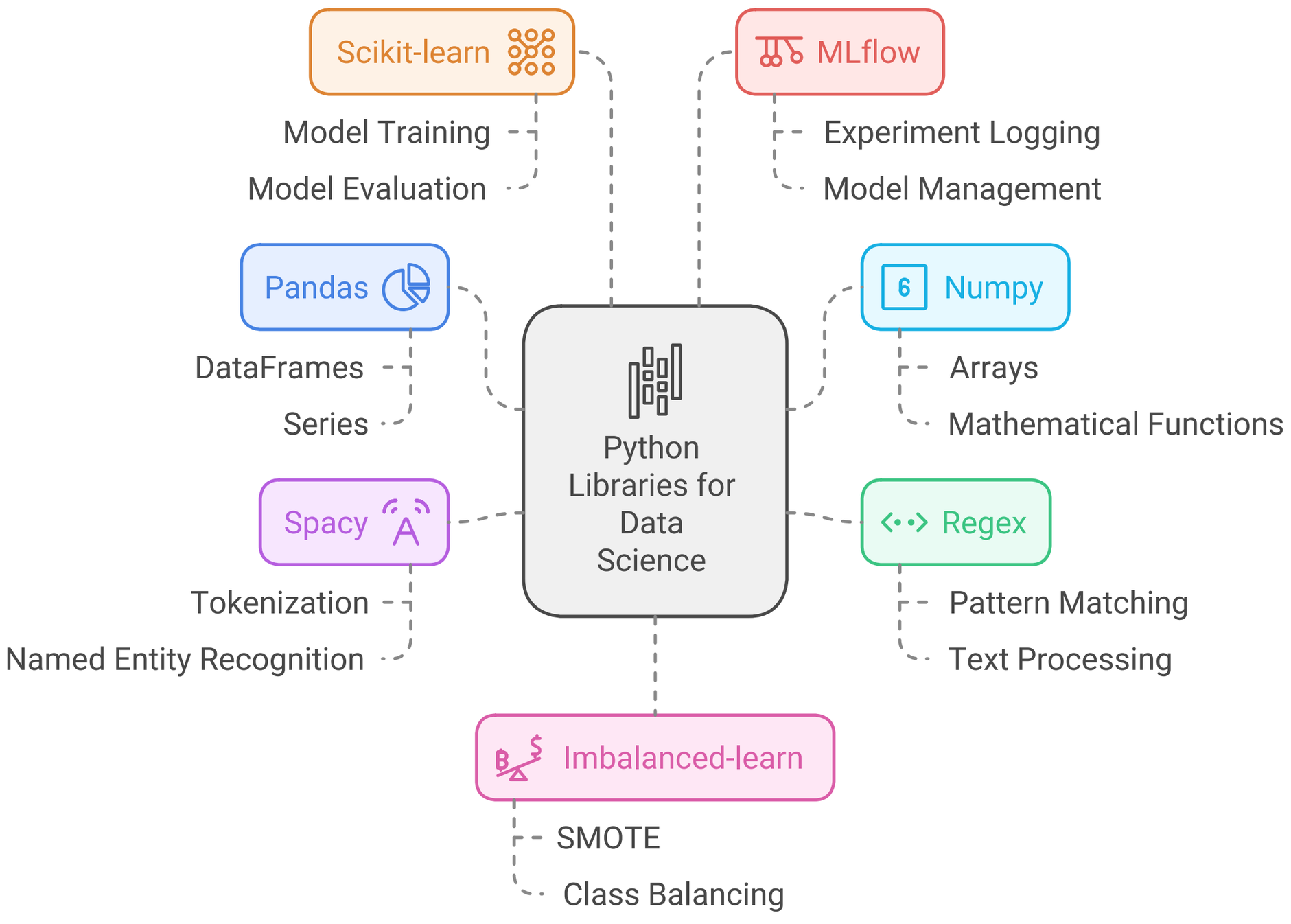
1. **Importing Libraries**:

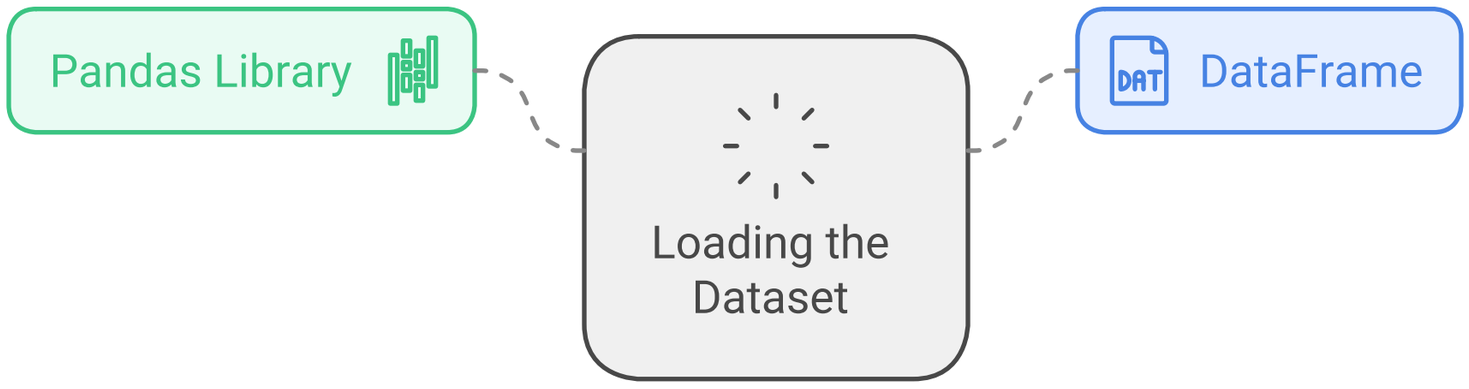
    - **import pandas as pd**: This library is used for data manipulation and analysis.
    - **import numpy as np**: This library provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
    - **import re**: This library provides support for regular expressions.
    - **import spacy**: This library is used for natural language processing tasks.
    - **from sklearn.feature_extraction.text import TfidfVectorizer**: This module from scikit-learn is used for converting text data into numerical feature vectors.

