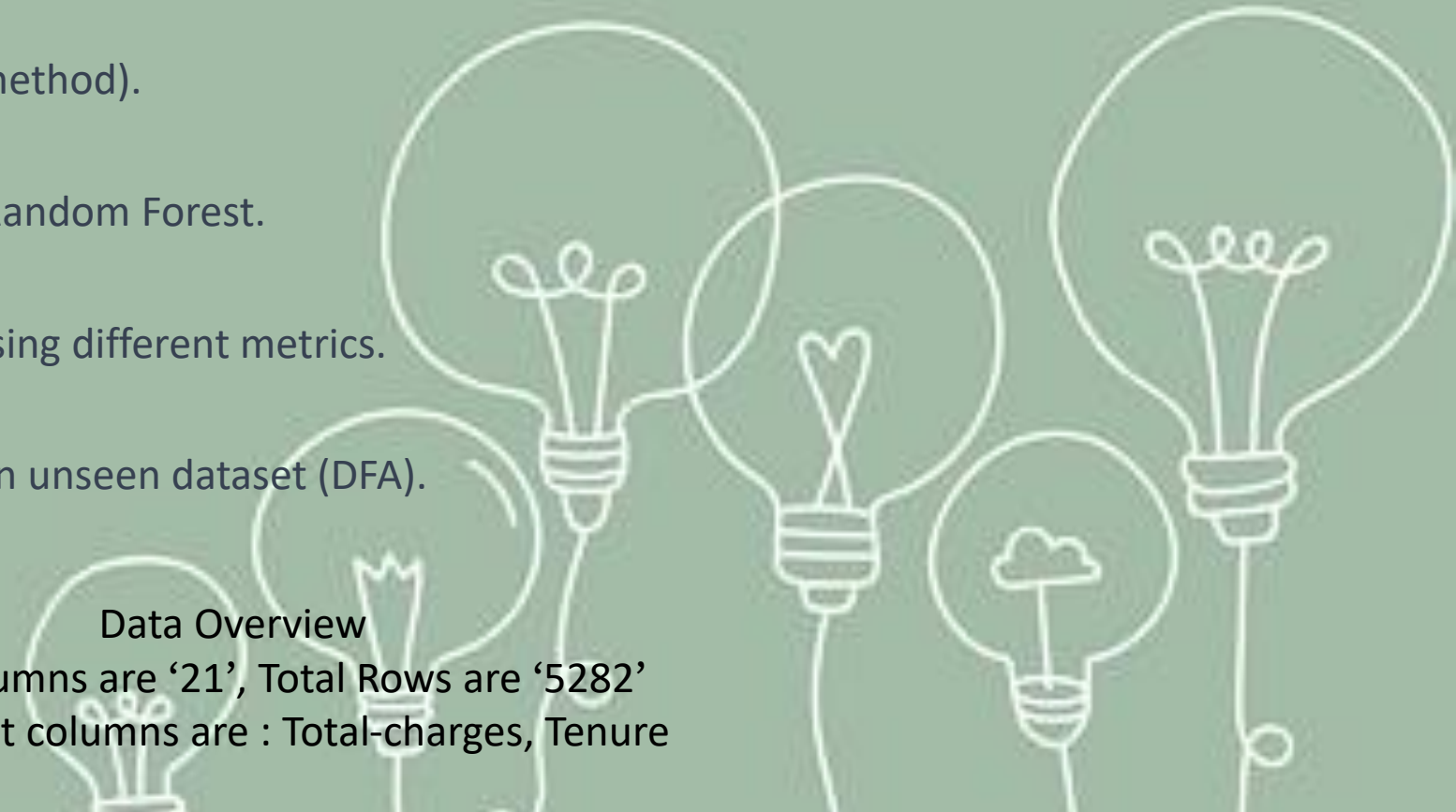
- Objective: "Our primary goal is to understand and predict customer churn in a telecom company using available data."

- Importance: "Predicting churn allows a company to target customers with retention strategies, saving potential revenue."

Presented By : Rishi Solanki

Process of Analysis

- **Data Inspection:** Checking the quality and reliability of the data.
- **Feature Engineering:** Handling bad data/anomalies, wrong data types, duplicate values, and outliers.
- **Exploratory Data Analysis:** Understanding the distribution and relationships in the data.
- **Pre-processing:** Label encoding, scaling (Z-score method).
- **Model Implementation:** Logistic Regression and Random Forest.
- **Evaluation:** Assessing the model's performance using different metrics.
- **Prediction on New Data:** Applying the model to an unseen dataset (DFA).



