**Infosys Springboard Internship 4.0**



**Telecom Churn Prediction Documentation**

**By**

**Amit Kumar Jha**

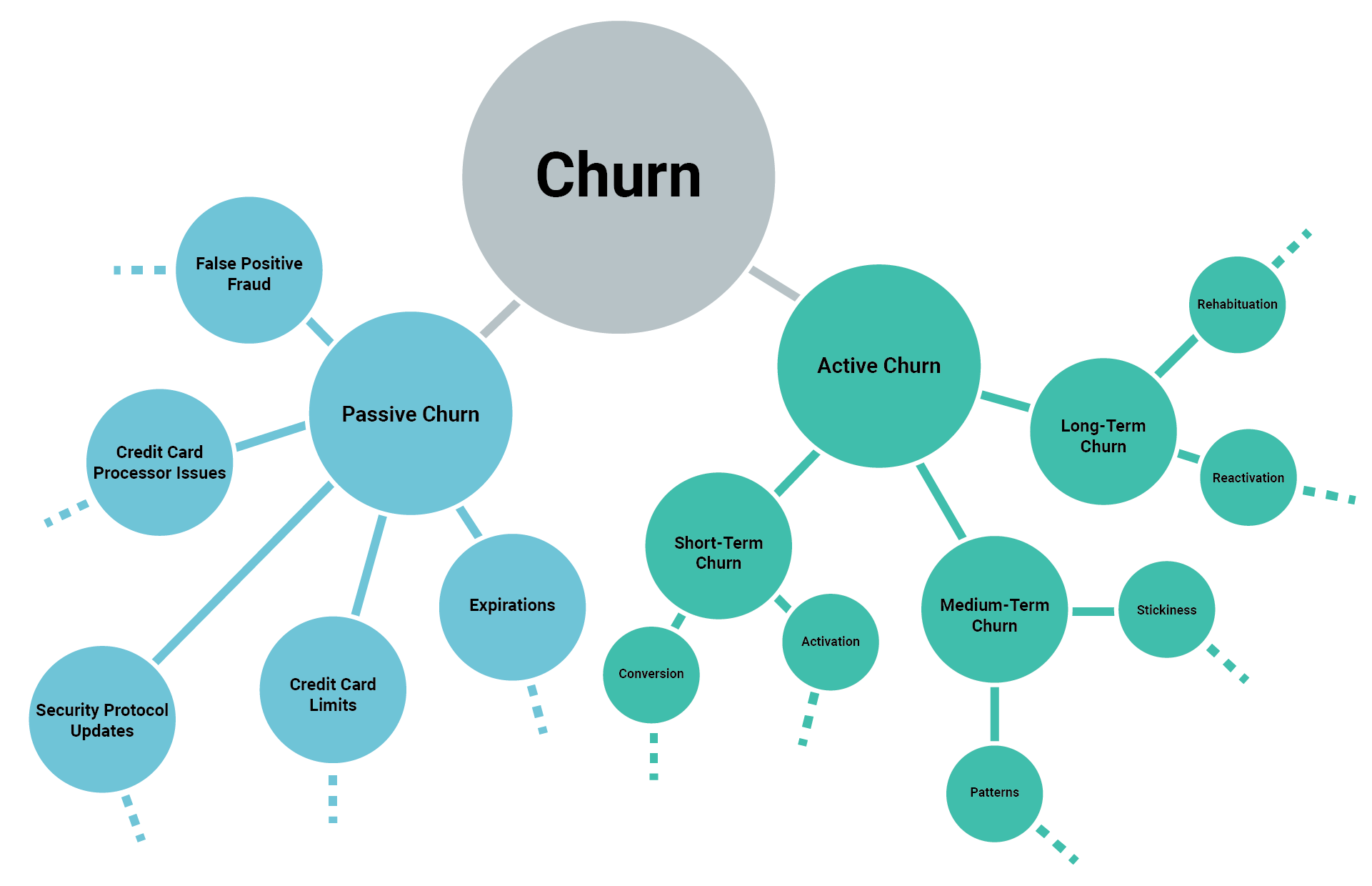
**Table of Contents**

1. **Introduction**
2. **Milestone 1: Data Gathering and Initial Analysis**
   * 1.1 Gathering Data
   * 1.2 Understanding the Data
   * 1.3 Download Required Tools
3. **Milestone 2: Data Pre-processing**
   * 2.1 Convert Data Types
   * 2.2 Removing Duplicate Records
   * 2.3 Removing Unique Value Variables
   * 2.4 Removing Zero Variance Variables
   * 2.5 Outlier Treatment
   * 2.6 Missing Value Treatment
   * 2.7 Removing Highly Correlated Variables
   * 2.8 Multicollinearity (VIF > 5)
4. **Milestone 3: Random Forest Model Building**
   * 3.1 Data Splitting
   * 3.2 Model Building
   * 3.3 Model Evaluation
5. **Milestone 4: Multiple Model Building and Comparison**
   * 4.1 Building Other Models
   * 4.2 Model Comparison
   * 4.3 Hyperparameter Tuning
6. **Final Steps: Selecting the Best Model**
