Project Report

Intern Name: Ahmed Islam

Department: Artificial Intelligence

Organization: IR Solutions

Designation: AI Intern

Week # 01

Submitted to: Asad Ali

Submission Date: 18-July-2025

Report Type: Combined Tasks Documentation

Theoretical Understanding

Working with Numerical data

Integers and floating-point numbers are the basically Numerical data.

Feature Vectors:

Model digests an array of floating-point values that are called features vectors.

A feature vector is a list of numerical values that represent the measurable properties or characteristics of an object or data point, used as input for machine learning models.

Feature vectors transform raw data into a format that machine learning algorithms can understand and utilize.

Feature Engineering:

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used by machine learning models to improve their performance.

There are two techniques that we use in feature Engineering:

1. **Normalization:** Converting the numerical values in the standard range
2. **Binning:** also refer as a bucket. Converting the numerical values in a bucket of range.
3. **Quantile buckets:** dividing a dataset in the equal example of bins.

Before creating feature vectors, we recommend studying numerical data in two ways:

- Visualize your data in plots or graphs.
- Get statistics about your data.

Standard deviation: is a measure of how spread out a set of values is, relative to its mean (average).

Outliers: are the values that are distinct from most of the other values in a feature label.

Clipping technique is used to handle the outlier:

Replacing values that are too high and too low with boundary values of percentiles.

Also known as capping and truncating.

There are three popular techniques of normalization:

1. linear scaling
2. Z-score scaling
3. log scaling

Normalization technique	Formula	When to use
Linear scaling	$\boxed{x' = \frac{x - x_{min}}{x_{max} - x_{min}}}$	When the feature is mostly uniformly distributed across range. **Flat-shaped**
Z-score scaling	$x' = \frac{x - \mu}{\sigma}$	When the feature is normally distributed (peak close to mean). **Bell-shaped**
Log scaling	$x' = log(x)$	When the feature distribution is heavy skewed on at least either side of tail. **Heavy Tail-shaped**
Clipping	If $x > max$, set $x' = max$ If $x < min$, set $x' = min$	When the feature contains extreme outliers.

Linear scaling

Linear scaling (more commonly shortened to just scaling) means converting floating-point values from their natural range into a standard range—usually 0 to 1 or -1 to +1.

Linear scaling is a good choice when all of the following conditions are met:

- The lower and upper bounds of your data don't change much over time.
- The feature contains few or no outliers, and those outliers aren't extreme.
- The feature is approximately uniformly distributed across its range. That is, a histogram would show roughly even bars for most values.

Use the following formula to scale to the standard range 0 to 1, inclusive:

$$x' = (x - x_{min})/(x_{max} - x_{min})$$

where:

- x' is the scaled value.
- x is the original value.
- x_{min} is the lowest value in the dataset of this feature.
- x_{max} is the highest value in the dataset of this feature.

Z-score scaling

A Z-score is the number of standard deviations a value is from the mean.

Representing a feature with **Z-score scaling** means storing that feature's Z-score in the feature vector. For example, the following figure shows two histograms:

- On the left, a classic normal distribution.
- On the right, the same distribution normalized by Z-score scaling.

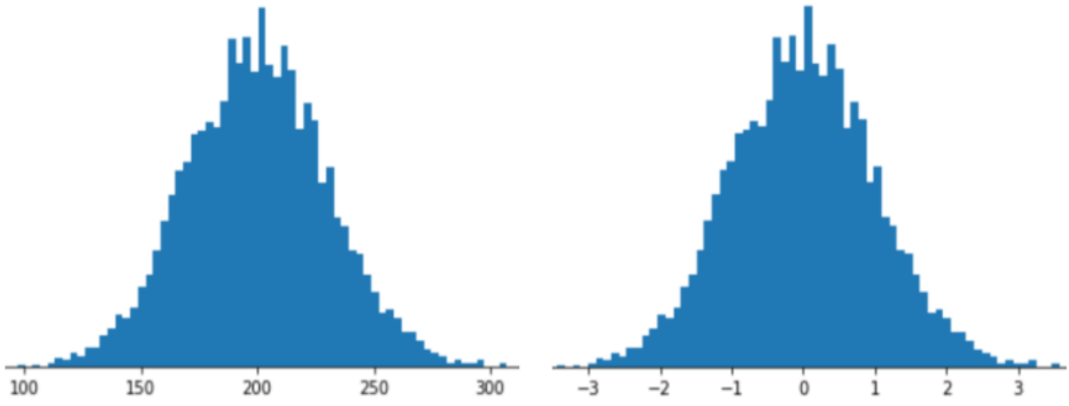


Figure 4. *Raw data (left) versus Z-score (right) for a normal distribution.*

Z-score scaling is also a good choice for data like that shown in the following figure, which has only a vaguely normal distribution.

Use the following formula to normalize a value, x, to its Z-score:

$$x' = (x - \mu)/\sigma$$

where:

- x' is the Z-score.
- x is the raw value; that is, x is the value you are normalizing.
- μ is the mean.
- σ is the standard deviation.

