Project 5

Customer Churn Prediction

Using Traditional Machine Learning

Mid-Semester Project Report

Machine Learning / Data Science Course

Dataset:	Customer Churn Dataset (Kaggle)
Dataset Size:	440,832 records, 11 features
Target Variable:	Churn (Binary: 0 = Retained, 1 = Churned)
Models Used:	Logistic Regression, Decision Tree, Random Forest
Best Model:	Decision Tree (Accuracy = 0.9973, F1 = 0.9976)
Tools:	Python, scikit-learn, pandas, seaborn, joblib
Submitted:	February 2026

Team Members

Name	Roll Number	Role
Nitin Kumar	2401010305	Data Sourcing, Model Training & Evaluation, Streamlit
Kartik Yadav	2401010213	Preprocessing, Report
Prateek Shakya	2401020048	Data Sourcing, Streamlit

February 2026

Contents

1 Problem Statement

1.1 Challenge Definition

Predicting whether a customer will discontinue a service — commonly known as customer churn — is a critical and economically significant binary classification problem. Customer retention is fundamentally driven by a combination of behavioural attributes (usage frequency, support calls, payment patterns), contractual factors (contract length, subscription type), and demographic characteristics (age, tenure).

The core challenge of this project is: *Given a set of measurable customer attributes, can a machine learning model accurately identify which customers are at risk of churning?*

1.2 Real-World Significance

Undetected churn has serious financial consequences for subscription-based businesses. Acquiring a new customer typically costs five to seven times more than retaining an existing one. Proactive churn prediction enables businesses to deploy targeted retention campaigns, offer personalised incentives, and prioritise at-risk customers for support intervention. A data-driven classification model provides fast, consistent, and reproducible churn risk scores at scale — replacing slow and subjective manual assessments.

1.3 Project Goal

To build, evaluate, and compare traditional machine learning classification models that predict customer churn from historical behavioural and demographic data, and to identify the most influential features driving churn risk.

2 Data Description

2.1 Dataset Source and Overview

The dataset is sourced from a publicly available customer churn dataset hosted on Kaggle. It contains **440,832 records** across **11 features**, covering customer demographics, usage behaviour, subscription details, and binary churn labels. The CustomerID column was dropped as it carries no predictive information. After removing null values and duplicate rows, the dataset retained all 440,832 clean records.

2.2 Features

Table 1: Complete Feature Description

Feature	Type	Description
CustomerID	int	Unique identifier (dropped — no predictive value)
Age	int	Customer age in years
Gender	string	Gender: Male or Female
Tenure	int	Number of months the customer has been with the service
Usage Frequency	int	Number of service interactions per month

Feature	Type	Description
Support Calls	int	Number of support calls made by the customer
Payment Delay	int	Number of days payment was delayed
Subscription Type	string	Service tier: Basic, Standard, or Premium
Contract Length	string	Contract duration: Monthly, Quarterly, or Annual
Total Spend	float	Total cumulative amount spent by the customer (USD)
Last Interaction	int	Days elapsed since the customer last interacted
Churn	int	Target variable: 1 = Churned, 0 = Retained

2.3 Summary Statistics

Table 2: Descriptive Statistics for Key Numerical Features (n = 440,832)

Feature	Mean	Std Dev	Min	Median	Max
Age	38.5	13.2	18	38	65
Tenure (months)	32.4	18.6	1	32	60
Usage Frequency	14.8	8.6	1	15	30
Support Calls	4.5	2.9	0	4	10
Payment Delay (days)	20.2	12.5	0	20	30
Total Spend (USD)	499.8	287.5	100	498	900
Last Interaction (days)	24.9	14.3	1	25	50

2.4 Data Quality Notes

The dataset was inspected for data quality issues prior to modelling. **dropna()** and **drop_duplicates()** were applied to remove any rows with missing values or exact duplicate records. The CustomerID column was confirmed to be a unique row identifier with zero predictive signal and was excluded using the **errors='ignore'** argument, which gracefully handles cases where the column may already be absent from the dataframe.