   * 5.1 Data Pre-processing Recap
   * 5.2 Final Model Selection
   * 5.3 Documentation and Reporting
7. **Conclusion**
8. **References**

**Introduction**

Churn prediction is crucial for telecom companies aiming to reduce customer turnover. By predicting which customers are likely to leave, companies can take proactive steps to retain them. This documentation outlines the entire process of building a churn prediction model, from data gathering and pre-processing to model building, evaluation, and final selection.

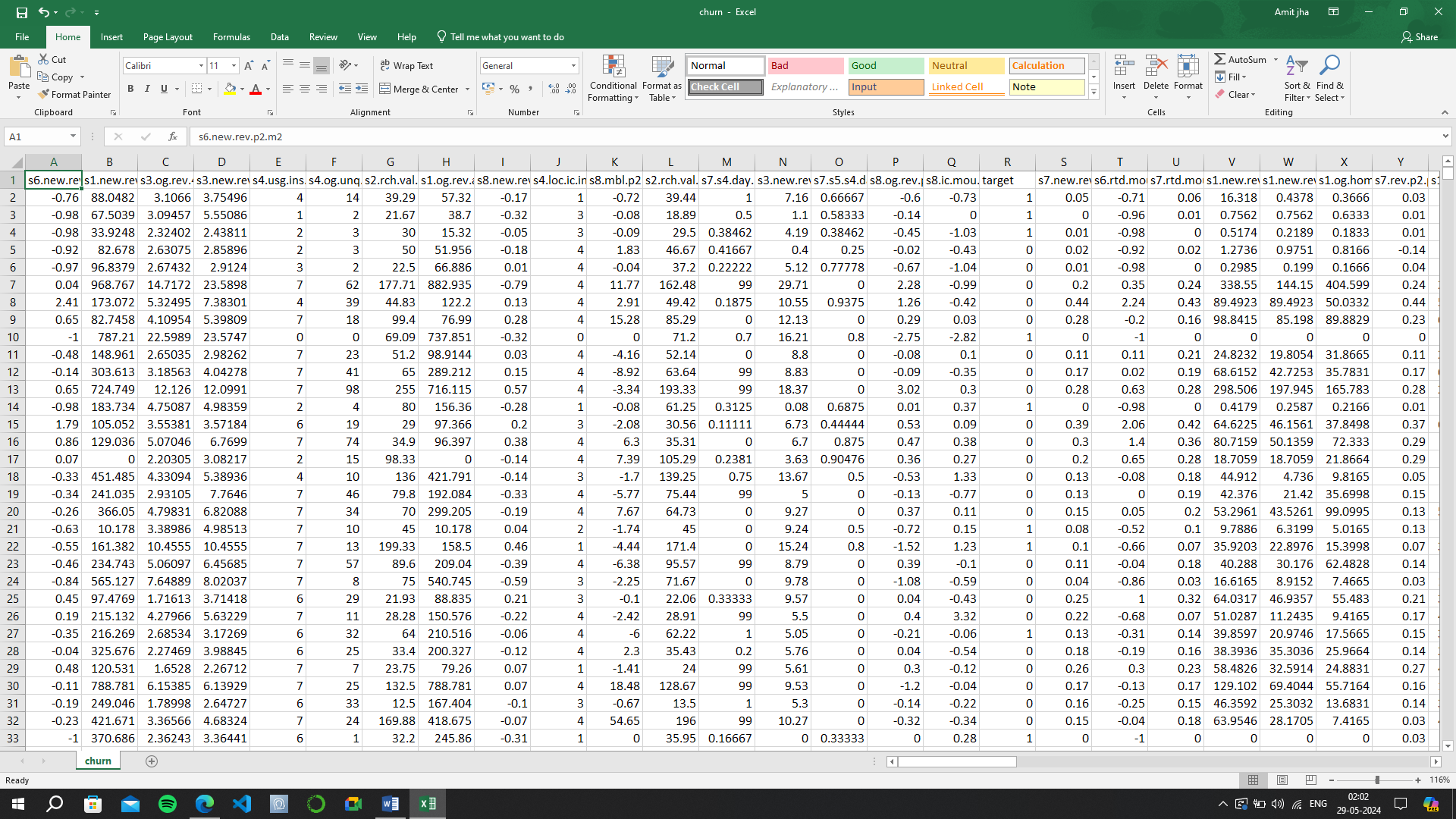
Predicting churn helps telecom companies safeguard their revenue streams. Acquiring new customers is often more expensive than retaining existing ones. By identifying customers at risk of leaving, companies can take targeted actions, such as offering personalized discounts or improving service features, to retain them. This proactive approach not only reduces marketing and acquisition costs but also ensures a stable and predictable revenue base.