Log scaling

Log scaling computes the logarithm of the raw value. In theory, the logarithm could be any base; in practice, log scaling usually calculates the natural logarithm (ln).

Use the following formula to normalize a value, x, to its log:

$$x' = ln(x)$$

where:

- x' is the natural logarithm of x.
- original value = 54.598

Log scaling is helpful when the data conforms to a power law distribution.

Binning:

Binning (also called discretization) is a data preprocessing technique where continuous numerical values are grouped into discrete intervals or bins.

This is often used to reduce the effects of minor observation errors and to simplify models.

Binning (also called bucketing) is a feature engineering technique that groups different numerical subranges into bins or buckets. In many cases, binning turns numerical data into categorical data.

Binning is a good alternative to scaling or clipping when either of the following conditions is met:

- The overall linear relationship between the feature and the label is weak or nonexistent.
- When the feature values are clustered.

Quantile Bucketing:

Quantile binning groups data such that each bin contains (approximately) the same number of data points. It is based on the distribution of the data, not the range.

Scrubbing

Scrubbing refers to the process of cleaning raw data to improve its quality, consistency, and usability for analysis or modeling.

It is also called data cleansing, data cleaning, or data preprocessing.

Purpose of Scrubbing

To detect and fix:

- Missing values
- Outliers
- Inconsistent formats
- Invalid entries
- Duplicate records
- Typos and spelling mistakes

Scrubbing ensures that garbage in, garbage out (GIGO) doesn't happen — poor quality input data leads to poor model performance.

Polynomial transforms

When the ML practitioner has domain knowledge suggesting that one variable is related to the square, cube, or other power of another variable, it's useful to create a synthetic feature from one of the existing numerical features.

Working with categorical data

Categorical data has a specific set of possible values. For example:

- The different species of animals in a national park
- The names of streets in a particular city
- Whether or not an email is spam
- The colors that house exteriors are painted
- Binned numbers, which are described in the Working with Numerical Data module

Encoding

Encoding is the process of converting information into a format that can be used by computers, such as converting text into a sequence of numbers. This is necessary because computers can only process numerical data, and encoding allows us to represent and store various types of information in a way that computers can understand and manipulate.

Vocabulary and one-hot encoding

When a categorical feature has a low number of possible categories, you can encode it as vocabulary. With a vocabulary encoding, the model treats each possible categorical value as a separate feature. During training, the model learns different weights for each category.

One hot-encoding: One-Hot Encoding is a technique to convert categorical values into a binary vector format so that they can be used in ML models.

What is a Sparse Feature in Machine Learning?

A sparse feature is a feature (or input column) where most of the values are zero (or missing/empty). It's common in high-dimensional data, especially with categorical variables, text, and recommendation systems.

Categorical data: Common issues

Human raters

Data manually labeled by human beings is often referred to as gold labels, and is considered more desirable than machine-labeled data for training models, due to relatively better data quality.

Inter-Rater Agreement

Inter-Rater Agreement (IRA) refers to the degree of consistency or agreement among multiple human annotators (raters) who evaluate or label the same data.

Machine raters

Machine-labeled data, where categories are automatically determined by one or more classification models, is often referred to as silver labels. Machine-labeled data can vary widely in quality. Check it not only for accuracy and biases but also for violations of common sense, reality, and intention.

High Dimensionality

High dimensionality refers to datasets that have a very large number of features (variables) compared to the number of samples.

Too many columns not enough rows

Feature Cross

Feature crosses are created by crossing (taking the Cartesian product of) two or more categorical or bucketed features of the dataset. Like polynomial transforms, feature crosses allow linear models to handle nonlinearities. Feature crosses also encode interactions between features.

Polynomial transforms typically combine numerical data, while feature crosses combine categorical data.

Difference between Scaling and Normalization

Aspect	**Scaling (Standardization)**	**Normalization (Min-Max)**
Output range	Can be **any range** (mean = 0, std = 1)	**Fixed range**, usually [0, 1]
Handles outliers	✅ More robust	❌ Sensitive to outliers
Use case	Good for **ML algorithms** like SVM, logistic regression	Good for **neural networks**, distance-based models
Method	Z-score standardization	Min-max rescaling

Dataset, Generalization and Overfitting

Dataset: collection of relevant examples.

Imputation is the process of generating well-reasoned data, not random or deceptive data. Be careful: good imputation can improve your model; bad imputation can hurt your model.

Labels

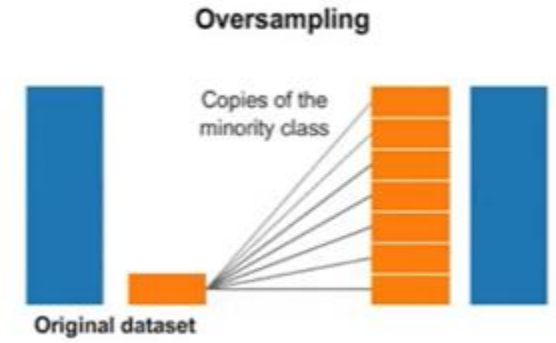
Labels are the target values or outputs that a machine learning model tries to predict.