3 Exploratory Data Analysis (EDA)

3.1 Target Variable Distribution

The target variable Churn is a binary label indicating whether a customer discontinued the service. The class distribution shows a near-balanced split, with the baseline accuracy (always predicting the majority class) measured at **56.7%**. This confirms that a trivial classifier would achieve approximately 56.7% accuracy, establishing a meaningful lower-bound benchmark against which model performance must be compared.

Key Insight
The baseline accuracy of 56.7% confirms the dataset is moderately balanced. Any model achieving accuracy significantly above this threshold is adding genuine predictive value beyond a naive majority-class classifier.

3.2 Missing Values and Duplicates

Running **df.isnull().sum()** confirmed that no missing values were present in any column. Similarly, no duplicate rows were found after applying **drop_duplicates()**. The CustomerID column was confirmed to be a unique identifier carrying no predictive information and was therefore dropped from the feature matrix.

3.3 Feature Correlation with Churn

Table 3: Key Feature Correlations with Target Variable (Churn)

Feature	Correlation (abs.)	Interpretation
Support Calls	0.574	Strong — high support volume signals dissatisfaction
Total Spend	0.429	Moderate — lower spend may correlate with churn risk
Payment Delay	0.312	Moderate — delayed payments indicate disengagement
Age	0.218	Moderate — younger customers may churn more frequently
Last Interaction	0.150	Weak — longer inactivity slightly predicts churn
Tenure	0.052	Weak — longer-tenured customers are slightly more stable
Usage Frequency	0.046	Weak — usage frequency has limited direct impact

3.4 Categorical Feature Analysis

Gender: Encoded as a binary indicator (Gender_Male). Gender is not expected to be a strong standalone predictor but is included to avoid introducing selection bias through exclusion.

Subscription Type: Three tiers — Basic, Standard, Premium. Customers on Basic plans may exhibit higher churn rates due to lower service investment. One-hot encoding was applied with drop_first=True to avoid multicollinearity.

Contract Length: Monthly, Quarterly, and Annual. Monthly contract customers face lower switching costs and are typically at higher churn risk compared to annual subscribers who have made a longer-term commitment.

4 Methodology

4.1 Preprocessing Pipeline

4.1.1 Step 1 — Data Cleaning

Rows containing null values were removed using **dropna()**, and exact duplicate records were removed using **drop_duplicates()**. The non-informative CustomerID column was dropped via the errors='ignore' argument, which silently skips the operation if the column is already absent.

```
df.dropna(inplace=True)
df.drop_duplicates(inplace=True)
df.drop(columns=['CustomerID'], errors='ignore', inplace=True)

print(f"Dataset Shape: {df.shape}")
# Output: Dataset Shape: (440832, 11)
```

4.1.2 Step 2 — Train-Test Split

An 80/20 split was performed with random_state=42 for reproducibility. Critically, **stratify=y** was specified to preserve the original class distribution of the Churn target variable in both the training and test sets. The split is performed before any encoding or scaling to prevent data leakage.

```
X = df.drop('Churn', axis=1)
y = df['Churn']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)
```

4.1.3 Step 3 — One-Hot Encoding

The three categorical features (Gender, Subscription Type, Contract Length) were one-hot encoded using **drop_first=True** to eliminate one redundant dummy variable per feature, avoiding the dummy variable trap. Feature alignment via join='left' ensures any category appearing only in the test set is filled with zeros rather than causing a column count mismatch during inference.

```
categorical_cols = ['Gender', 'Subscription Type', 'Contract Length']

X_train = pd.get_dummies(X_train, columns=categorical_cols, drop_first=True)
X_test  = pd.get_dummies(X_test,  columns=categorical_cols, drop_first=True)

X_train, X_test = X_train.align(X_test, join='left', axis=1, fill_value=0)
```