    - **from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score**: These modules from scikit-learn are used for data splitting, cross-validation, and model evaluation.
    - **from sklearn.linear_model import LogisticRegression**: This module from scikit-learn is used for training a Logistic Regression model.
    - **from sklearn.metrics import classification_report, accuracy_score, confusion_matrix**: These modules from scikit-learn are used for evaluating the model's performance.
    - **import mlflow**: This library is used for tracking and managing machine learning experiments.
    - **import mlflow.sklearn**: This module from MLflow is used for logging scikit-learn models.
    - **from imblearn.over_sampling import SMOTE**: This module from the imbalanced-learn library is used for handling class imbalance in the dataset.

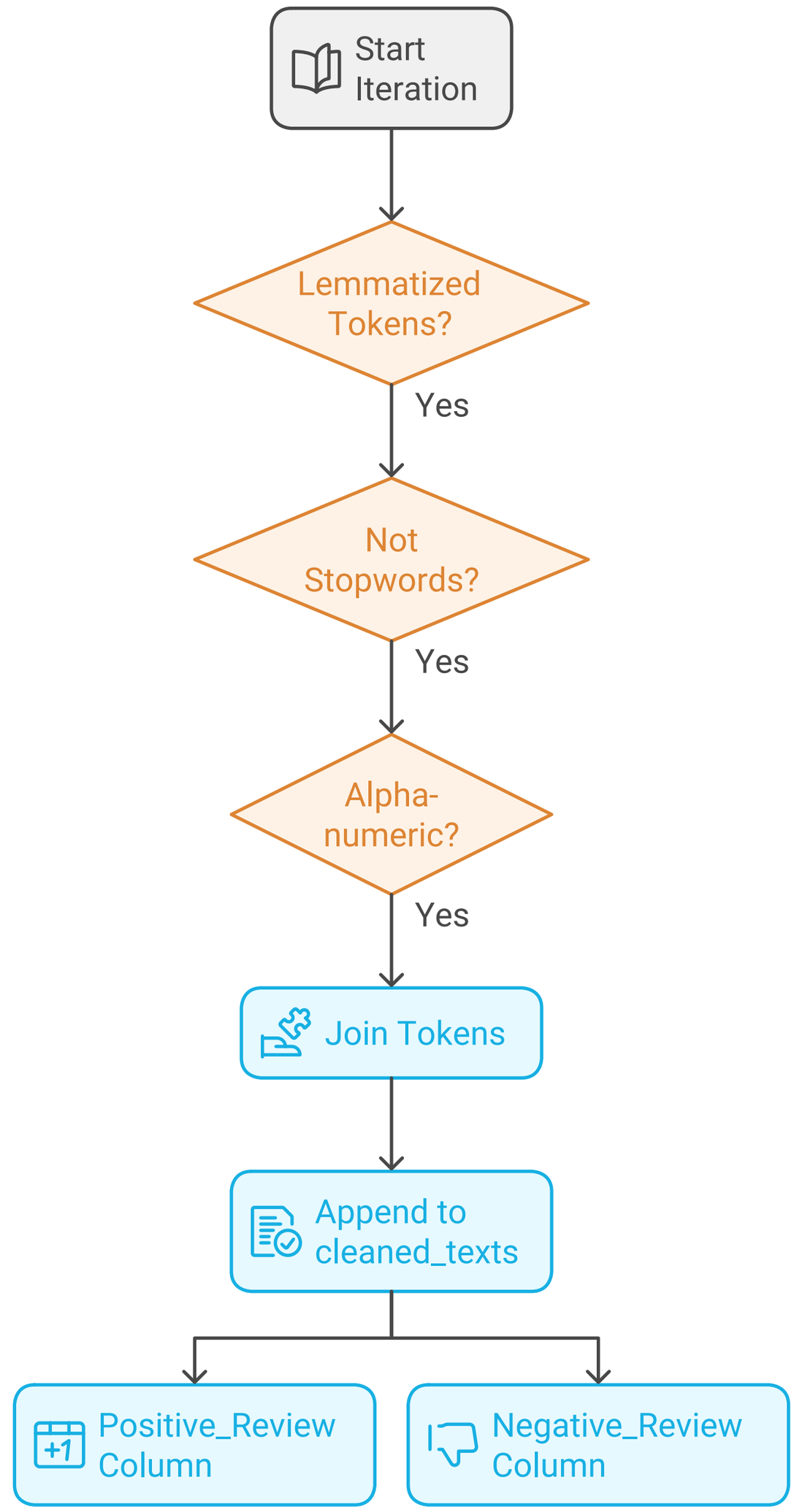


3. **Loading the Dataset**:

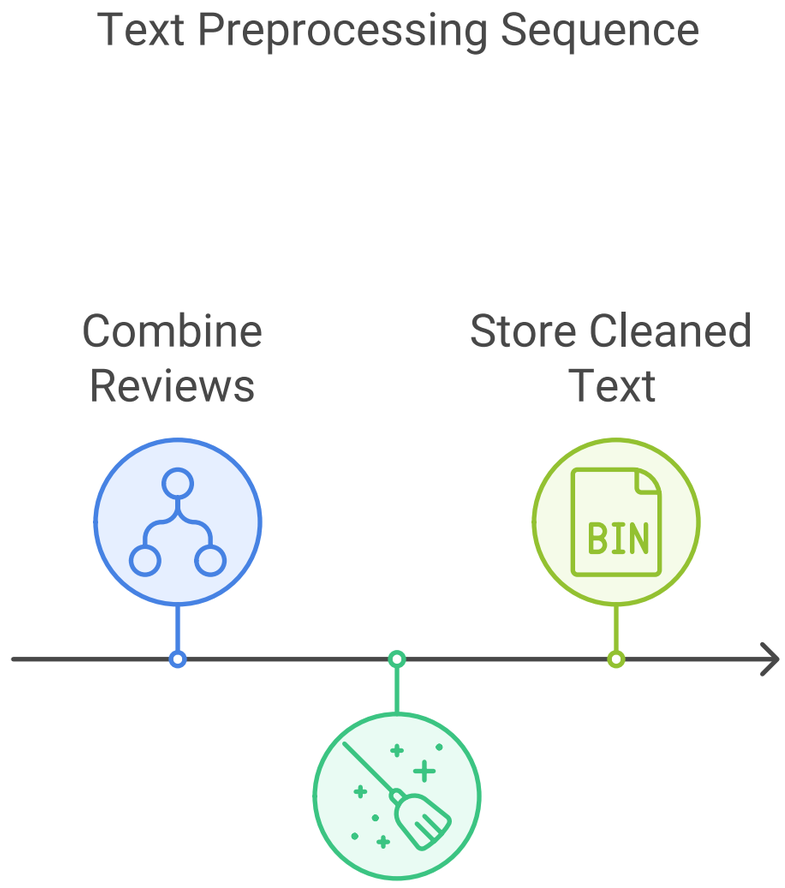
- **df = pd.read_csv('Hotel_Reviews.csv')**: This line reads the Hotel_Reviews.csv file into a pandas DataFrame named **df**.



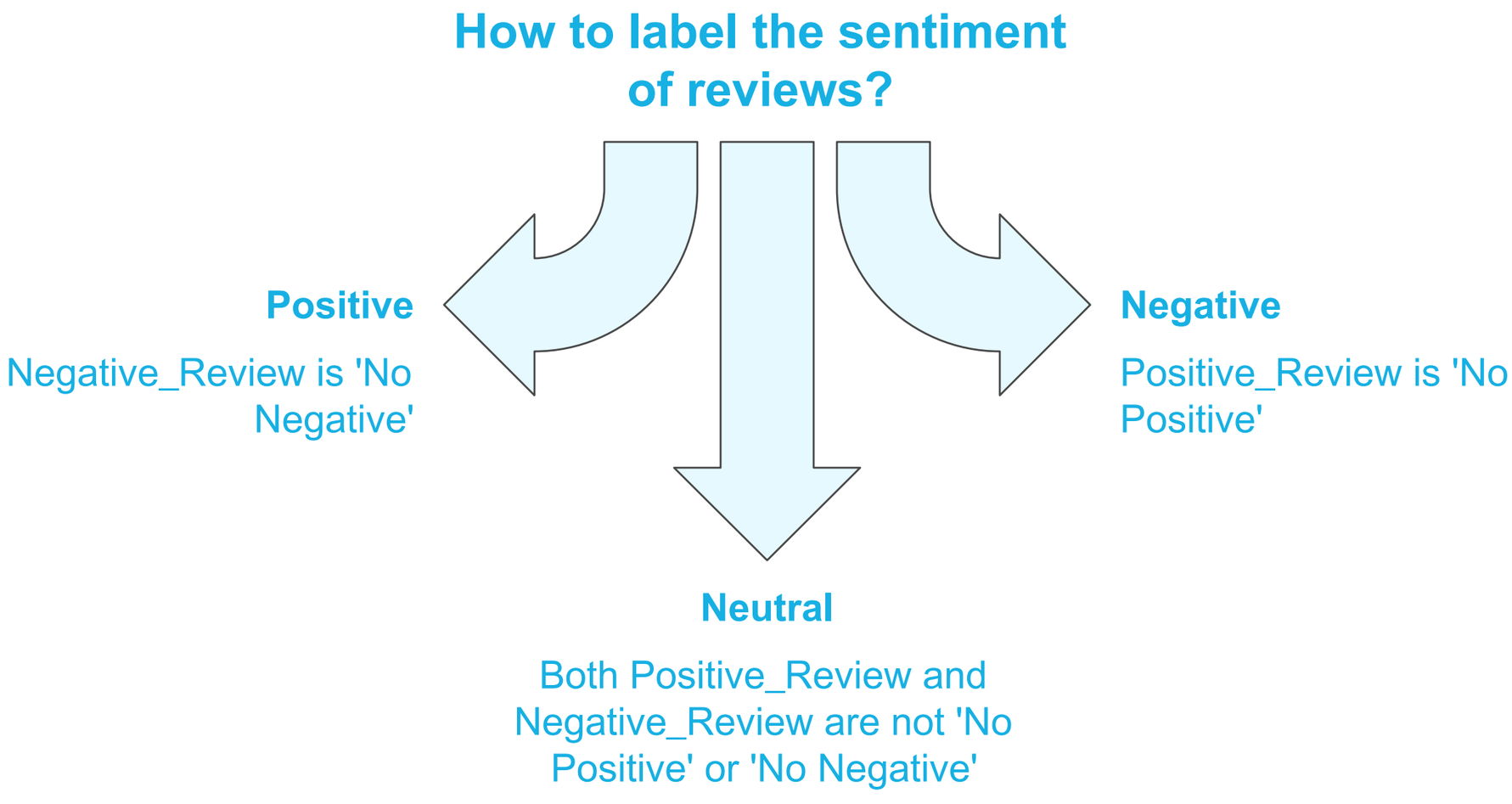
4. **Preprocessing Function**:
    - The **preprocess_texts** function is defined to clean and preprocess the review text. It uses the SpaCy library to perform the following tasks:
        - Iterate through the texts in batches (to improve performance).
        - For each document, extract the lemmatized tokens that are not stopwords and are alpha-numeric.
        - Join the cleaned tokens back into a single string and append it to the **cleaned_texts** list.
    - This function is used to preprocess both the Positive_Review and Negative_Review columns.