**Milestone 1: Data Gathering**

**Step 1 - Gathering Data**

**Description:** The first step is to collect all relevant data. This data can come from various sources, such as company databases, CRM systems, and customer interaction logs. We used SQL for querying databases and Python libraries like Pandas and NumPy for data manipulation. Gathering a comprehensive dataset is crucial as it directly impacts the model's performance and reliability.



**Step 2 - Understanding the Data**

**Need:** To identify data issues, understand feature relationships, and gain insights.

**Description:** Before diving into model building, it's essential to understand the data. We inspected the columns, data types, and generated summary statistics. Visualization tools like Matplotlib and Seaborn helped us understand data distributions and relationships. This step helps identify potential data issues and gives insights into how different features relate to churn.

**Step 3 - Download Required Tools**

**Need:** To ensure the necessary software and libraries are available for data analysis and model building.

**Description:** To work with the data, we needed several tools:

* Python
* Jupyter Notebook
* Pandas
* NumPy
* Matplotlib
* Seaborn
* Scikit-learn

During this phase, we encountered typical challenges like compatibility issues and installation errors. Ensuring the environment is set up correctly is vital for smooth data processing and analysis.

**Milestone 2: Data Pre-processing**

Data pre-processing is essential for transforming raw data into a clean, usable format, which is critical for accurate analysis and effective machine learning. Key reasons for data pre-processing include handling missing values, removing noise, ensuring consistency, normalizing and scaling features, extracting and selecting relevant features, encoding categorical variables, balancing the dataset, detecting and handling outliers, improving model performance, and ensuring data quality.

### Main Steps in Data Pre-processing:

1. **Data Cleaning**: Address missing values, remove duplicates, and correct errors.
2. **Data Integration**: Combine data from various sources and ensure consistency.
3. **Data Transformation**: Normalize, scale, and encode data.
4. **Data Reduction**: Select important features and reduce dimensionality.
5. **Data Discretization**: Convert continuous data into discrete categories.
6. **Data Augmentation**: Generate synthetic data to balance the dataset and increase its size.

Effective data pre-processing leads to reliable, high-quality data that enhances the performance and accuracy of machine learning models.

**Step 1 - Convert Data Types**

**Need:** To ensure all variables are correctly interpreted by the analysis tools.

**Description:** Some variables were misclassified. For example, numerical values might be stored as strings. We ensured all variables had appropriate data types, converting numerical strings to integers where necessary. Correct data types are essential for accurate analysis and modelling.

**Step 2 - Removing Duplicate Records**

**Need:** To prevent skewing the analysis and model training.

**Description:** Duplicates can skew analysis and model training. We used Pandas' drop\_duplicates() function to eliminate redundant data entries. This step ensures that each data point is unique, providing a cleaner dataset.

**Step 3 - Removing Unique Value Variables**

**Need:** To remove variables that do not provide predictive power.

**Description:** Variables with no predictive power, such as columns with unique values across all records, were identified and removed. These variables do not contribute to the model and can be safely discarded to reduce complexity.

