Sentiment Analysis
(Random Forest Classifier)

1. Classification and Workflow

The primary goal of a **classification model** is to accurately predict a dependent variable (DV) using one or more independent variables (IVs).

- **Dependent Variable (DV):** The variable being predicted or classified. Commonly referred to as Y, target, output, or response variable. In classification, the DV typically represents distinct categories or classes.
- **Independent Variables (IVs):** The predictors or features used to forecast the DV. Also known as X variables, features, predictors, explanatory variables, or covariates. These variables help explain patterns or correlations with the DV and serve as the model's input.

A standard classification workflow consists of several key steps:

Project Scoping	• Define the business objective, success criteria, and stakeholders. • Understand the broader context and the desired outcome of the model.
Preliminary Quality Assurance	• Align with stakeholders on scope and goals. • Review data access, governance policies, and modeling framework.
Data Profiling	• Understand the structure and quality of your data. • Assess variable types, null values, distributions, and key statistics.
Feature Engineering	• **One-Hot Encoding:** Convert categorical variables into binary indicator variables. • **Scaling:** Normalize numeric features to a common range (e.g., 0–1 or z-scores). • **Log Transformation:** Reduce skewness in data for more stable modeling. • **Discretization:** Bucket continuous variables into discrete bins. • **Date Extraction:** Decompose date/time fields into parts (e.g., day, month, weekday). • **Boolean Logic:** Create interaction features using logical conditions.
Data Splitting	• Divide the dataset into training and testing subsets. • **Training Set:** Used to fit and optimize the model. • **Test Set:** Used to evaluate model performance on unseen data. • This step is essential to prevent overfitting, where the model performs well on training data but poorly on new, unseen data.
Model Training	• Train one or more classification models using the training dataset. • Apply the trained models to the test data to assess their generalization performance.
Model Selection & Hyperparameter Tuning	• Choose the best-performing model based on evaluation metrics (e.g., accuracy, ROC-AUC, F1-score). Fine-tune model hyperparameters to optimize performance and reduce bias or variance. • Plan for monitoring and re-tuning the model over time to handle concept drift (changes in data patterns).

A **Confusion Matrix** is a tabular representation of actual vs. predicted classifications on a test dataset. It is one of the most widely used tools for evaluating the performance of classification models.

- **Accuracy:** Measures the overall correctness of the model. When the correct predictions of both classes are equally important
- **Precision:** Indicates the proportion of predicted positives that were correct. When false positives are more harmful than false negatives
- **Recall:** Measures how well the model identifies all actual positives. When false negatives are more harmful than false positives

2. Decision Trees and Random Forests

A **Decision Tree** is a supervised machine learning algorithm that uses a sequence of binary (true/false) rules to classify data. It is suitable for both binary and multiclass classification problems.

- Each **internal node** in the tree represents a decision rule based on one independent variable (IV).
- Each **split** divides the dataset into two groups, ideally separating observations to favor one class in each subset.
- The **tree** continues to split the data recursively, aiming to create purer subsets concerning the target variable.

Key considerations in building decision trees:

Entropy	• Measures the impurity or uncertainty in a dataset. • A perfectly homogeneous set has **entropy = 0**, while maximum uncertainty has higher entropy.
Information Gain	• The reduction in entropy after a dataset is split on an attribute. It helps determine which variable provides the most effective split. • The process of selecting splits is automated. The algorithm evaluates multiple variables and thresholds to maximize information gain.
Splitting	• Variables that better reduce entropy (uncertainty) are preferred for splitting • For continuous variables, the algorithm tests various thresholds and chooses the split point that maximizes information gain.
Stopping	• Excessive splitting can lead to overfitting, where the model fits the training data too well but performs poorly on unseen data. • This is controlled using hyperparameters, such as: **maximum tree depth, minimum samples per leaf, and minimum samples required to split a node**
Root Node	• First split may not be the best overall. A locally optimal split may not lead to the most accurate model globally. • To address this, ensemble methods like **Random Forests** are often used.