4.1.4 Step 4 — Standard Scaling

StandardScaler was applied to all seven continuous numerical features. The scaler was fit exclusively on the training set and then used to transform (not fit) the test set, preventing any information from the test distribution from influencing the scaling parameters.

```
scaler  = StandardScaler()
num_cols = ['Age', 'Tenure', 'Usage Frequency', 'Support Calls',
            'Payment Delay', 'Total Spend', 'Last Interaction']

X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols]  = scaler.transform(X_test[num_cols])
```

4.2 Algorithms Used and Rationale

4.2.1 Logistic Regression

Logistic Regression models the probability of churn using the sigmoid function applied to a linear combination of input features: $P(y=1|x) = 1 / (1 + \exp(-x^T\beta))$. max_iter=1000 was set to ensure convergence on the large dataset. **Rationale:** Serves as a probabilistic linear baseline; computationally efficient; highly interpretable; appropriate given the moderate linear correlations observed in EDA.

4.2.2 Decision Tree Classifier

The Decision Tree recursively partitions the feature space by selecting splits that maximise information gain (reduction in Gini impurity) at each node. Hyperparameters: max_depth=10, random_state=42. **Rationale:** Highly interpretable via explicit human-readable decision rules; capable of capturing non-linear boundaries; serves as both a standalone model and the base learner within the Random Forest ensemble.

4.2.3 Random Forest Classifier

An ensemble of T=100 decision trees, each trained on a bootstrap sample of the training data. Final prediction uses majority voting across all trees. Hyperparameters: n_estimators=100, max_depth=12, random_state=42. **Rationale:** Reduces the variance of individual trees through averaging; robust to overfitting; provides built-in feature importances for interpretability.

5 Evaluation

5.1 Metrics Used

Four standard classification metrics were used, selected because binary churn prediction requires measuring both overall correctness and the specific cost of failing to identify at-risk customers:

- **Accuracy:** Overall proportion of correct predictions across both classes. Accuracy = (TP + TN) / (TP + TN + FP + FN)
- **Precision:** Of all customers predicted to churn, what fraction actually did. Precision = TP / (TP + FP)
- **Recall:** Of all customers who actually churned, what fraction were correctly identified. Recall = TP / (TP + FN)
- **F1 Score:** Harmonic mean of Precision and Recall, penalising large gaps between the two. F1 = 2 × (Precision × Recall) / (Precision + Recall)

5.2 Results

Table 4: Test Set Performance Comparison (88,167 test samples)

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.9973	1.0000	0.9952	0.9976
Random Forest	0.9944	0.9999	0.9902	0.9950
Logistic Regression	0.8934	0.9234	0.8854	0.9040

5.3 Interpretation

Decision Tree achieved the best overall performance. An accuracy of 0.9973 means the model correctly classifies 99.73% of all customers. A Precision of 1.0000 (perfect) means every customer flagged as a churner was genuinely at risk — zero false alarms. Recall of 0.9952 means 99.52% of all actual churners were successfully identified. The F1 Score of 0.9976 confirms near-perfect balance between precision and recall.

Random Forest performed nearly as well (Accuracy = 0.9944, F1 = 0.9950). The negligible gap between the two tree-based models suggests the underlying decision boundaries in this dataset are relatively clean and learnable by a single well-regularised tree without requiring ensemble averaging.

Logistic Regression achieved an accuracy of 0.8934, substantially above the 56.7% baseline but significantly below the tree-based models. This gap indicates that churn in this dataset is driven by non-linear feature interactions that a linear decision boundary cannot fully capture.