**Step 4 - Removing Zero Variance Variables**

**Need:** To remove non-informative features that do not vary.

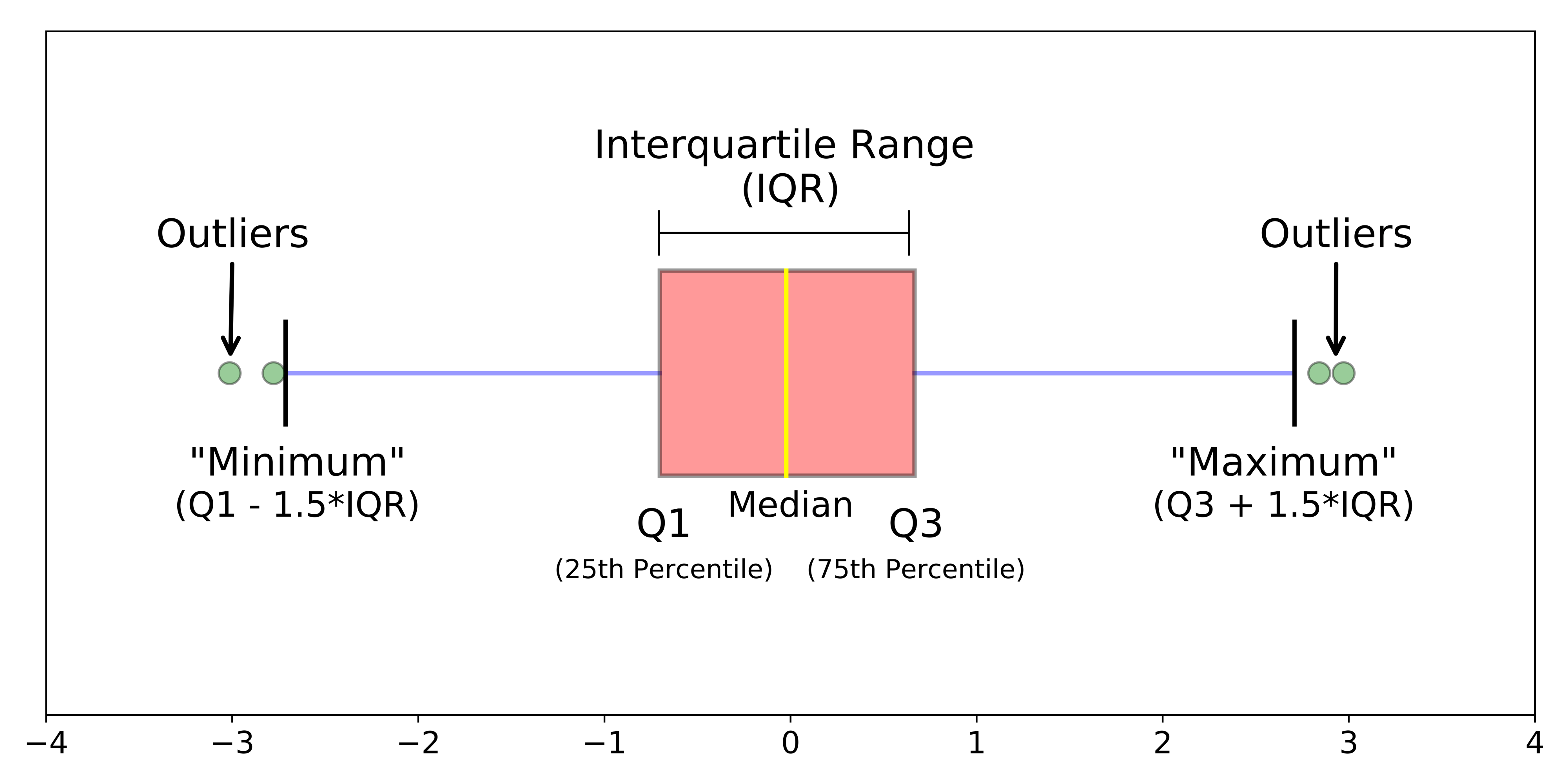
**Description:** Columns with zero variance, which means they have the same value across all records, do not contribute to the model's predictive power. We identified and dropped these columns to simplify the dataset.

**Step 5 - Outlier Treatment**

**Need:** To prevent extreme values from distorting the model.

**Description:** Outliers can distort model performance. We used several techniques to handle them:

* **Boxplot Method:** Removed outliers beyond Q1-(1.5*IQR) and Q3+(1.5*IQR).



* **Standardization:** Applied the +/- 3 Sigma approach.
* **Capping & Flooring:** Set upper and lower bounds for outliers. Handling outliers ensures that extreme values do not unduly influence the model.

**Step 6 - Missing Value Treatment**

**Need:** To handle incomplete data without losing significant information.

**Description:** Handling missing data is critical. We:

* Removed records with less than 5% missing values.
* Removed variables with more than 50% missing values.
* Imputed missing values using the mean or median for numerical variables and the mode for categorical variables. Addressing missing values prevents loss of valuable information and improves model accuracy.

**Step 7 - Removing Highly Correlated Variables**

**Need:** To reduce redundancy and multicollinearity.

**Description:** To reduce multicollinearity, we calculated the correlation matrix and removed one of each pair of highly correlated variables. High correlation between two variables means that they provide redundant information, which can negatively affect model performance.