Data Overview
Total Columns are ‘21’, Total Rows are ‘5282’
Important columns are : Total-charges, Tenure

Data Inspection & Initial Handling

A. Introduction to the Dataset:

- **Brief Overview:**
  - Imported essential libraries: NumPy, Pandas, Seaborn, Matplotlib.
  - Used **df.head()**, **df.info()**, and **df.shape** to get an initial sense of the data.

B. Data Imbalance Issue:

- **Churn Distribution:**
  - The churn rate is imbalanced with a 75:25 ratio.
  - Visual: Pie chart showing the distribution of Churn values.

C. Identifying and Handling Anomalies:

- **Total Charges Column:**
  - Problem: Some entries were ' ' (blank spaces).
  - Solution: Replaced ' ' with NaN, then imputed NaN values using the median.

- **Data Types:**
  - Addressed issues of incorrect data types using **astype()**.

D. Null Values & Outliers:

- **Null Value Inspection:** Used **isnull()** to check for missing values.

- **Outliers:** Checked for outliers using box plots. No significant outliers found.

Duplicates and EDA process

A. Addressing Duplicates:

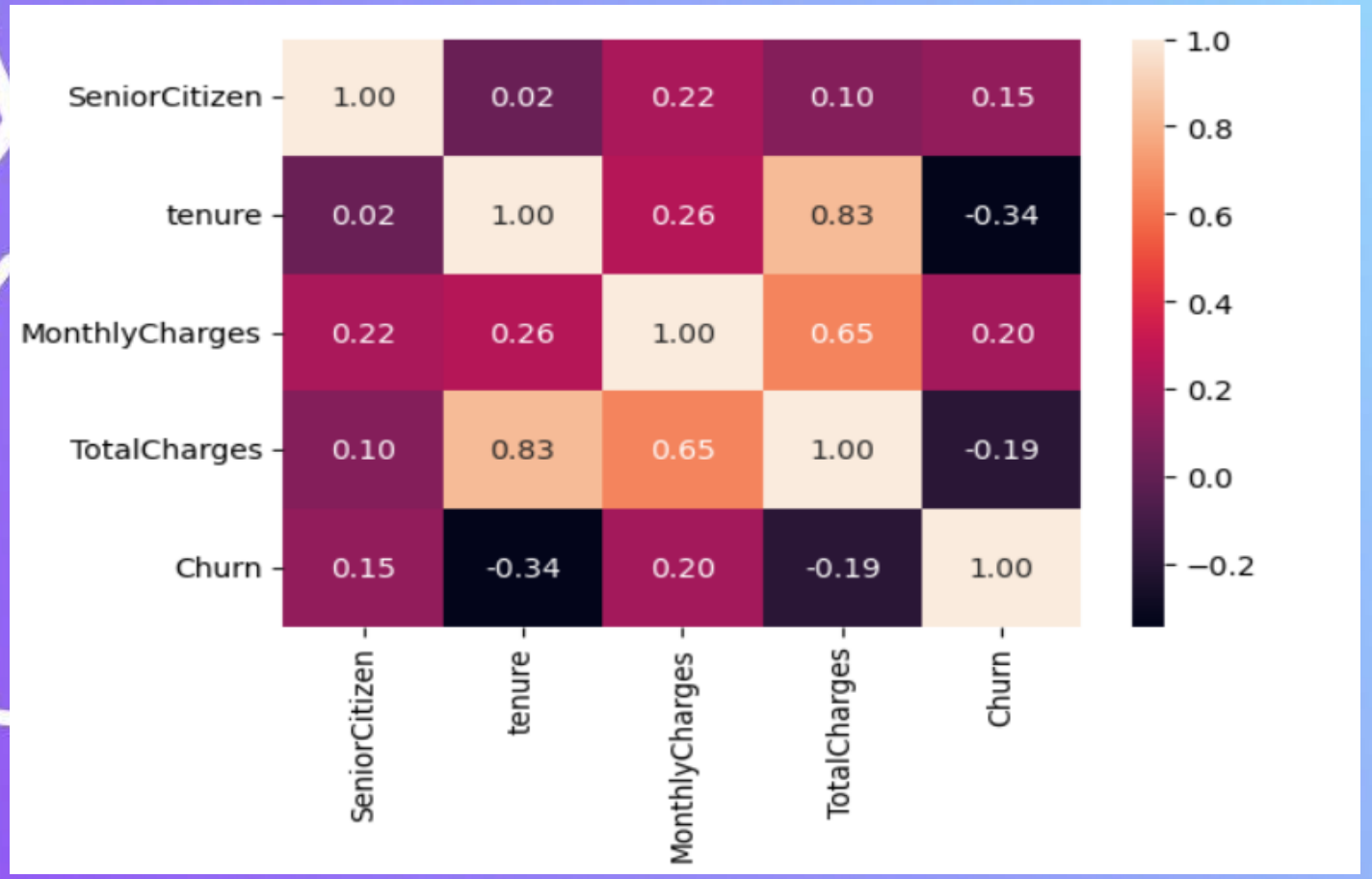
- **Identifying Duplicates:**
  - Dropped the **customer_id** column to get a true sense of data redundancy.
  - Used **df.duplicated()** to spot duplicated rows.