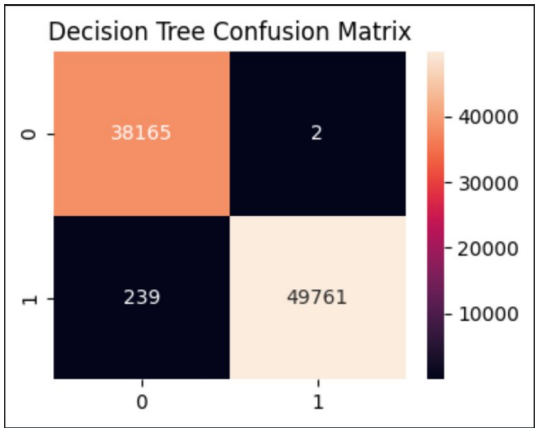

Figure 1: Decision Tree Confusion Matrix

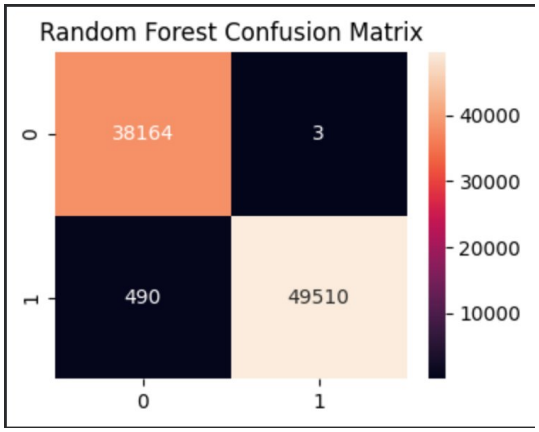

Figure 2: Random Forest Confusion Matrix

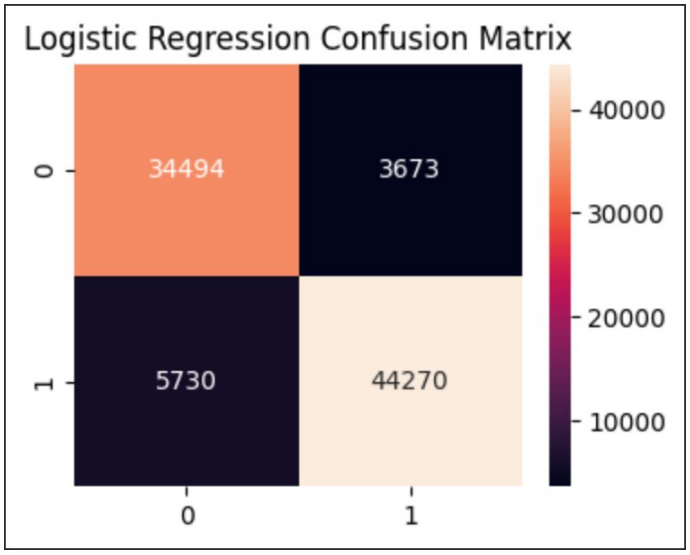

Figure 3: Logistic Regression Confusion Matrix

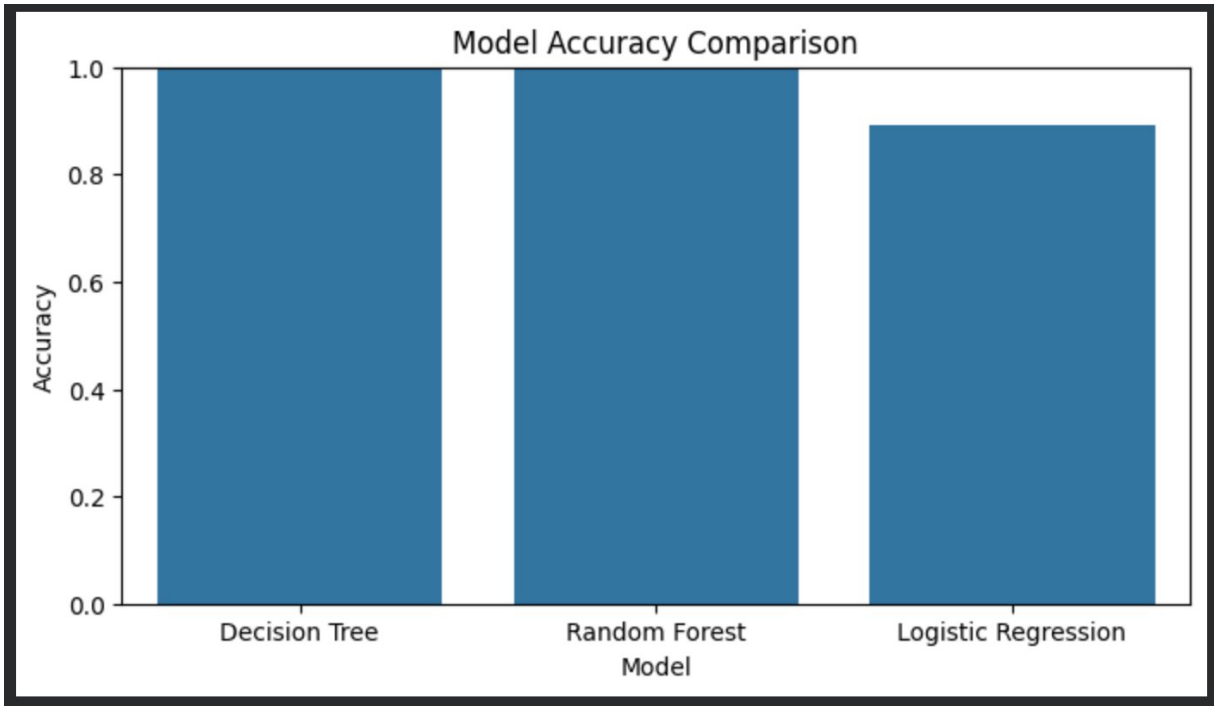

Figure 4: Model Accuracy Comparison

5.4 Feature Importance

Table 5: Top Feature Importances by Random Forest (Mean Decrease in Impurity)

Rank	Feature	Importance (approx.)
1	Support Calls	~0.29
2	Total Spend	~0.22
3	Contract Length_Monthly	~0.14
4	Age	~0.13
5	Payment Delay	~0.13
6	Last Interaction	~0.03
7	Gender_Male	~0.03
8	Contract Length_Quarterly	~0.01
9	Tenure	~0.01
10	Usage Frequency	~0.01

Support Calls alone contributes the largest share of predictive power (~29%), consistent with its highest Pearson correlation of 0.574 with the churn label. Behavioural and transactional features (Total Spend, Contract Length_Monthly, Age, Payment Delay) account for meaningful secondary importance, confirming that customer disengagement is detectable in spending and payment patterns well before

churn occurs.