**Steps for removing variables based on correlation analysis:**

1. **Calculate Correlation Matrix**: Compute the correlation coefficients between all pairs of variables in the dataset.
2. **Visualize Correlation Heatmap**: Create a heatmap to visually inspect the correlation matrix and identify highly correlated variables.
3. **Set Correlation Threshold**: Decide on a threshold value (e.g., 0.7 or -0.7) to define high correlation.
4. **Identify Highly Correlated Variables**: Look for pairs of variables with correlation coefficients above the set threshold.
5. **Evaluate Variable Importance**: Assess the importance of each variable to the prediction task and business context.
6. **Choose One Variable from Each Pair**: Select one variable to retain from each highly correlated pair based on relevance and predictive power.
7. **Repeat for All Pairs**: Iterate through all pairs of highly correlated variables, removing redundant variables until no more removals can be made without compromising model performance.
8. **Validate Model Performance**: Evaluate the model's performance on a validation dataset after removing correlated variables to ensure stability and improvement.
9. **Document Rationale**: Document the reasons for removing specific variables in the project documentation for transparency and reproducibility.

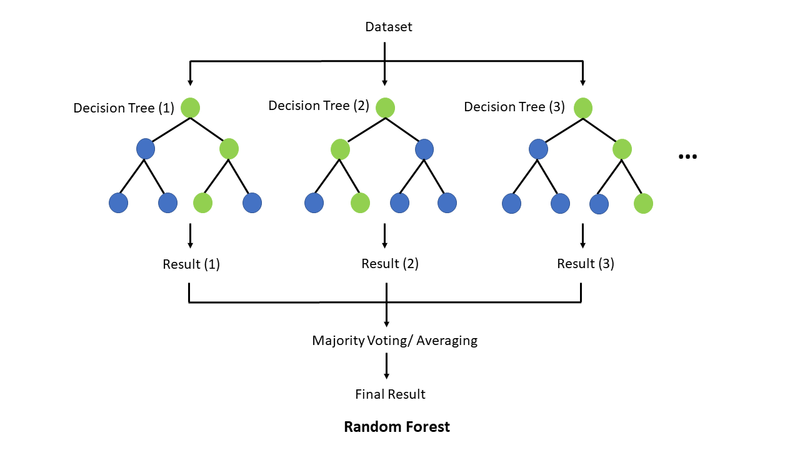
**Step 8 - Multicollinearity (VIF > 5)**

**Need:** To ensure independent variables are truly independent.

**Description:** Multicollinearity refers to the situation where several independent variables are highly correlated, making it difficult to isolate the effect of each variable. We addressed this by calculating the Variance Inflation Factor (VIF) and removing variables with VIF > 5. A high VIF indicates that a variable is highly correlated with other variables in the dataset, which can inflate standard errors and make the model less reliable.

**Milestone 3: Random Forest Model Building**

Random Forest is a powerful tree learning technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance. In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results. Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments.



**Step 1 - Data Splitting**

**Need:** To evaluate the model's performance on unseen data.

**Description:** We split the data into training and testing sets using Scikit-learn's train\_test\_split() function to ensure our model could be properly evaluated. Typically, 70-80% of the data is used for training, and the remaining 20-30% is used for testing. This split allows us to evaluate the model's performance on unseen data.

**Step 2 - Model Building**

**Need:** To create a robust initial model.

**Description:** We started with a Random Forest model. This involved:

* Initializing the Random Forest classifier.
* Training the model using the training data. Random Forest is an ensemble learning method that builds multiple decision trees and merges them to get a more accurate and stable prediction.