- **Direct Label:** A direct label is a label that directly measures the outcome you want to predict.
- **Proxy Label:** A proxy label is a substitute for the real outcome — used when the true label isn't available, delayed, or hard to collect.

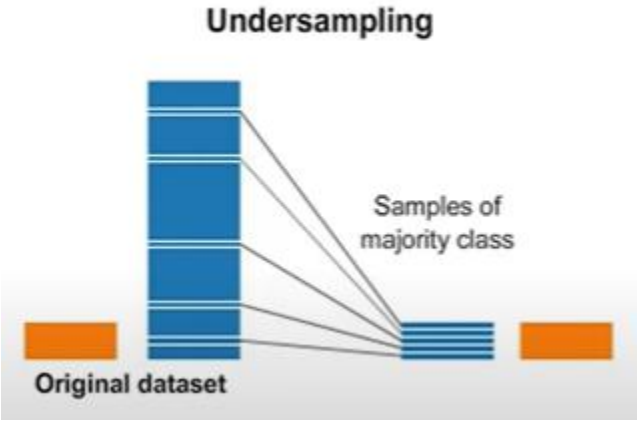
Imbalanced data

In imbalanced datasets, models tend to favor the majority class, because it dominates the data.

- **Downsampling:** Downsampling means reducing the number of majority class samples to match the minority class.
- **Upweighting:** Upweighting means keeping all your data, but telling the model to pay more attention to the minority class by increasing its loss function weight.

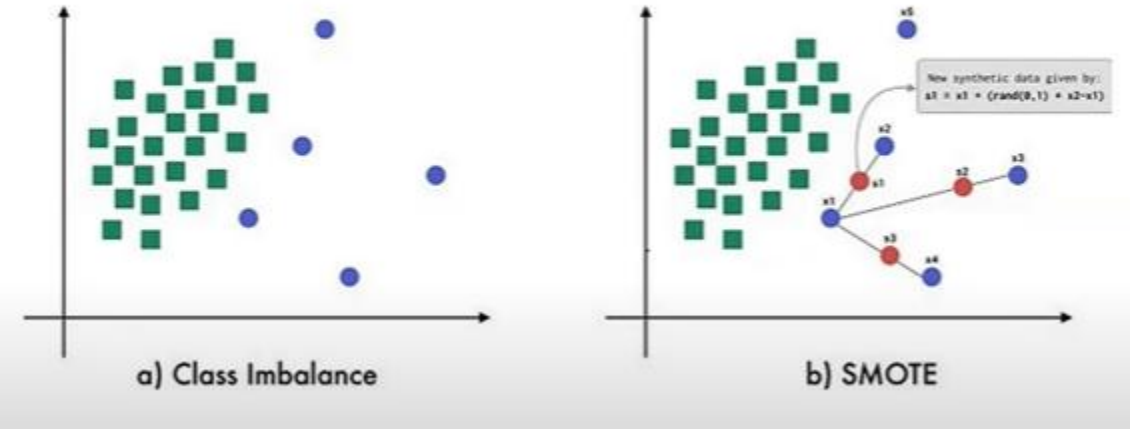


- **Undersampling**



- **SMOKE(synthetic Minority oversampling Technique):** This technique also do a oversampling like increase the number of minority classes for the balance.

3. SMOTE popular → Oversampling → min → upsample
03 May 2024 01:20



Generate new data points instead of duplication.

5 KNN

- Train a KNN on minority class observations - find each observation's 5 closest neighbours

To create the new synthetic data:

- Select examples from the minority class at random
- Select a neighbour of each example at random (for the interpolation)
- Extract a random number between 0 and 1
- Calculate the new examples as = original sample - factor * (original sample - neighbour)
- The final dataset consists of the original dataset + the newly created examples

Overfitting & Underfitting

Overfitting: Overfitting happens when the model learns the training data too well — even the noise or irrelevant patterns.

Underfitting: Underfitting happens when the model is too simple to capture the patterns in the data.

Regularization

Regularization is a technique used in machine learning to prevent overfitting by penalizing model complexity.

Types of Regularization

- **L1 Regularization (Lasso):** Adds the absolute value of the weights to the loss.
- **L2 Regularization (Ridge):** Adds the squared value of the weights to the loss.
- **Elastic Net:** A combination of L1 and L2

Regularization Rate

The regularization rate, often denoted by λ (lambda) or α (alpha), is a hyperparameter that controls how much regularization is applied to the model.

Task 1

Customer Churn Prediction Using Logistic Regression

Question No 1: How many rows and columns are in the dataset? List the column names.