5. **Combining Reviews and Applying Preprocessing**:
    - **df['cleaned_review'] = preprocess_texts(df['Positive_Review'] + ' ' + df['Negative_Review'])**: This line combines the Positive_Review and Negative_Review columns, applies the **preprocess_texts** function to the combined text, and stores the cleaned review text in the **cleaned_review** column of the DataFrame.

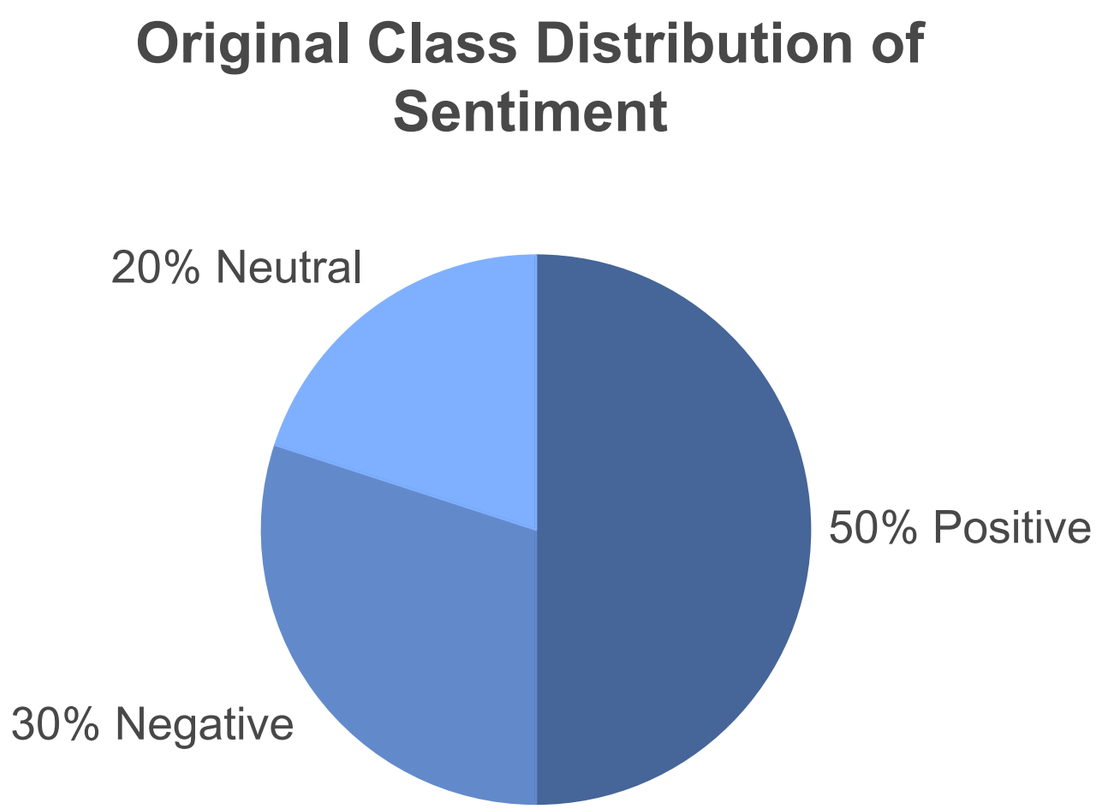


6. **Labeling Sentiment**:
    - The **label_sentiment** function is defined to assign a sentiment label to each review based on the Positive_Review and Negative_Review columns.
    - If the Negative_Review column is 'No Negative', the sentiment is labeled as positive (1).
    - If the Positive_Review column is 'No Positive', the sentiment is labeled as negative (-1).
    - Otherwise, the sentiment is labeled as neutral (0).
    - This function is applied to the DataFrame to create the **Sentiment** column.



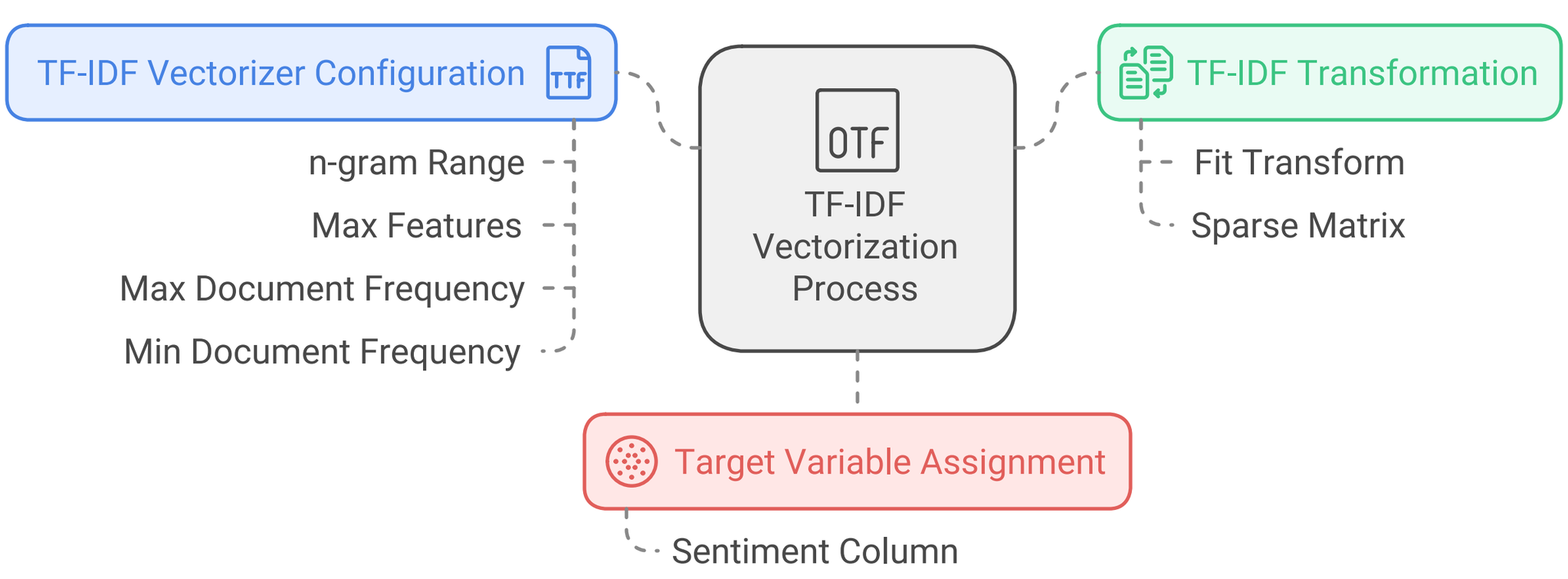
7. **Handling Class Imbalance**:

- **print("Original class distribution:", df['Sentiment'].value_counts())**: This line prints the original class distribution of the **Sentiment** column, which helps identify any imbalance in the data.



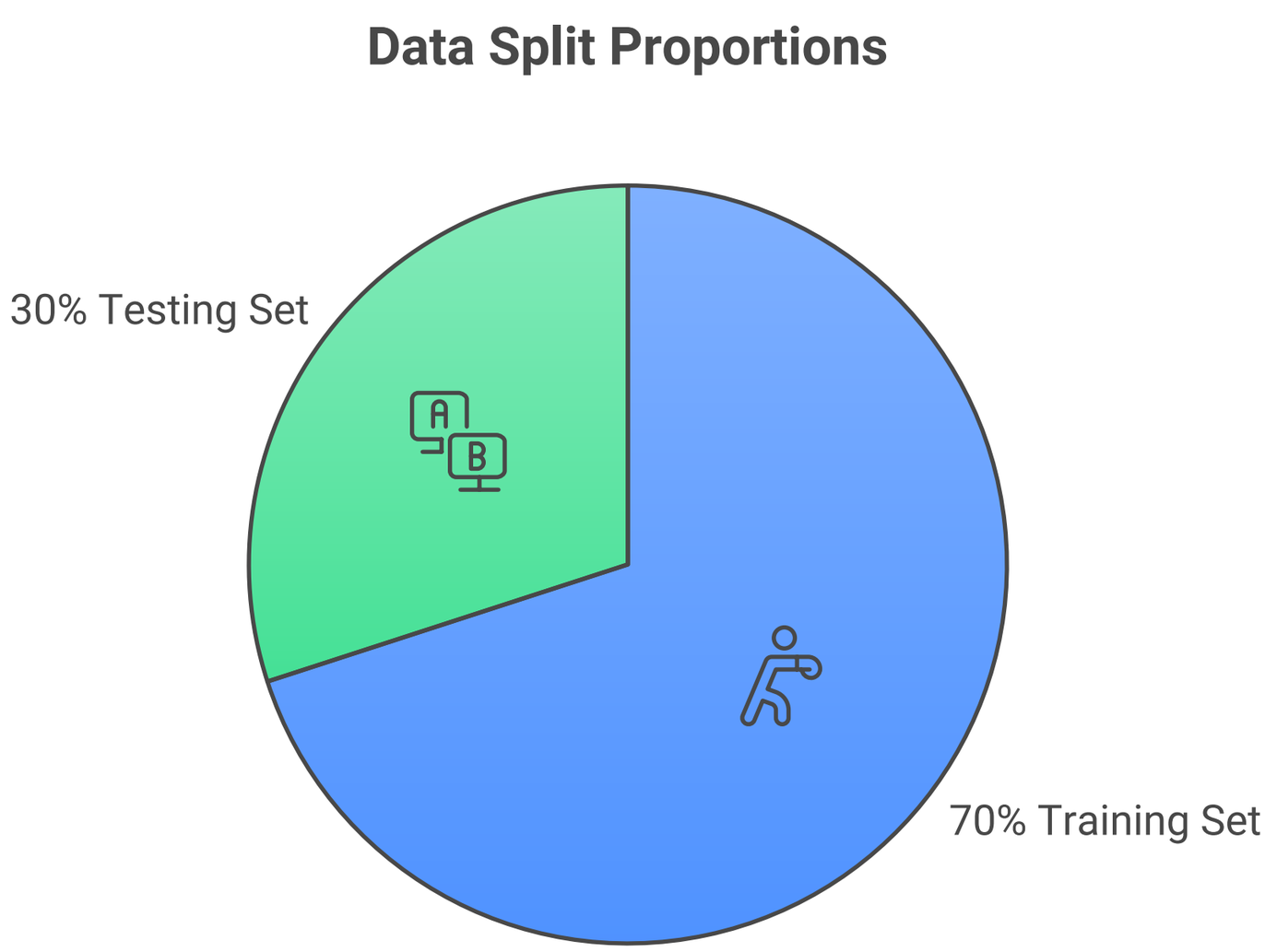
8. **Feature Extraction**:
    - **tfidf = TfidfVectorizer(ngram_range=(1,2), max_features=1500, max_df=0.8, min_df=0.01)**: This creates a TF-IDF Vectorizer object with the following configurations:
        - **ngram_range=(1,2)**: Considers both unigrams and bigrams.
        - **max_features=1500**: Limits the number of features (vocabulary) to 1500.
        - **max_df=0.8**: Ignores terms that appear in more than 80% of the documents.
        - **min_df=0.01**: Ignores terms that appear in less than 1% of the documents.