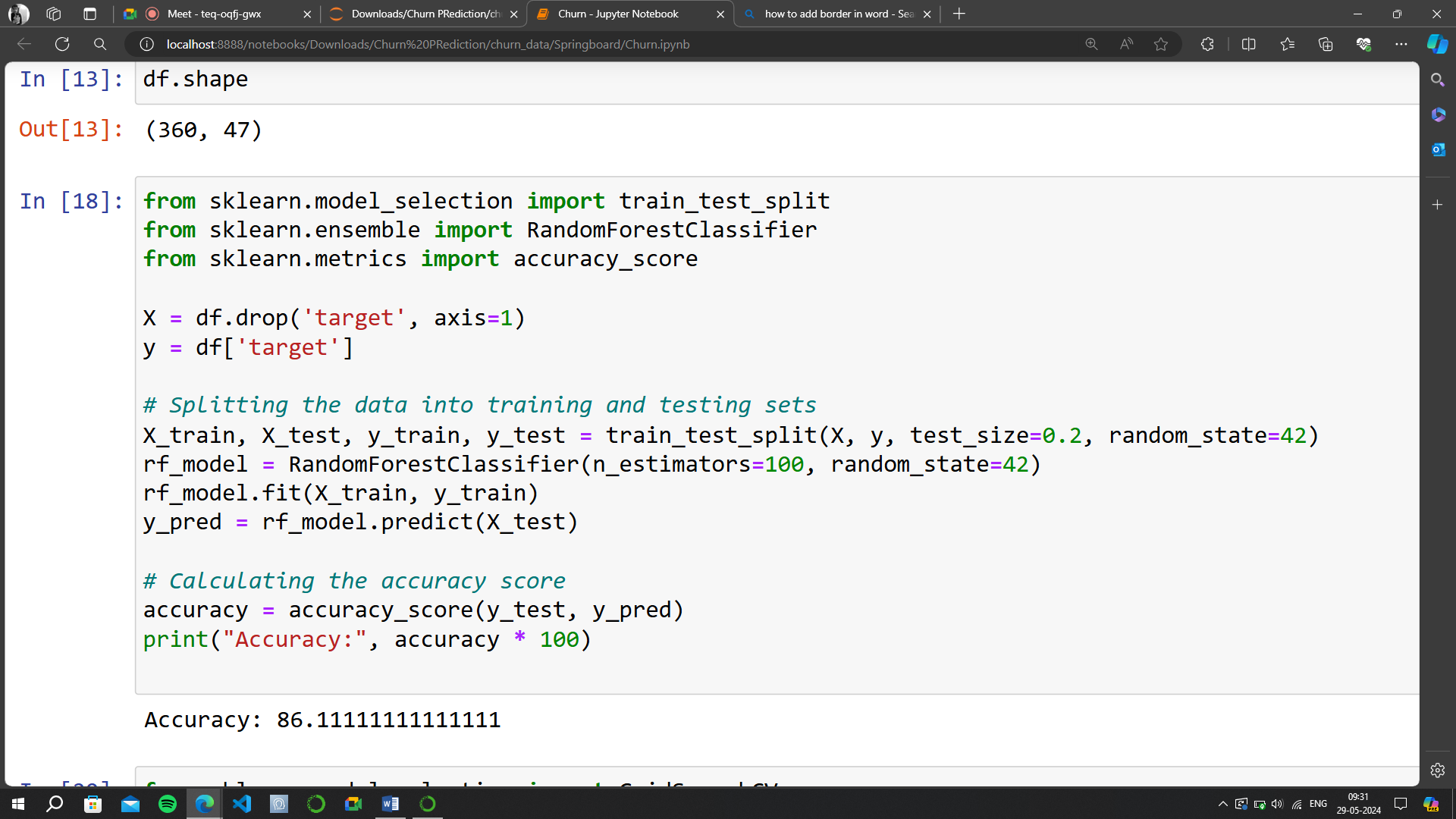
**Step 3 - Model Evaluation**

**Need:** To assess and validate model performance.

**Description:** To assess the model's performance, we looked at various metrics:

* **Accuracy:** The ratio of correctly predicted instances to the total instances.
* **Precision:** The ratio of correctly predicted positive observations to the total predicted positives.
* **Recall:** The ratio of correctly predicted positive observations to all observations in the actual class.
* **F1-score:** The weighted average of Precision and Recall.
* **AUC-ROC Curve:** The Area Under the Receiver Operating Characteristic Curve, which measures the ability of the model to distinguish between classes. These metrics provide a comprehensive view of the model's performance.

**Output**



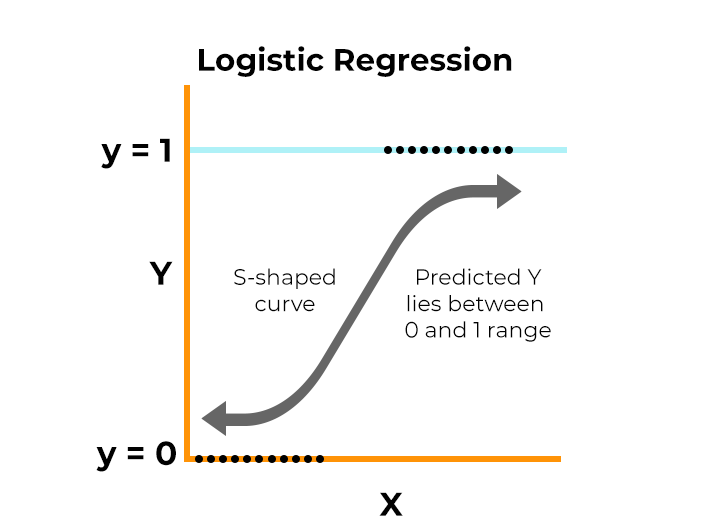
**Milestone 4: Multiple Model Building and Comparison**