- **Action on Duplicates:**
  - Found 13 duplicate entries.
  - Removed duplicates using **df = df.drop_duplicates()**.

B. Visualization & Insights:

- **Heatmap for Correlation:**
  - Used a heatmap to understand the correlation among numerical features.
  - Insight: This helped in understanding which features may influence the outcome 'Churn' the most.

Heat Map



"From our analysis, it's evident that there's a strong correlation between 'tenure' and 'total charges'. Additionally, 'monthly charges' also exhibit a significant relationship with 'total charges'. However, it's noteworthy that the 'churn' column doesn't demonstrate a direct high correlation with any singular feature."

Converting Categorical to Numerical Data

1.Why Encoding?

- Machine learning models comprehend numbers, not words. Thus, we convert (or "encode") categorical data into a numerical format.

2.Testing the Waters:

- Initially, encoding was tested on the **gender** column to ensure the process's efficacy.
  - Code: **df['Gender_encoded'] = df['gender'].map({'Male':0,'Female':1})**

3.Broad Application:

- Next, the encoding technique was applied widely across several columns:
  - Example mappings:
    - **Partner**: {'No':0,'Yes':1}
    - **Internet service**: {'DSL':1,'No':0,'Fiber optic':2}
    - ... and so on for other columns.

4.Dataset Refinement:

- Post-encoding, to streamline our dataset:
  - Removed the original **gender** column since 'Gender_encoded' now held its numerical representation.

5.Encoding Verification:

- Post the encoding process, it's always a good practice to cross-verify the mapping to ensure data integrity.
  - **Partner**: [1, 0]
  - **Dependents**: [1, 0]
  - **PhoneService**: [0, 1]
  - **MultipleLines**: [2, 0, 1]
  - **InternetService**: [1, 0, 2]
  - ... (This can continue for the rest of the columns)

Scaling the Data Using Z-score

1.Why Scaling?

- In machine learning, often features have different scales. For instance, **MonthlyCharges** might range in the thousands while **tenure** can be in single digits.
- When features have different scales, some models, especially distance-based models like K-Means or models using gradient descent optimization, can be biased towards the features with higher magnitudes.
- Scaling ensures every feature gets an equal weightage and avoids bias.

2.Z-score Scaling - An Overview

- It's a standardization method where:
  - Each value is subtracted from the mean of the column and then divided by the standard deviation.
  - Mathematical representation**: X-(mean)/SD**

3.Features Scaled:

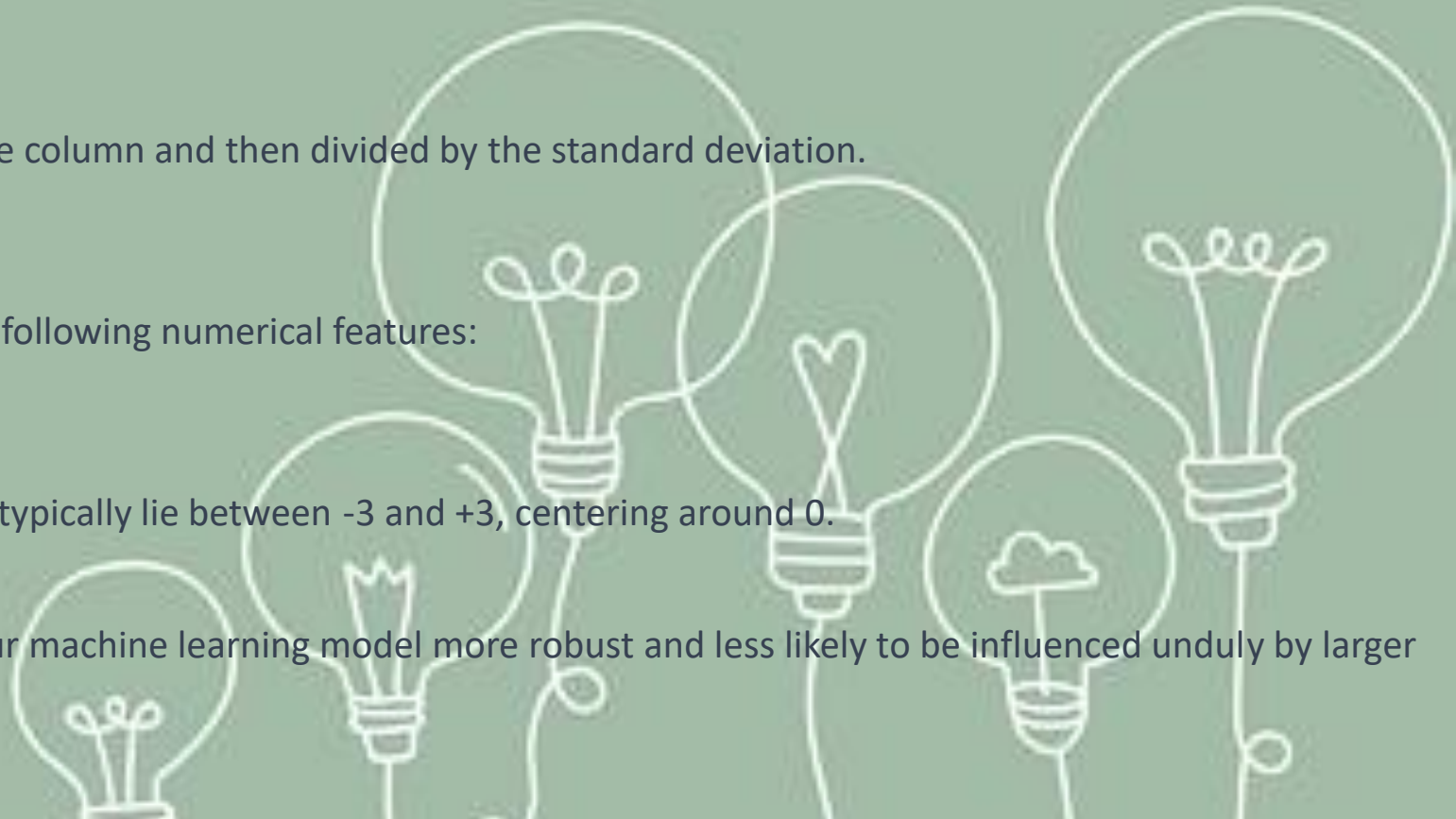
- For this dataset, Z-score scaling was applied to the following numerical features:
  - **Total-Charges, tenure, Monthly-Charges**

4.Resultant Range:

- After Z-score scaling, the values for these features typically lie between -3 and +3, centering around 0.

5.The Big Advantage:

- Ensuring all features have the same scale makes our machine learning model more robust and less likely to be influenced unduly by larger magnitude features.



Model Implementation

1.Choice of Model - Logistic Regression:

- Logistic Regression is especially useful when predicting the probability of an occurrence, especially when the output or the target variable is binary - in our case, it's 'Churn' with values either 'Yes' or 'No'.
- Given the nature of our problem and the data, Logistic Regression stands out as a go-to model.

2.Data Preparation:

- **Train-test Split**: The dataset is divided into two parts:
  - **Training data (80%)**: Used to train the model.
  - **Testing data (20%)**: Used to evaluate the model's performance on unseen data.
- Reason for Split: Ensures the model is robust and doesn't simply memorize the data, which helps in validating its performance on fresh, unseen data.

3.Handling Imbalanced Data:

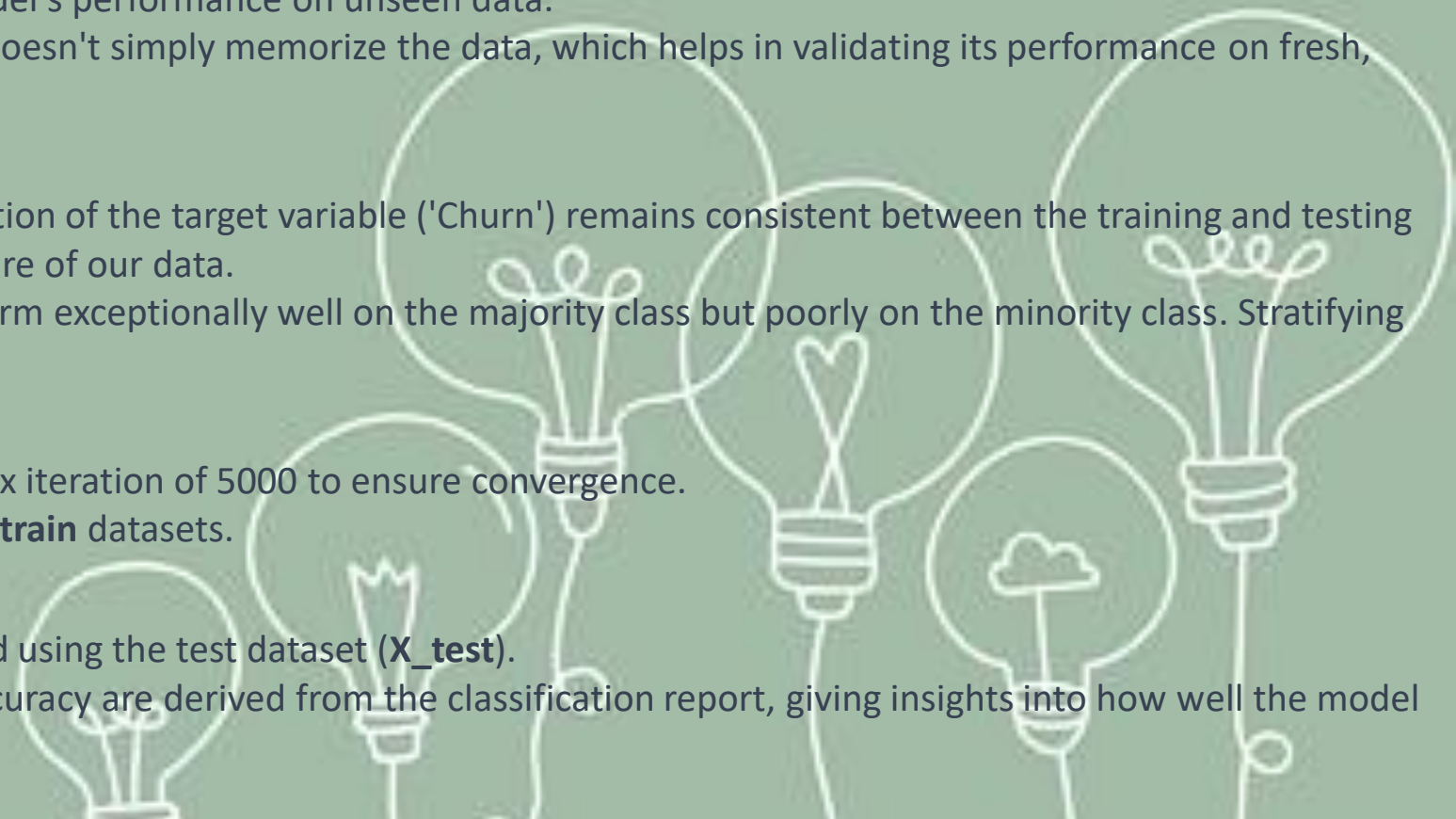
- The **stratify=y** parameter ensures that the distribution of the target variable ('Churn') remains consistent between the training and testing datasets. This is essential given the imbalanced nature of our data.
- Imbalanced datasets can lead to models that perform exceptionally well on the majority class but poorly on the minority class. Stratifying counters this.

4.Building and Training the Model:

- The Logistic Regression model is created with a max iteration of 5000 to ensure convergence.
- The model is then trained using the **X_train** and **y_train** datasets.

5.Evaluating the Model:

- Once trained, the model's performance is validated using the test dataset (**X_test**).
- Metrics such as Precision, Recall, F1-Score, and Accuracy are derived from the classification report, giving insights into how well the model is performing.



Logistic Regression Model Score

Classification Report for Training Data:

	precision	recall	f1-score	support
0	0.84	0.90	0.86	3101
1	0.64	0.51	0.57	1114
accuracy			0.79	4215
macro avg	0.74	0.70	0.71	4215
weighted avg	0.78	0.79	0.79	4215

Classification Report for Test Data:

	precision	recall	f1-score	support
0	0.84	0.88	0.86	775
1	0.61	0.52	0.56	279
accuracy			0.78	1054
macro avg	0.72	0.70	0.71	1054
weighted avg	0.78	0.78	0.78	1054

The Logistic Regression model demonstrates consistent performance on both the training and test datasets, with an accuracy of 79% and 78% respectively. This balanced performance indicates a robust model that's neither overfitting nor underfitting. Such consistency suggests the model is well-tuned and reliable for predicting customer churn but we can improve this model, with the help of smot method (synthetic minority over sampling technique)

Model Implementation

1.Model Choice: The **Random Forest Classifier** is an ensemble learning technique that can handle binary output such as our churn prediction, making it an appropriate model choice.