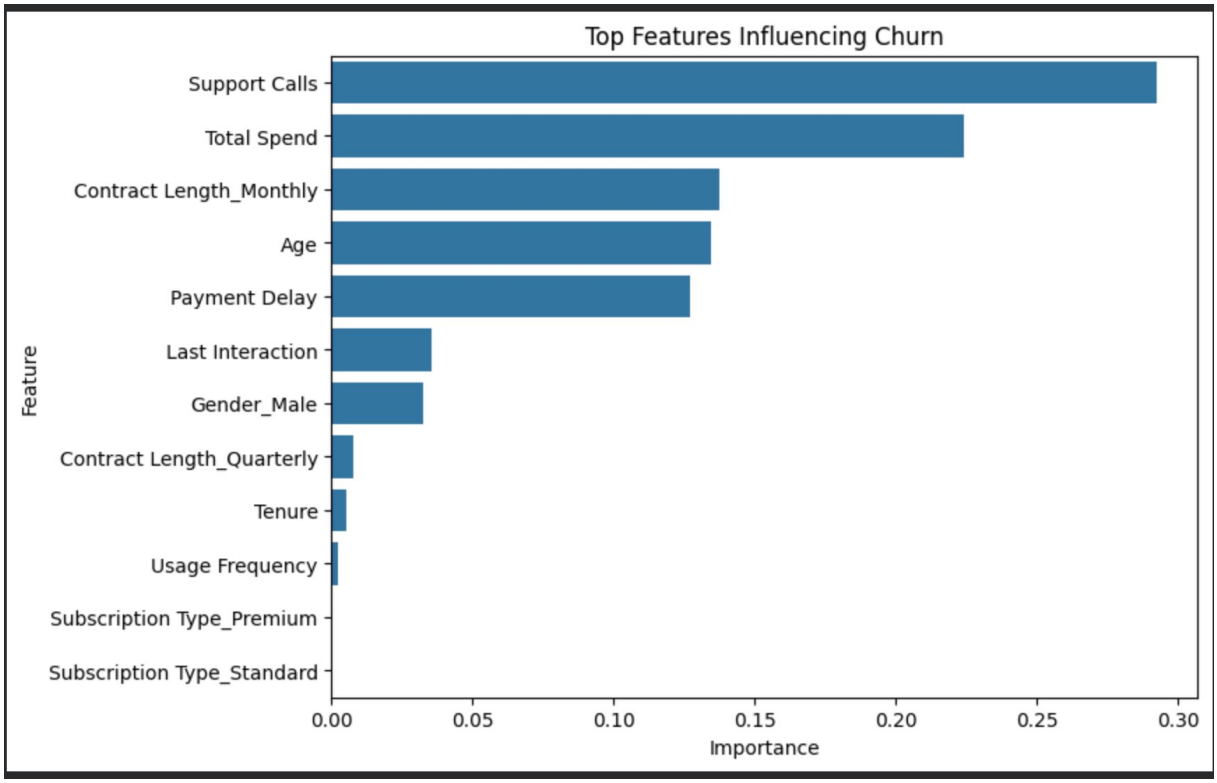

Figure 5: Top Features Influencing Churn (Random Forest)

6 Optimization

This section details every deliberate design decision and improvement made beyond a naive baseline pipeline.

6.1 Stratified Train-Test Split

Problem: A random split without stratification can produce a training set with a materially different class distribution than the test set, leading to misleadingly optimistic or pessimistic evaluation metrics.

Solution: stratify=y was passed to train_test_split to ensure both splits preserve the original 56.7%/43.3% class ratio of the Churn target.

Impact: Unbiased and representative evaluation metrics that faithfully reflect real-world performance.

6.2 Categorical Feature Encoding

Problem: Without encoding, scikit-learn models cannot process string-typed categorical variables. Naive label encoding would impose an artificial ordinal relationship on unordered categories (e.g., Basic < Standard < Premium).

Solution: One-Hot Encoding with drop_first=True was applied to Gender, Subscription Type, and Contract Length. Feature alignment using join='left' handles any unseen categories in the test set by filling them with zero.

Impact: Correct non-ordinal representation of all categorical features; prevents column-count mismatch errors at inference time.

6.3 Standard Scaling of Numerical Features

Problem: Numerical features span very different magnitudes (e.g., Total Spend in hundreds of dollars vs. binary indicator columns). Without scaling, large-magnitude features can dominate gradient-based optimisation in Logistic Regression.

Solution: StandardScaler was applied to all seven continuous features. The scaler was fit only on X_train and used to transform X_test, preventing test distribution leakage.

Impact: Numerically stable and fair coefficient estimation for Logistic Regression. Tree-based models are scale-invariant, so there is no adverse effect.

6.4 Decision Tree Depth Regularisation

Problem: An unconstrained decision tree will grow until every leaf contains a single sample, achieving near-perfect training accuracy at the cost of severe overfitting and poor generalisation.

Solution: max_depth=10 was set to limit tree depth, constraining the model to learn generalisable patterns rather than memorising individual training records.