Answer:

- Rows: 28,382
- Columns: 21
- Column Names: 'customer_id', 'vintage', 'age', 'gender', 'dependents', 'occupation', 'city', 'customer_nw_category', 'branch_code', 'days_since_last_transaction', 'current_balance', 'previous_month_end_balance', 'average_monthly_balance_prevQ', 'average_monthly_balance_prevQ2', 'current_month_credit', 'previous_month_credit', 'current_month_debit', 'previous_month_debit', 'current_month_balance', 'previous_month_balance', 'churn'

Question No 2: Identify missing values. How were missing values handled for gender, dependents, occupation, and city?

```
missing = df_churn[['gender', 'dependents', 'occupation', 'city']].isnull().sum()
print(missing)
```

Column	Data Type	Strategy Used
gender	Object	Filled with **mode**
occupation	Object	Filled with **mode**
dependents	Float	Filled with **mode**
City	Float	Filled with **mode**
days_since_last_transaction	Float	Filled with **Mean**

3. Apply one-hot encoding to the occupation column. Why is one-hot encoding preferred over label encoding for this feature?

df_churn = pd.get_dummies(df_churn, columns=['occupation'], drop_first=True)

One-hot encoding is preferred for nominal (non-ordered) categorical features like occupation because:

- It prevents the model from assuming any ordinal relationship between encoded values (which would happen with label encoding).
- Each category gets its own binary column

4. Use StandardScaler and log transformation for numerical columns. Explain why log transformation was applied to balance features.

- **Log transformation** reduces skewness and compresses extreme values (outliers), making the data more normally distributed, which benefits many models like logistic regression.
- **StandardScaler** ensures all numeric features have mean 0 and standard deviation 1, aiding convergence and performance.

5. Split the dataset into training (2/3) and validation (1/3) with stratify=y. Why is stratification important?

```
# Separating features (X) and target (y)
X = df_churn_copy.drop('churn', axis=1)  # Features
y = df_churn_copy['churn']  # Target variable

# random_state is set for reproducibility of the split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
                                                    random_state=42, stratify=y)
```

Stratification ensures that both training and validation sets have the same class distribution as the original dataset. This is crucial in imbalanced classification problems to avoid biasing the model during training or evaluation.

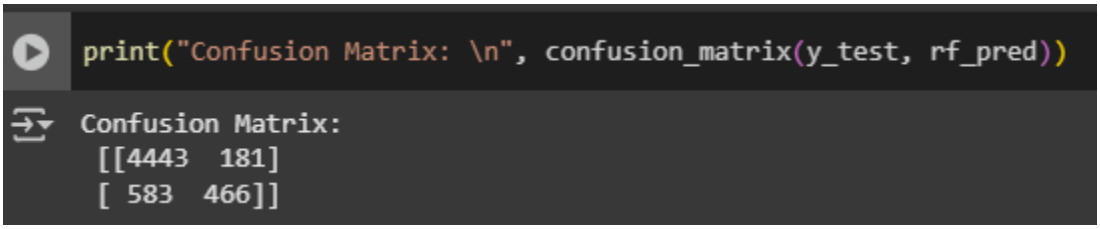
6. Train a logistic regression model on the baseline features. List the selected features and explain their relevance based on EDA insights.

```
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train, y_train)
```

7. What does the AUC indicate about model performance?

- The AUC (Area Under Curve) measures how well the model distinguishes between classes.
- An AUC of 1.0 = perfect, 0.5 = random guessing.
- For example, an AUC of 0.88 means the model is very good at separating churn vs. non-churn.

8. Generate and interpret the confusion matrix. What are the implications of the recall score?



- The AUC (Area Under Curve) measures how well the model distinguishes between classes.
- An AUC of 1.0 = perfect, 0.5 = random guessing.

9. Discuss how overfitting was addressed in the model. Suggest one method to improve the recall score.

Overfitting Prevention:

- Used log transformation to reduce extreme outlier
- Applied regularization in LogisticRegression
- Maintained feature scaling and stratified train-test split

To improve recall:

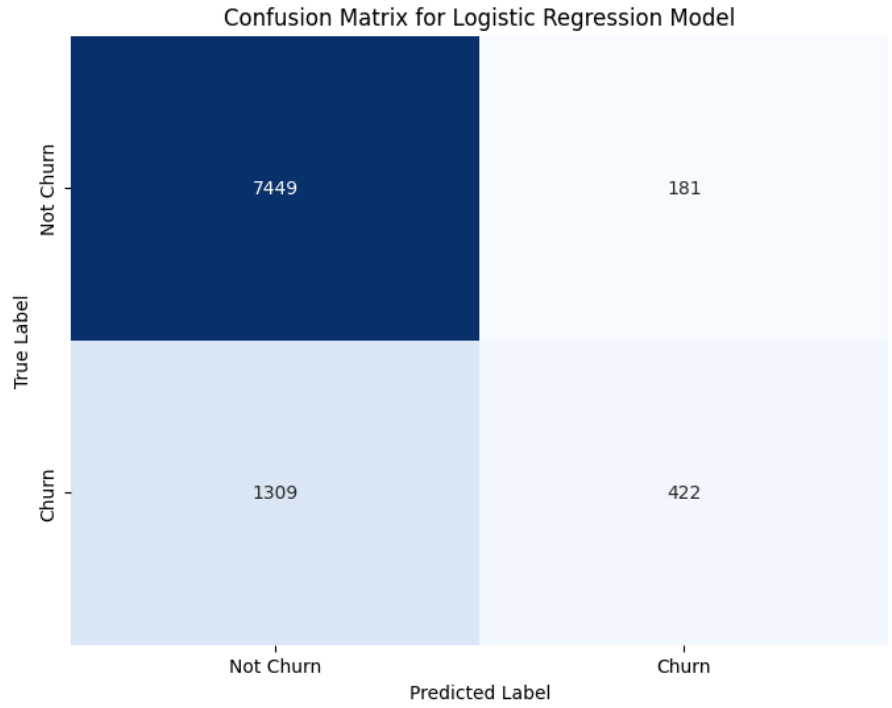
- Use class weighting (class_weight='balanced') or sample weighting
- Try more robust models like XGBoost
- Use threshold tuning or SMOTE to improve minority class detection