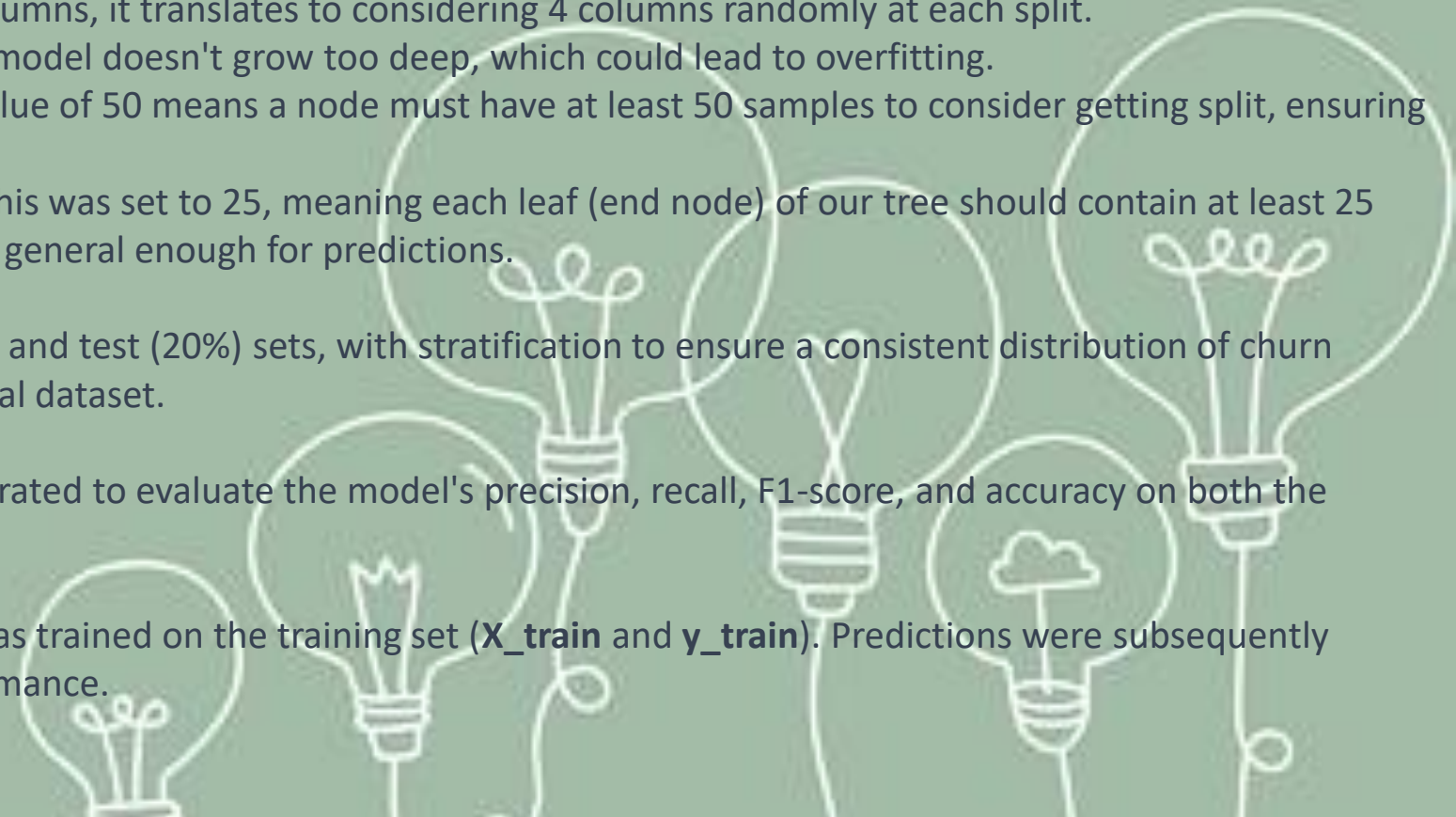
2.Parameter Details:

- **Number of Trees (n_estimators):** 100 trees were used to ensure diverse decision boundaries and to achieve a robust ensemble.
- **Maximum Features (max_features):** We chose 'sqrt' which means at each split in the tree, the algorithm considers the square root of the total number of features. In our dataset with 19 columns, it translates to considering 4 columns randomly at each split.
- **Tree Depth (max_depth):** Set at 15, ensuring the model doesn't grow too deep, which could lead to overfitting.
- **Minimum Sample Split (min_samples_split):** A value of 50 means a node must have at least 50 samples to consider getting split, ensuring the tree doesn't become overly complex.
- **Minimum Samples in Leaf (min_samples_leaf):** This was set to 25, meaning each leaf (end node) of our tree should contain at least 25 samples. This setting ensures that the tree remains general enough for predictions.