**Step 1 - Building Other Models**

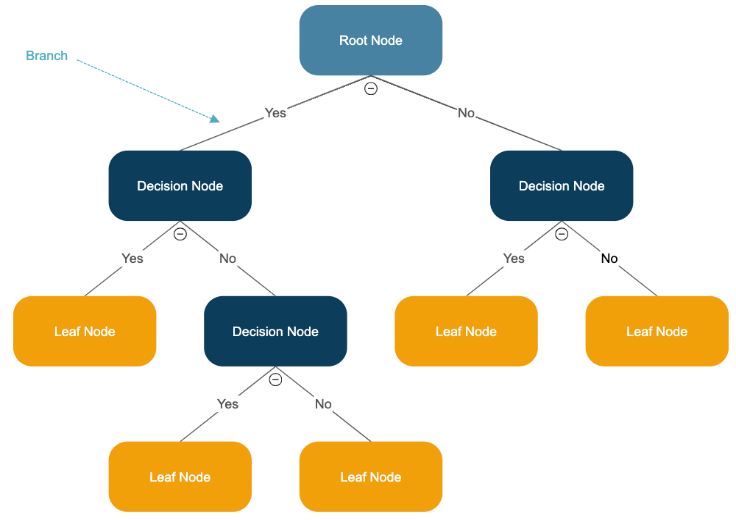
**Need:** To explore and identify the most effective algorithm.

**Description:** To improve prediction accuracy, we explored different algorithms including:

* **Logistic Regression:** A statistical model that uses a logistic function to model a binary dependent variable.



* **Decision Trees:** A model that uses a tree-like graph of decisions and their possible consequences.



**Step 2 - Model Comparison**

**Need:** To select the best performing model based on various metrics.

**Description:** We compared the performance of these models using the same metrics mentioned in section 3.3. Visualizations, such as ROC curves and confusion matrices, helped us understand which model performed best. Comparing models ensures that we select the most effective one.