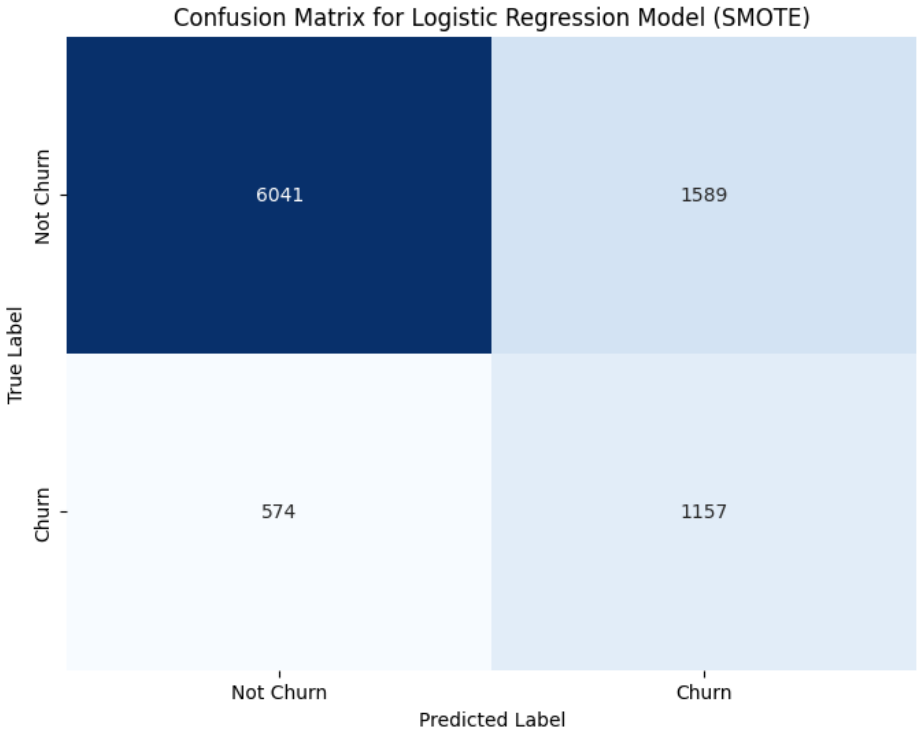
10. Save the model using pickle. Why is this step important for reproducibility?

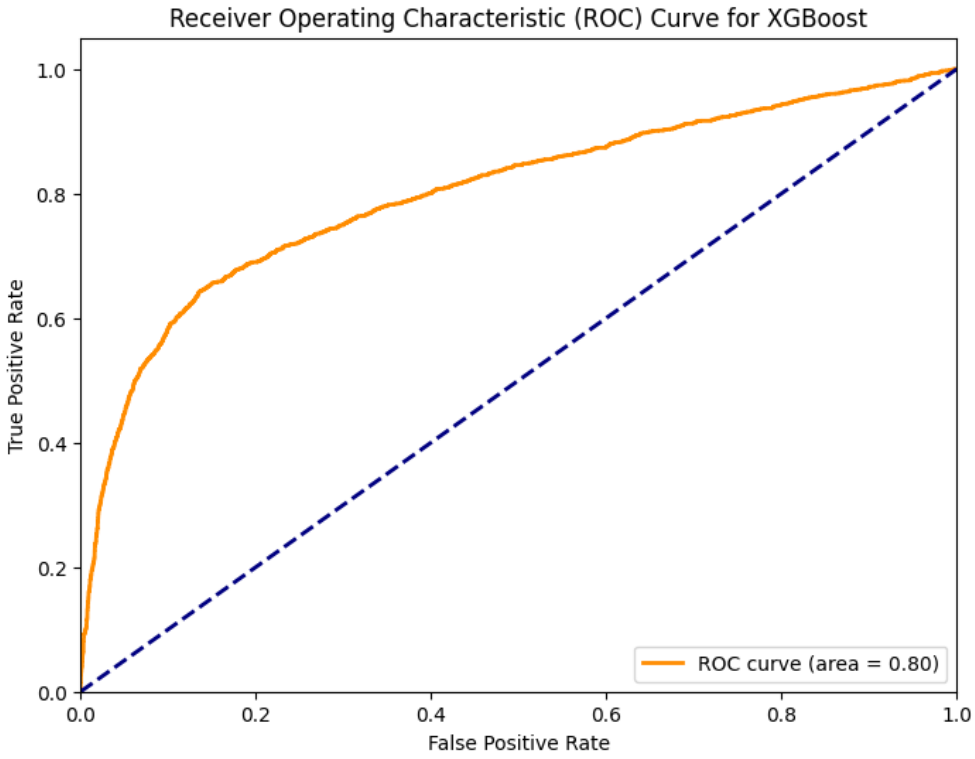
Saving the model ensures:

- Reproducibility: the same results can be reused later
- Deployment: model can be integrated into production
- Timesaving: avoid retraining every time

Results:


Confusion Matrix for Logistic Regression Model
Not Churn
Churn
True Label
7449
181
1309
422
Not Churn
Churn
Predicted Label


Confusion Matrix for Logistic Regression Model (SMOTE)
Not Churn
Churn
True Label
6041
1589
574
1157
Not Churn
Churn
Predicted Label


Receiver Operating Characteristic (ROC) Curve for XGBoost
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.80)

Task 2

Heart Disease Prediction

Objective

The goal of this task was to predict whether a patient is likely to develop heart disease using various machine learning models and compare their performance based on standard evaluation metrics.

Dataset Overview

- Loaded using: pd.read_csv()
- Rows: **303**, Columns: **14**
- Target Column: target (1 = Heart Disease, 0 = No Heart Disease)

Data Preprocessing

Missing Values:

- No missing values were found in the dataset.

Outlier Detection & Handling:

- Outliers were observed in columns like chol, trestbps, and thalach.
- Handled using:
    - **Log transformation** to reduce skewness
    - **StandardScaler** to normalize the scale of features

Train-Test Split

- Performed using train_test_split()
    - 80% Train, 20% Test

- stratify=y used to maintain class balance

Model Training

Logistic Regression

- Used as a **baseline linear model**
- Provided interpretability and clear feature coefficients

K-Nearest Neighbors (KNN)

- Distance-based model
- Accuracy improved significantly after **hyperparameter tuning (K selection)**

Random Forest Classifier

- Tree-based ensemble model
- Provided good performance and **feature importance**

Model Evaluation

Metrics Used:

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix

Cross-Validation:

- **5-Fold Cross-Validation** was applied on Logistic Regression for robust evaluation

Hyperparameter Tuning (KNN)

- Used training and testing score plots to select **optimal k**.
- Found the sweet spot where the model balanced bias and variance (typically between k=5 to k=9).
- Helped boost performance and avoid overfitting.

Feature Importance

- Extracted from **Logistic Regression coefficients**
- Top features:
    - thalach, cp, sex, oldpeak
- Visualized using a bar plot

Why Did KNN Perform Better?

Small Dataset (303 rows)

- KNN performs well on small to medium datasets because it relies on distance from nearby examples, which can be very informative in low-data settings.

Well-Scaled Features

- After applying StandardScaler, all features had similar scales.
- KNN uses Euclidean distance — so scaling had a big impact on model performance.