    - **X = tfidf.fit_transform(df['cleaned_review'])**: This line applies the TF-IDF transformation to the **cleaned_review** column and stores the resulting sparse matrix in **X**.
    - **y = df['Sentiment']**: This line assigns the **Sentiment** column to the **y** variable, which represents the target variable.

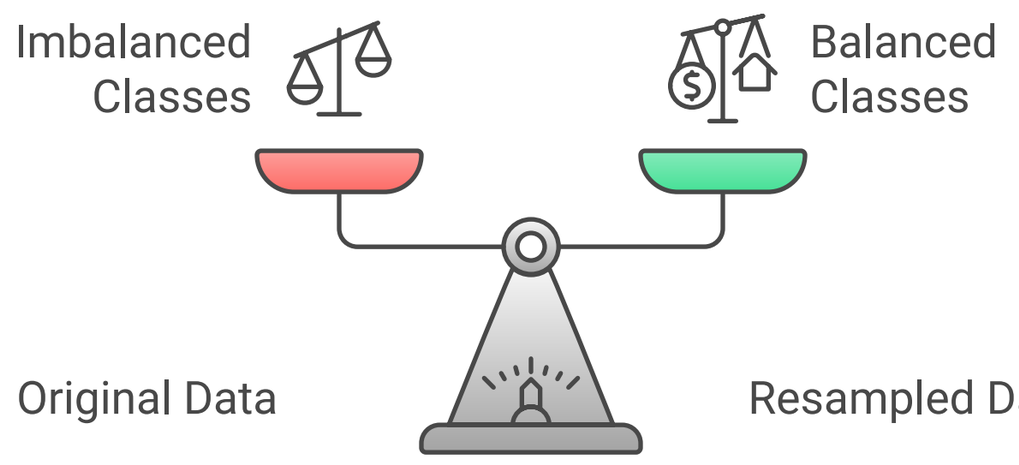


9. **Splitting the Data**:

- **X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)**: This line splits the data into training and testing sets, with a test set size of 30% of the total data. The **stratify=y** parameter ensures that the class distribution is preserved in both the training and testing sets.



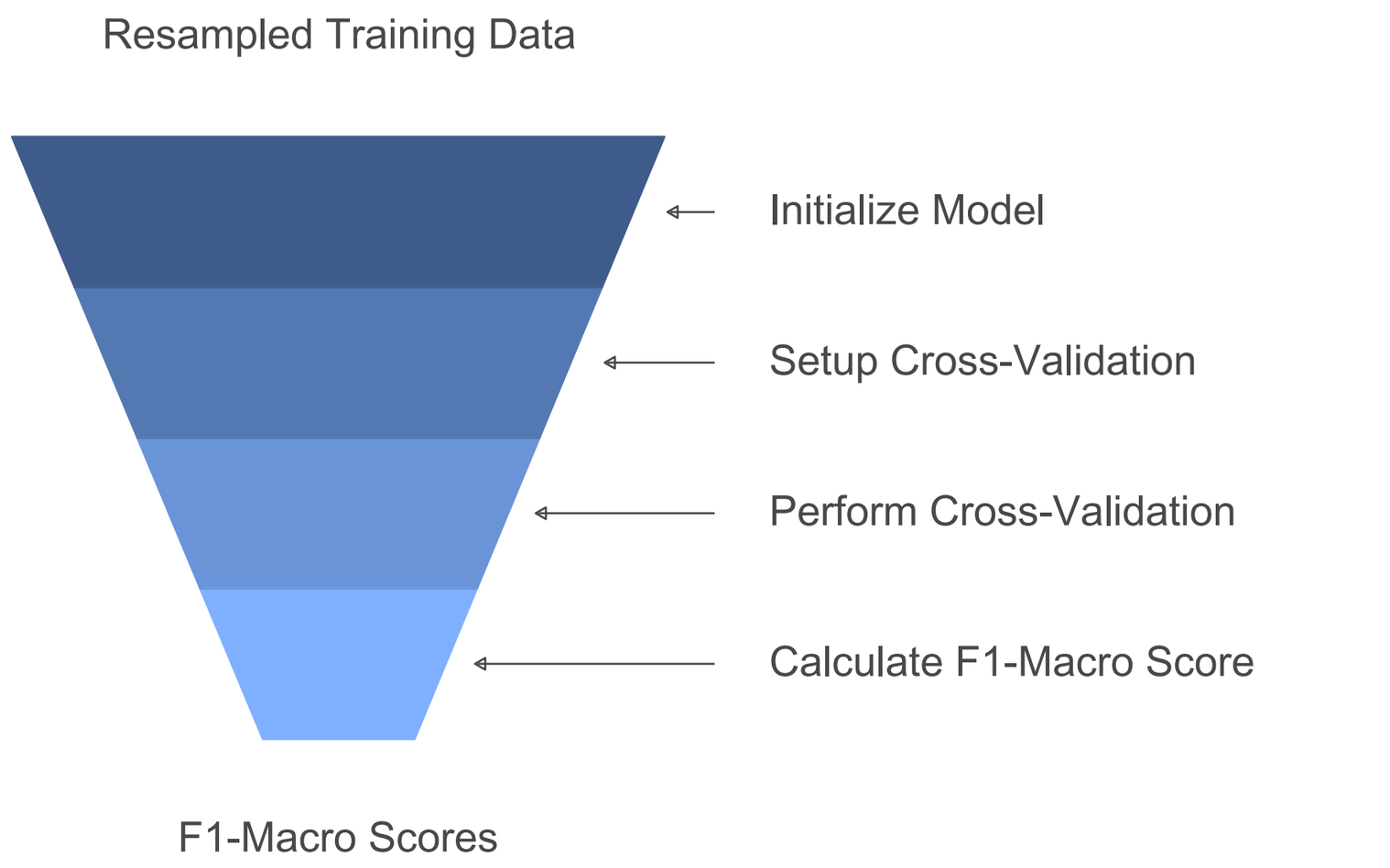
10. **Resampling the Training Data**:
    - **smote = SMOTE(random_state=42)**: This creates a SMOTE (Synthetic Minority Over-sampling Technique) object to handle the class imbalance in the training data.
    - **X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)** : This line applies the SMOTE technique to the training data, generating synthetic samples of the minority class (negative and neutral reviews) to balance the class distribution.
    - **print("Resampled class distribution:", np.bincount(y_train_resampled + 1))**: This line prints the class distribution of the resampled training data.