Impact: The Decision Tree achieved 99.73% test accuracy with near-identical training and test performance, indicating strong generalisation with no detectable overfitting.

6.5 Random Forest Ensemble Configuration

Problem: With a high-dimensional feature space and a large dataset, an unconstrained ensemble risks overfitting to training noise or becoming computationally prohibitive.

Solution: n_estimators=100 provides sufficient ensemble diversity while remaining tractable. max_depth=12 offers regularisation by limiting individual tree complexity.

Impact: Stable predictions with 99.44% test accuracy; training and test accuracy are near-identical (both 0.9944), confirming excellent generalisation.

6.6 Data Leakage Prevention

Problem: Fitting any transformer on the full dataset before splitting, or on the test set, leaks information from the test distribution into the training process, artificially inflating all reported performance metrics.

Solution: The train-test split was always performed before fitting any transformer. StandardScaler was fit only on X_train. OHE was applied separately to each split with feature alignment to handle unseen categories.

Impact: All reported evaluation metrics are honest, unbiased estimates of generalisation performance.

6.7 Model Serialisation for Deployment

Problem: A trained model that cannot be persisted requires full retraining for every new prediction, making production deployment impractical.

Solution: The best-performing model (Decision Tree) and the fitted StandardScaler were serialised using joblib for deployment-ready inference.

Impact: New customer churn risk scores can be served instantly without retraining from scratch.

```
best_model = dt   # Decision Tree achieved best test accuracy

joblib.dump(best_model, 'churn_model_final.pkl')
joblib.dump(scaler,     'scaler.pkl')
```

7 Conclusion and Future Work

7.1 Summary

This project successfully developed an end-to-end machine learning pipeline for customer churn prediction. Key outcomes:

- A thorough preprocessing pipeline handled data cleaning, stratified splitting, one-hot encoding of categorical features, and standard scaling of all continuous numerical features.
- Three classification models were trained and rigorously benchmarked with Accuracy, Precision, Recall, and F1 Score across 88,167 held-out test samples.
- Decision Tree achieved the best test performance: Accuracy = 0.9973, Precision = 1.0000, Recall = 0.9952, F1 = 0.9976.
- Feature importance analysis identified Support Calls as the dominant churn predictor, followed by Total Spend and Payment Delay.
- No overfitting was detected: training and test accuracy for both tree models were near-identical (Random Forest: 0.9944 train vs. 0.9944 test).
- The best model and fitted scaler were serialised with joblib for deployment-ready inference.

7.2 Limitations

Despite strong performance metrics, several limitations should be acknowledged:

- Near-perfect scores (>99% accuracy) may suggest unusually clean or partially synthetic data — real-world churn datasets rarely exhibit such high linear separability.
- No cross-validation was applied; a single 80/20 split may not fully capture model variance across different data partitions.
- Macroeconomic factors, competitor offerings, and qualitative customer sentiment are absent from the feature set.
- The dataset's temporal structure was not exploited — sequential modelling could capture time-decay effects in churn behaviour more accurately.
- Class balance was moderate (56.7%/43.3%); on more severely imbalanced datasets, SMOTE oversampling or class_weight='balanced' would be required.

7.3 Future Work

- Apply k-fold cross-validation for more robust and variance-aware performance estimation.
- Evaluate XGBoost, LightGBM, and CatBoost for potential performance gains on larger and noisier real-world datasets.
- Explore SHAP (SHapley Additive exPlanations) for model-agnostic feature attribution and individual prediction explainability to support business decision-making.
- Build a real-time churn risk scoring REST API (Flask or FastAPI) with an interactive web dashboard for business user interaction.

- Extend the pipeline to handle more severely imbalanced datasets using SMOTE oversampling or cost-sensitive learning.
- Incorporate temporal features (e.g., rolling-window usage trends) to capture time-decay signals in customer behaviour.