A **Random Forest** is an ensemble learning method that combines multiple decision trees to produce a more robust and accurate prediction.

- Each tree is trained on a random subset of the training data using bootstrapping. At each node, a random subset of features is considered for splitting. The model combines the predictions from all trees.
- **Advantages:** Reduces overfitting (by averaging multiple models, variance is minimized without significantly increasing bias), improves accuracy (ensemble methods generally outperform individual models, especially on complex or noisy datasets), handles missing values, and maintains high performance even when some data is uninformative or irrelevant.
- For **classification**, the majority vote is used. For **regression**, the average of all tree predictions is taken.
- Hyperparameter tuning involves systematically adjusting these settings to enhance model performance. Effective tuning can improve prediction accuracy and help prevent overfitting or underfitting. Common optimization techniques include **grid search** (which exhaustively tests all combinations of specified hyperparameter values), **random search** (which tests a random subset of combinations for quicker results), and **Bayesian optimization** (which utilizes probabilistic models to intelligently select hyperparameters).

Breakdown of hyperparameters:

max_depth	• Tree complexity control • Limits tree growth to reduce overfitting
max_features	• Feature selection per split • Introduces diversity among trees
min_samples_leaf	• Minimum samples at leaf nodes • Larger sample size increases generalization
n_estimators	• Number of trees • Fewer trees are faster but less stable

3. Sentiment Analysis

Sentiment Analysis is a Natural Language Processing (NLP) technique used to identify and extract the emotional tone conveyed in text-based data. It is most commonly implemented as a classification task that categorizes text into sentiment classes such as positive, negative, or neutral.

- While rooted in NLP, sentiment analysis typically involves **transforming unstructured text into structured numerical inputs**, enabling machine learning models to interpret and analyze the data effectively.
- Sentiment models often employ a Bag of Words (BoW) approach, where text is converted into numerical representations based on the frequency or presence (1/0) of specific keywords.

Different aspects of sentiment analysis:

Text Preprocessing and Quality Assurance	• Removing punctuation, capitalization, and special characters • Correcting spelling and grammatical errors • Ensuring proper text encoding (e.g., UTF-8) • Applying lemmatization or stemming to reduce words to their root form • Removing stop words (common words such as "a", "the", "is", "or", "of", etc.) that do not carry sentiment
Feature Engineering: Bag of Words	• **Tokenization:** Split text into individual words or tokens • **Vectorization:** Create features (columns) representing each word, using binary flags or term frequencies
Feature Engineering: TF-IDF Vectorization	• **Tokenization:** Split cleaned text into individual words (tokens) • **Term Frequency (TF):** Measures how frequently a word appears in a specific document • **Inverse Document Frequency (IDF):** Scale down the importance of commonly used words that appear across many documents • **TF-IDF Score:** Multiply TF and IDF to highlight terms that are both frequent in a document and rare across the corpus • **Vector Representation:** Convert each document into a numerical vector based on the TF-IDF scores of its terms
Language Nuances	• Use larger and more diverse training datasets with accurately labeled sentiment • Enhance preprocessing and feature engineering with techniques such as n-grams, part-of-speech tagging, or dependency parsing • Consider advanced models like **transformers (e.g., BERT),** which understand context and semantic relationships more deeply than traditional vectorization techniques

A standard sentiment analysis workflow consists of several key steps:

Data	• **Location:** https://github.com/f2005636/sentiment-analysis • **File:** input.csv (0-negative review, 1-positive review)
Clean the Text	• Convert to lowercase • Remove numbers and punctuation • Remove extra spaces • Remove common stop words (like "the", "and", "is") • Apply stemming (e.g., "running" becomes "run")
Convert Text to Numbers	• The cleaned text is converted into numerical features using **TF-IDF** • It is a technique that highlights important words
Model Selection	• Grid Search is used to find the best hyperparameters for a **Random Forest classifier** based on the AUC-ROC score. • The model is trained using the selected hyperparameters. • The model performance is evaluated using a **confusion matrix** and **AUC-ROC**