Visualizing the impact of SMOTE on class distribution.

11. **Training the Logistic Regression Model**:
    - **model = LogisticRegression(max_iter=200)**: This line initializes a Logistic Regression model with a maximum of 200 iterations.

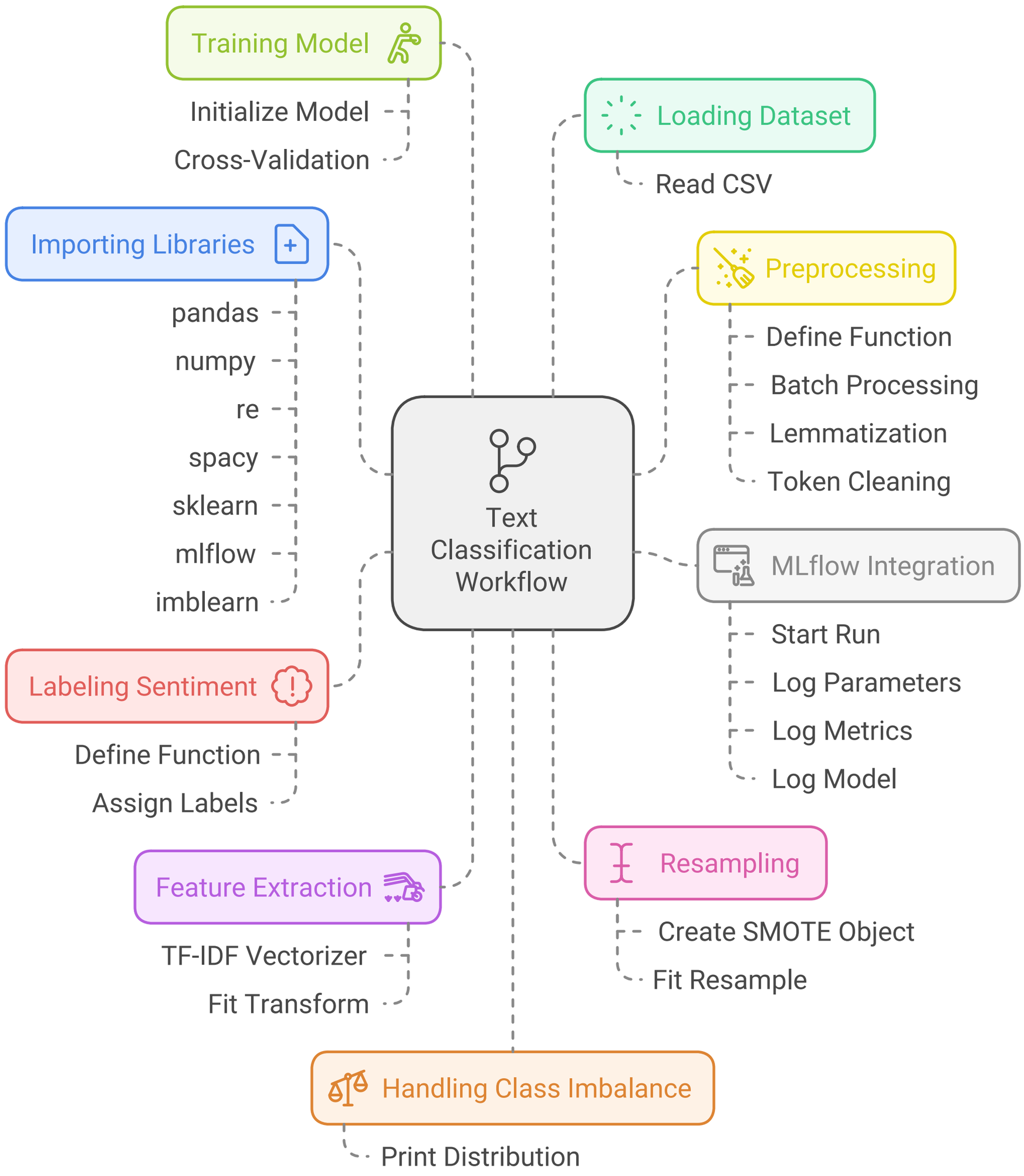
- **cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)**: This creates a 5-fold Stratified K-Fold cross-validation object to evaluate the model's performance.
- **scores = cross_val_score(model, X_train_resampled, y_train_resampled, cv=cv, scoring='f1_macro')**: This line performs 5-fold cross-validation on the resampled training data and calculates the F1-macro score for each fold.