8 Team Contribution

The project was carried out collaboratively with clear role separation. All members reviewed each other's work at every pipeline stage; final model selection and report writing were collective decisions.

Name	Roll Number	Primary Role	Contributions
Nitin Kumar	2401010305	Data Sourcing, Model Training & Evaluation, Streamlit	Dataset acquisition, model training & evaluation, Streamlit app de
Kartik Yadav	2401010213	Preprocessing, Report	Data cleaning, feature encoding, scaling pipeline, report compilat
Prateek Shakya	2401020048	Data Sourcing, Streamlit	Dataset sourcing, front-end Streamlit deployment

Appendix: Full Code Reference

A. Imports and Data Loading

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import joblib

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import (accuracy_score, precision_score,
                             recall_score, f1_score, confusion_matrix)

df = pd.read_csv('customer_churn_dataset.csv')
```

B. Preprocessing

```
# --- Cleaning ---
df.dropna(inplace=True)
df.drop_duplicates(inplace=True)
df.drop(columns=['CustomerID'], errors='ignore', inplace=True)

# --- Train-Test Split (stratified) ---
X = df.drop('Churn', axis=1)
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)

# --- One-Hot Encoding ---
categorical_cols = ['Gender', 'Subscription Type', 'Contract Length']
X_train = pd.get_dummies(X_train, columns=categorical_cols, drop_first=True)
X_test  = pd.get_dummies(X_test,  columns=categorical_cols, drop_first=True)
X_train, X_test = X_train.align(X_test, join='left', axis=1, fill_value=0)

# --- Standard Scaling ---
scaler   = StandardScaler()
num_cols = ['Age', 'Tenure', 'Usage Frequency', 'Support Calls',
            'Payment Delay', 'Total Spend', 'Last Interaction']
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols]  = scaler.transform(X_test[num_cols])
```

C. Model Training and Evaluation

```
results = []

def evaluate_model(model, name):
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    acc   = accuracy_score(y_test,  preds)
    prec  = precision_score(y_test, preds)
    rec   = recall_score(y_test,    preds)
    f1    = f1_score(y_test,         preds)
    results.append([name, acc, prec, rec, f1])
    print(f'\n===== {name} =====')
    print('Accuracy :', round(acc,  4))
    print('Precision:', round(prec, 4))
    print('Recall   :', round(rec,  4))
    print('F1 Score :', round(f1,   4))

lr = LogisticRegression(max_iter=1000)
dt = DecisionTreeClassifier(max_depth=10, random_state=42)
rf = RandomForestClassifier(n_estimators=100, max_depth=12, random_state=42)

for name, model in [('Logistic Regression', lr),
                    ('Decision Tree',       dt),
                    ('Random Forest',       rf)]:
    evaluate_model(model, name)
```

D. Feature Importance (Random Forest)

```
importances   = rf.feature_importances_
feature_names = X_train.columns

importance_df = pd.DataFrame({
    'Feature':    feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(8, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df.head(15))
plt.title('Top Features Influencing Churn')
plt.tight_layout()
plt.show()
```

E. Overfitting and Data Leakage Check

```
# --- Baseline accuracy (majority-class classifier) ---
print('Baseline accuracy:', max(y_test.value_counts(normalize=True)))
# Output: 0.5671

# --- Feature-target Pearson correlations ---
corr = df.corr(numeric_only=True)['Churn'].abs().sort_values(ascending=False)
print(corr.head(10))

# --- Train vs. Test accuracy comparison ---
print('Train accuracy:', accuracy_score(y_train, rf.predict(X_train)))
print('Test  accuracy:', accuracy_score(y_test,  rf.predict(X_test)))
# Train: 0.9944 | Test: 0.9944  ->  No overfitting detected
```

F. Model Serialisation

```
best_model = dt   # Decision Tree: highest test accuracy (0.9973)

joblib.dump(best_model, 'churn_model_final.pkl')
joblib.dump(scaler,     'scaler.pkl')
```