3.Splitting Strategy: Data was divided into training (80%) and test (20%) sets, with stratification to ensure a consistent distribution of churn outcomes, reflecting the imbalanced nature of the original dataset.

5.Performance Metrics: Classification reports were generated to evaluate the model's precision, recall, F1-score, and accuracy on both the training and test datasets.

4.Training and Prediction: After initializing, the model was trained on the training set (**X_train** and **y_train**). Predictions were subsequently made on both the training and test data to assess performance.



Random Forest Classifier Model Score

Classification Report for Training Data:

	precision	recall	f1-score	support
0	0.84	0.93	0.88	3101
1	0.72	0.49	0.58	1114
accuracy			0.82	4215
macro avg	0.78	0.71	0.73	4215
weighted avg	0.81	0.82	0.80	4215

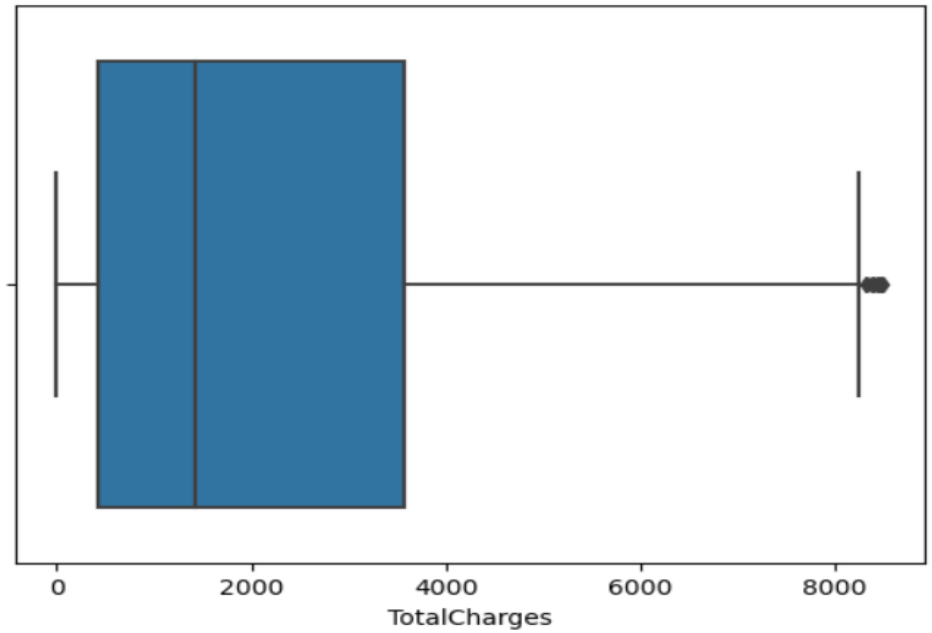
Classification Report for Test Data:

	precision	recall	f1-score	support
0	0.82	0.91	0.86	775
1	0.65	0.44	0.53	279
accuracy			0.79	1054
macro avg	0.73	0.68	0.70	1054
weighted avg	0.77	0.79	0.77	1054

The Random Forest model achieved an accuracy of 82% on training data and 79% on test data. However, with a 44% recall for churn on the test set, the model might miss identifying several potential churn cases, indicating it's not entirely robust, in this case we can again go through our feature-engineering process or we can try smot on method on this for better result.

Predictions on New Data (DFA)

1. **Briefly explain the new dataset without known churn outcomes:** The DFA dataset comprises customer details similar to our original training set, but lacks the actual 'Churn' outcomes. This dataset offers an opportunity to predict potential churners based on the trained models.
2. **Display a few sample predictions:** While processing DFA, we observed outliers in the 'Total-Charges' feature. A box plot visualization confirms outliers predominantly in the upper bound. Utilizing the IQR method, we capped these outliers, ensuring more robust predictions.