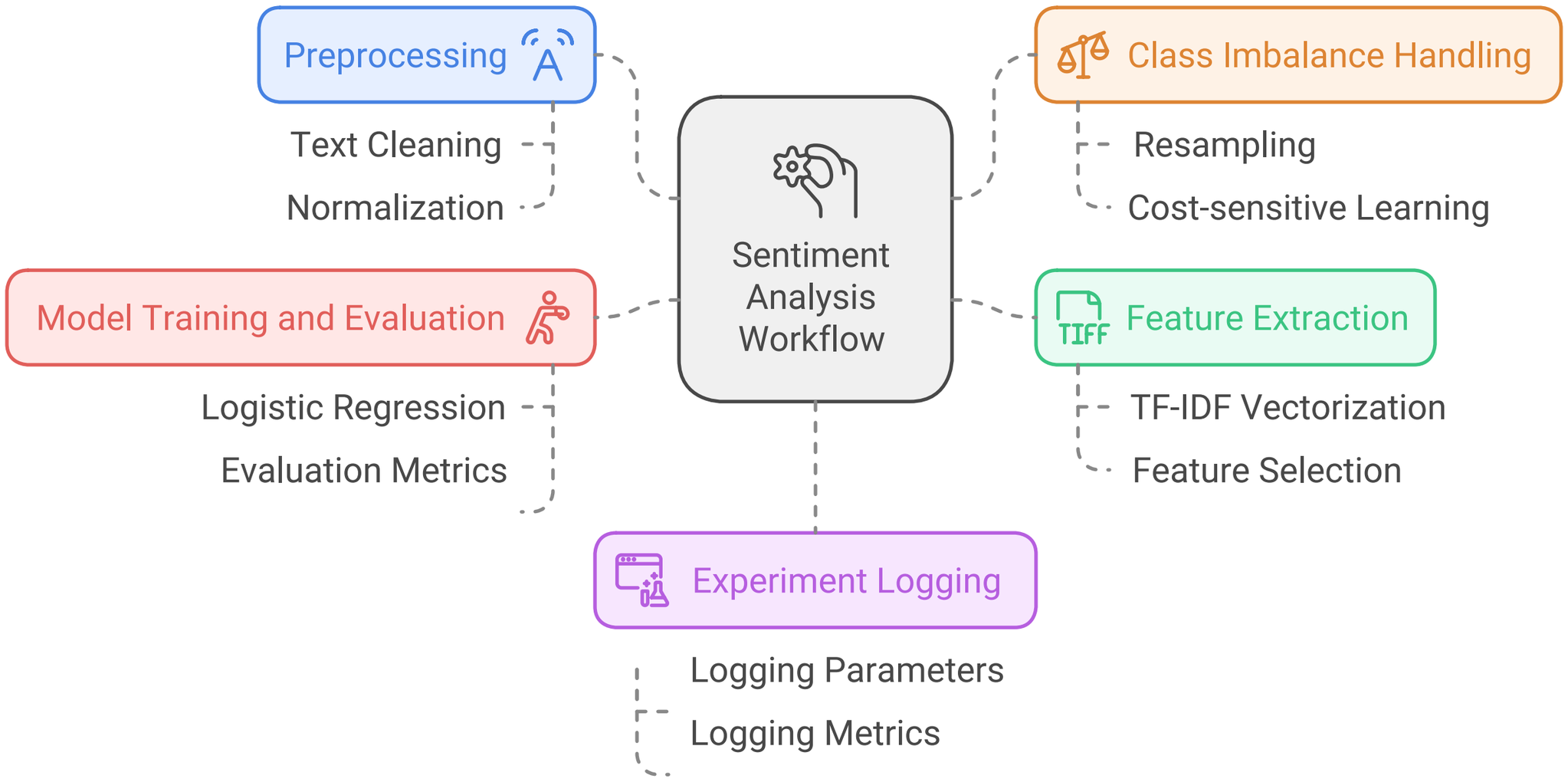
12. **MLflow Integration**:
    - **with mlflow.start_run():**: This block of code starts an MLflow run, which allows for tracking the experiment, logging the model, and recording the relevant parameters and metrics.
    - **model.fit(X_train_resampled, y_train_resampled)**: This line trains the Logistic Regression model on the resampled training data.
    - **y_pred = model.predict(X_test)**: This line uses the trained model to make predictions on the test data.
    - The following lines evaluate the model's performance on the test data:
        - **accuracy = accuracy_score(y_test, y_pred)**: Calculates the accuracy score.
        - **report = classification_report(y_test, y_pred, digits=4)**: Generates a classification report with precision, recall, F1-score, and support for each class.
        - **cm = confusion_matrix(y_test, y_pred)**: Computes the confusion matrix.

    - **mlflow.log_param("max_iter", 200)**: Logs the maximum number of iterations used for the Logistic Regression model.
    - **mlflow.log_param("resampling", "SMOTE")**: Logs the resampling technique used (SMOTE).
    - **mlflow.log_metric("accuracy", accuracy)**: Logs the accuracy metric.
    - **mlflow.log_metric("f1_macro", scores.mean())**: Logs the mean F1-macro score from the cross-validation.
    - **mlflow.sklearn.log_model(model, "model")**: Logs the trained Logistic Regression model.

- **mlflow.sklearn.log_model(tfidf, "tfidf_vectorizer")**: Logs the TF-IDF Vectorizer.



This code performs sentiment analysis on hotel reviews using a Logistic Regression model. It preprocesses the review text, handles class imbalance, extracts features using TF-IDF, trains and evaluates the model, and logs the experiment using MLflow. The detailed explanations provided should help you understand the purpose and functionality of each part of the code.


Sentiment Analysis Workflow
Preprocessing
Text Cleaning
Normalization
Class Imbalance Handling
Resampling
Cost-sensitive Learning
Model Training and Evaluation
Logistic Regression
Evaluation Metrics
Feature Extraction
TF-IDF Vectorization
Feature Selection
Experiment Logging
Logging Parameters
Logging Metrics