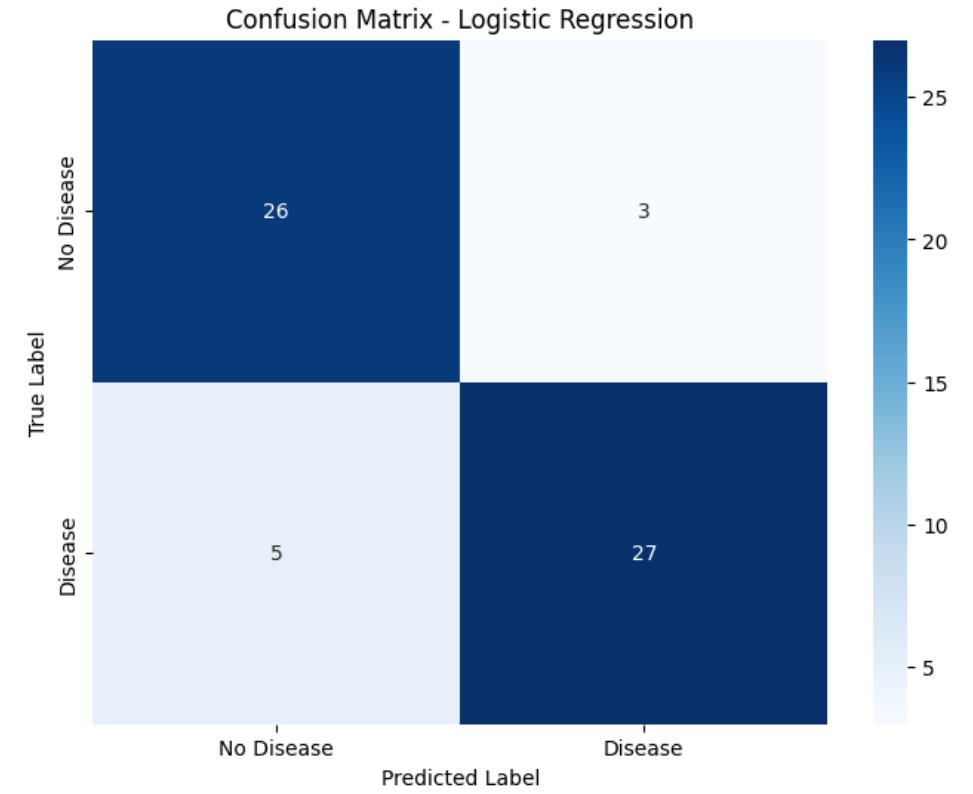
Clear Clusters in Data

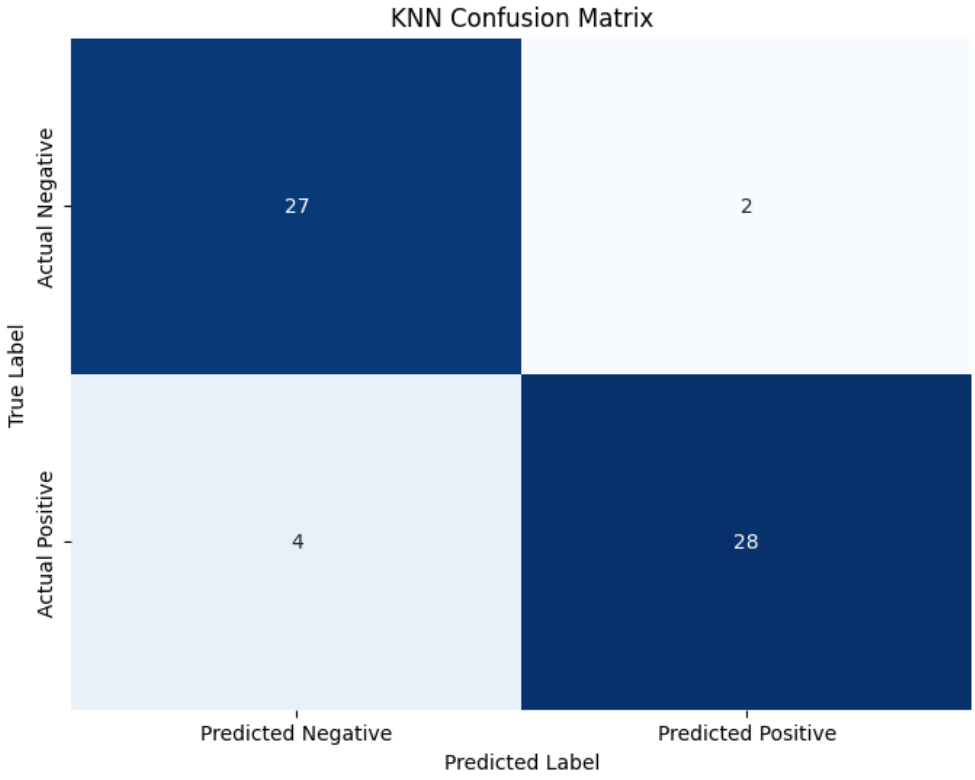
- Heart disease features like chest pain type, thalach, and oldpeak tend to naturally separate classes.
- KNN benefits when the target classes form **dense neighborhoods**.