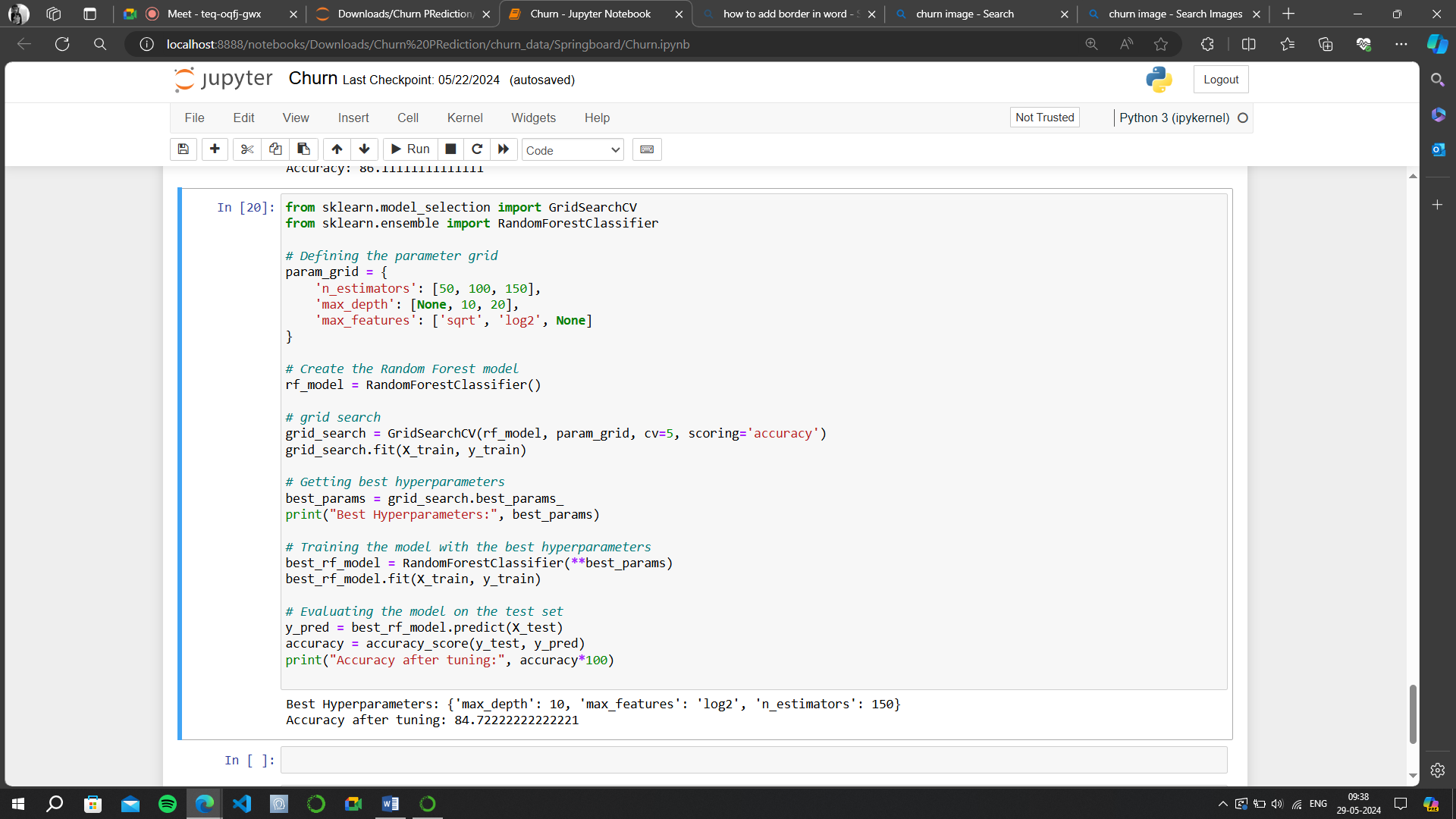
**Step 3 - Hyperparameter Tuning**

**Need:** To optimize model performance.

**Description:** We optimized model parameters to improve performance using techniques like:

* **Grid Search:** Exhaustively searching through a manually specified subset of the hyperparameter space.
* **Random Search:** Randomly searching through the hyperparameter space.
* **Bayesian Optimization:** A probabilistic model to find the minimum of a function, making the hyperparameter tuning process more efficient. Hyperparameter tuning helps fine-tune the model for better performance.

**Output**



**Final Steps: Selecting the Best Model**

**Step 1 - Data Pre-processing Recap**

**Need:** To ensure all data pre-processing steps were correctly implemented.

**Description:** We reviewed all pre-processing steps to ensure they were correctly implemented. This ensures the data is clean and ready for final model evaluation.

**Step 2 - Final Model Selection**

**Need:** To finalize the best model for deployment.

**Description:** Based on our evaluation metrics and business considerations, we chose the best-performing model. This model provided the best balance of accuracy, precision, recall, and other performance metrics.

**Step 3 - Documentation and Reporting**

**Need:** To provide a detailed and transparent account of the process.

**Description:** We documented the entire process, summarizing data pre-processing steps, model building and tuning, and presenting final model performance and recommendations. Proper documentation ensures that the process is transparent and reproducible.

**Conclusion**

As I wrap up this documentation journey, I reflect on the extensive process of creating a telecom churn prediction model. It all started with gathering data from different sources, which was quite an adventure in itself. Once I had the data in hand, I dove into understanding it, looking at what each piece meant and how they fit together. Then came the real work of cleaning it up—removing duplicates, fixing errors, and making sure everything was in order.

The modeling phase was where things got really interesting. I tried out several different algorithms, feeling a bit like a scientist in a lab. Each model had its own strengths and weaknesses, and it was exciting to see how they performed. I spent a lot of time tweaking parameters, trying to get the best results possible.

But perhaps the most satisfying part of this whole process was seeing the model in action. Knowing that it could help telecom companies predict customer churn and take proactive steps to prevent it felt really impactful. It's a small contribution, but one that could make a big difference in the long run.

As I look ahead, I see so many possibilities for improvement and further exploration. There's always more data to gather, more models to try, and more ways to refine the ones I already have. It's a never-ending journey, but one that's incredibly rewarding. And who knows? Maybe one day, I'll look back on this documentation and see it as just the beginning of something even bigger.

**References**

* **Pandas Documentation** - Pandas Documentation
* **NumPy Documentation** - NumPy Documentation
* **Matplotlib Documentation** - Matplotlib Documentation
* **Seaborn Documentation** - Seaborn Documentation
* **Scikit-learn Documentation** - Scikit-learn Documentation
* **Geekforgeeks -** [GeeksforGeeks | A computer science portal for geeks](https://www.geeksforgeeks.org/)
* **Random Forest** - [YouTube Video](https://www.youtube.com/watch?v=F9uESCHGjhA)