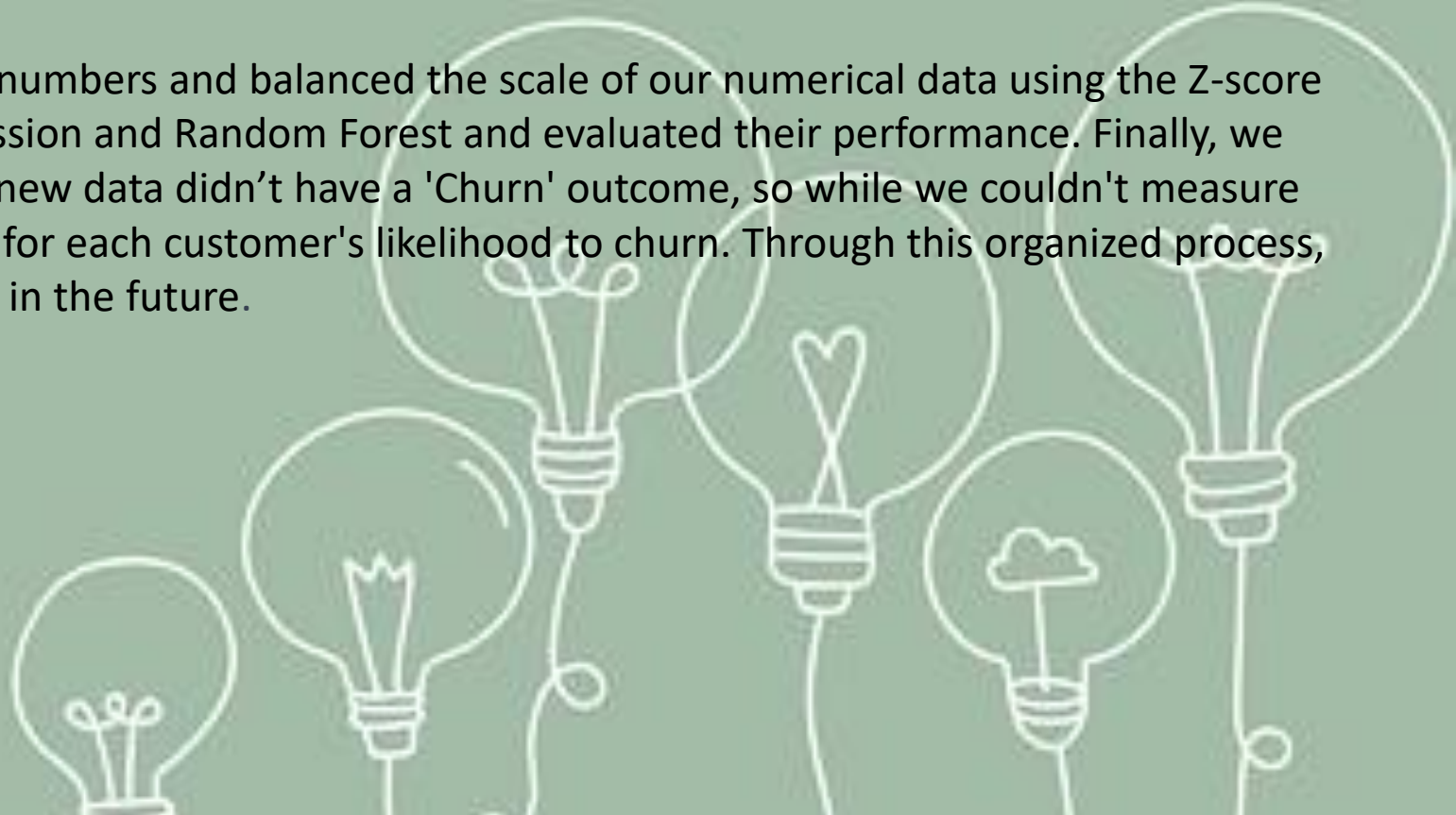



```
[156]: # calculate iqr,q1,q3
       q1 = dfa['TotalCharges'].quantile(0.25)
       q3 = dfa['TotalCharges'].quantile(0.75)
       iqr = q3 - q1
       #define bound
       lb = q1 - 1.5* iqr
       up = q3 + 1.5* iqr
       #solve the problem
       dfa['TotalCharges']= np.where(dfa['TotalCharges']<lb,lb,dfa['TotalCharges'])
       dfa['TotalCharges']= np.where(dfa['TotalCharges']>up,up,dfa['TotalCharges'])
```

From Inspection to Predictions: Our Data Journey

We began by carefully examining our data to make sure it's trustworthy. In the Feature Engineering step, we fixed issues like wrong data types, duplicates, and outliers. Next, with Exploratory Data Analysis, we got a clearer view of our data's patterns.

For Pre-processing, we changed category values to numbers and balanced the scale of our numerical data using the Z-score method. We then built models using Logistic Regression and Random Forest and evaluated their performance. Finally, we applied our trained model to the DFA dataset. This new data didn’t have a 'Churn' outcome, so while we couldn't measure accuracy in the usual way, we did make predictions for each customer's likelihood to churn. Through this organized process, we got a good idea of which customers might leave in the future.



Email: solankirishi004@gmail.com
Contact no : +91 8758478045