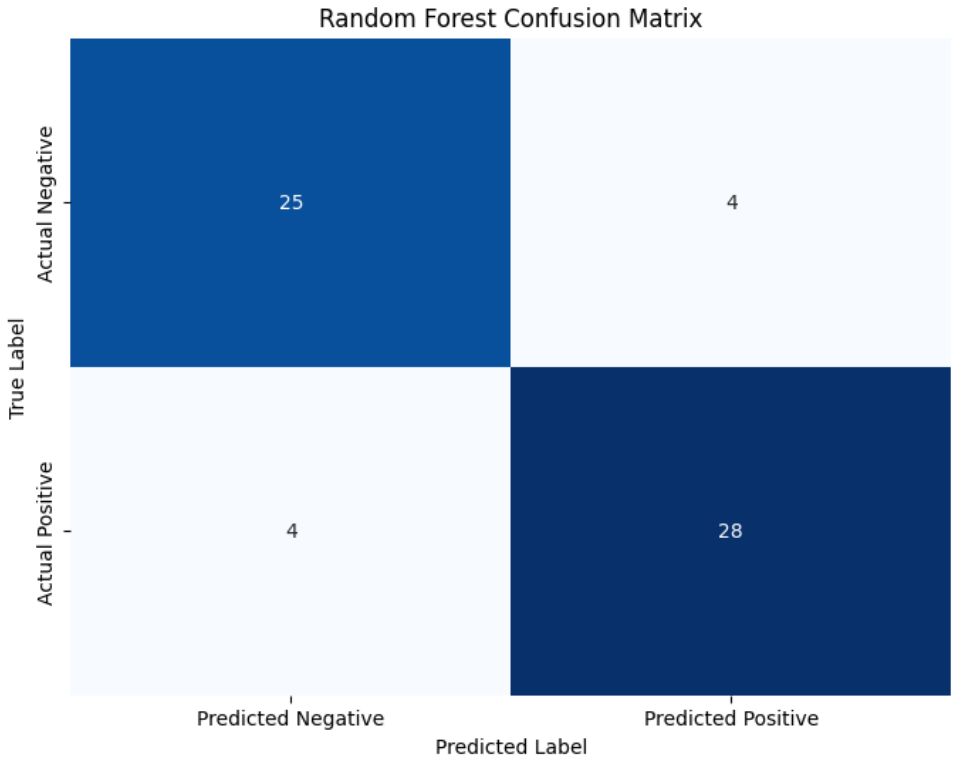
Proper k Selection

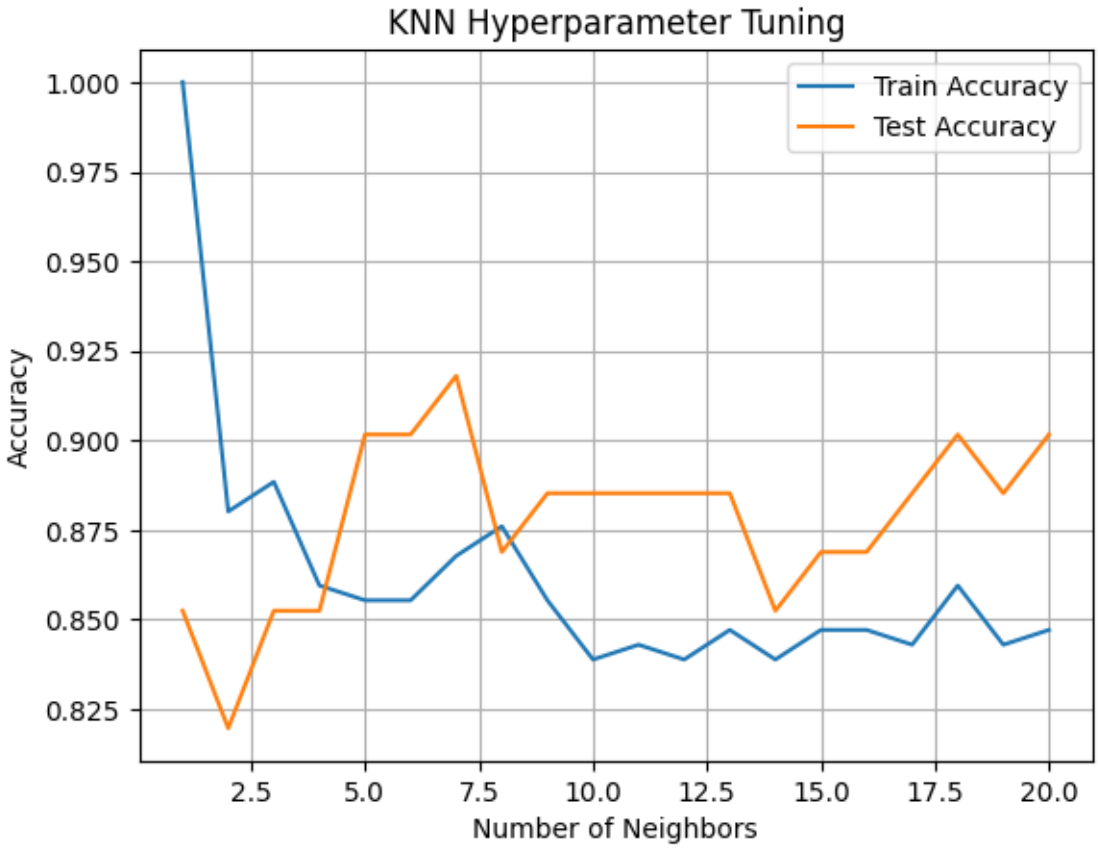
- Optimal k was chosen through hyperparameter tuning.
- This helped reduce overfitting (low k) and underfitting (high k).

Result:


Confusion Matrix - Logistic Regression
No Disease
Disease
True Label
26
3
5
27
No Disease
Disease
Predicted Label
25
20
15
10
5


KNN Confusion Matrix
Actual Negative
Actual Positive
True Label
27
2
4
28
Predicted Negative
Predicted Positive
Predicted Label


Random Forest Confusion Matrix
Actual Negative
Actual Positive
True Label
25
4
4
28
Predicted Negative
Predicted Positive
Predicted Label


KNN Hyperparameter Tuning
Train Accuracy
Test Accuracy
Accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of Neighbors