Predicting & Reducing Customer Churn Using Data Analytics

Name: A. W. D. H. Chamod Lakshitha

Module: BCS Level 4 - Data Analyst (DMP)

Project type: Workplace project

Date submitted:

Word count:

Table of Contents

1. Executive Summary

Customer churn is a major challenge for subscription businesses. When customers leave, companies lose revenue and also spend more money to attract new customers. Many organizations only respond after the customer has already cancelled which will be too late. The aim of this project is to predict churn earlier using data analysis. This allows businesses to act before the customer leaves.

The project has three main goals. The first is to build a model that predicts which customers are at risk of leaving. The second is to find the key factors that explain why churn happens. Thirdly is to make those recommendations actionable for both managers and service, marketing teams.

To achieve this, project will use the Telco Customer Churn dataset from Kaggle. The dataset contains customer details such as age, gender, contract type, payment method and service use. The data will be cleaned and prepared before modelling. The primary method used will be Logistic regression. A decision tree model may also be tested. The models will be checked using precision, recall and F1 score.

The outputs will include a prediction model, dashboards, and a final report. These results will give the business practical insights into churn and advice on how to reduce it. Only anonymized open data will be used, which ensures compliance with data protection rules.

2. Introduction & Context

Customer churn is one of the biggest problem for subscription-based businesses like telecoms, banks and online services. Churn happens when customers stop using a service. It directly reduces revenue and slow down growth. Research suggests that keeping customers is often less costly than acquiring new ones.[1] This makes reducing churn important for long-term success.

Many companies still deal with churn in a reactive way. A common example is offering discounts after the customer has already decided to cancel. This may help in the short term but it does not address the real reasons why customers leave. A better approach is to predict churn in advance. Using data analytics, businesses can find at risk customers early and act with targeted offers and providing better services.

This project will use customer data such as demographics, contract type, payment method and usage history to build a churn prediction model. The analysis will highlight the main factors linked to churn and provide practical recommendations for improving retention.

The main audience for this project will be managers, marketing teams and customer service staff. Managers want to see the financial impact. Marketing will focus on customer groups. Service teams will use the insights to improve support.

[1] Bhide, A. and Stevenson, H.H. (1990). Why Be Honest If Honesty Doesn’t Pay. [online] Harvard Business Review. Available at: https://hbr.org/1990/09/why-be-honest-if-honesty-doesnt-pay.

3. Requirements Statement & Project Plan

3.1 Requirements Statement

The main requirement of this project is to develop a predictive model that can identify customers at risk of churn. Stakeholders want a solution that provides early warnings so that the business can take proactive steps to improve retention.

The deliverables will include:

- A cleaned and prepared dataset with key customer details.
- A predictive model that produces churn risk scores.
- Dashboards and visual reports showing churn rates, risk segments, and important drivers.
- A final summary report with recommendations for action.

The project will focus on using demographic, account, and service usage data. It will not include competitor analysis or detailed financial forecasting, as these are beyond the scope.

3.2 Project Plan

The project will follow the data analysis lifecycle:

1. Problem Definition – define KPIs such as churn rate, prediction accuracy, and retention rate.

2. Data Collection – gather open-source datasets (Telco Customer Churn from Kaggle).

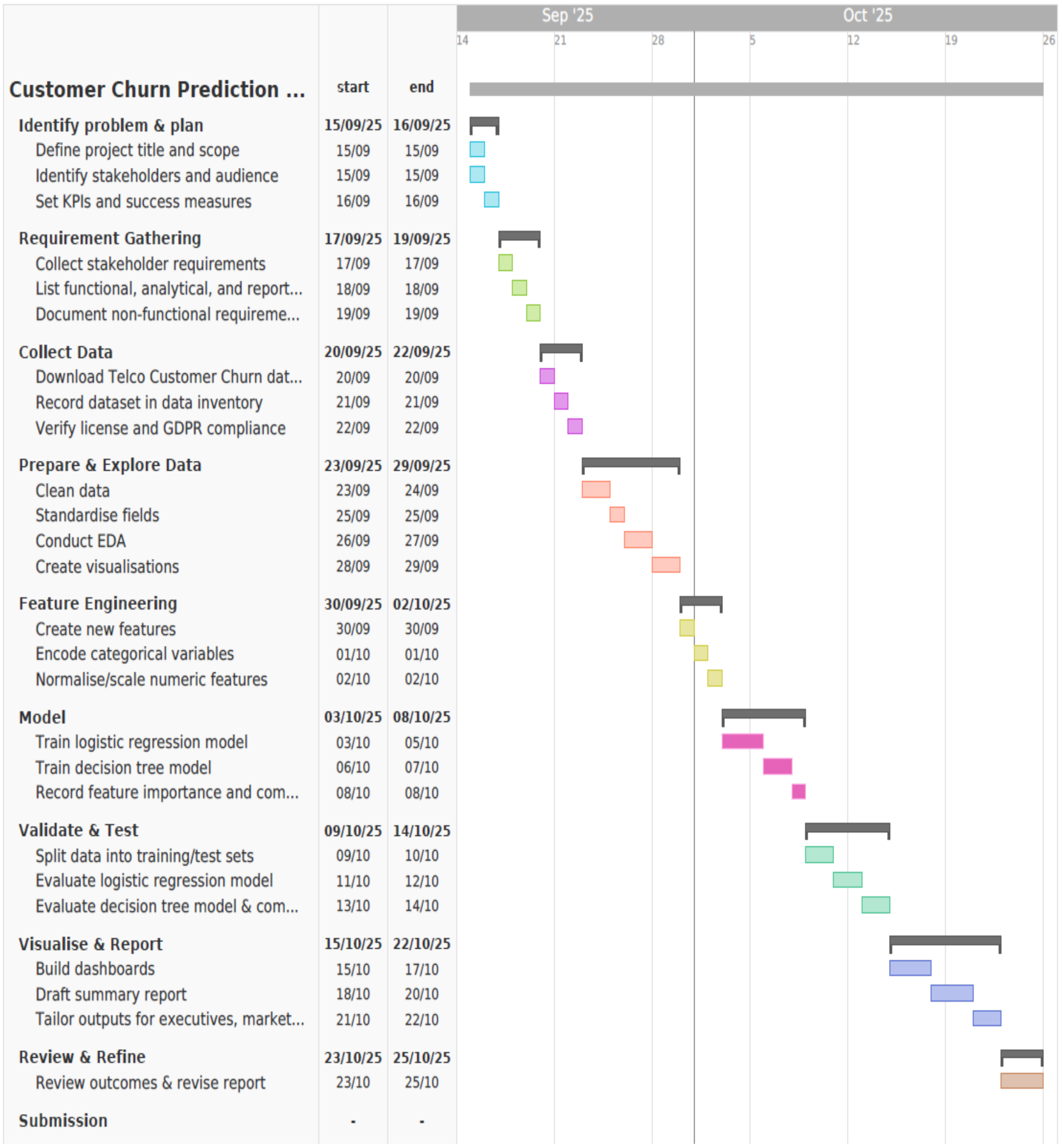
3. Data Preparation – Prepare the data by fixing errors, filling in missing information, and adding useful details like how long a customer has stayed.

4. Analysis & Modelling – apply logistic regression and compare with decision trees for prediction.

5. Validation & Testing – split the data into training and test sets, and evaluate results using accuracy, precision, recall, and F1 score.

6. Communication – produce dashboards and reports tailored to executives, marketing, and customer service.

3.3 Project Timeline



Customer Churn Prediction ...	start	end
Identify problem & plan	**15/09/25**	**16/09/25**
Define project title and scope	15/09	15/09
Identify stakeholders and audience	15/09	15/09
Set KPIs and success measures	16/09	16/09
Requirement Gathering	**17/09/25**	**19/09/25**
Collect stakeholder requirements	17/09	17/09
List functional, analytical, and report...	18/09	18/09
Document non-functional requireme...	19/09	19/09
Collect Data	**20/09/25**	**22/09/25**
Download Telco Customer Churn dat...	20/09	20/09
Record dataset in data inventory	21/09	21/09
Verify license and GDPR compliance	22/09	22/09
Prepare & Explore Data	**23/09/25**	**29/09/25**
Clean data	23/09	24/09
Standardise fields	25/09	25/09
Conduct EDA	26/09	27/09
Create visualisations	28/09	29/09
Feature Engineering	**30/09/25**	**02/10/25**
Create new features	30/09	30/09
Encode categorical variables	01/10	01/10
Normalise/scale numeric features	02/10	02/10
Model	**03/10/25**	**08/10/25**
Train logistic regression model	03/10	05/10
Train decision tree model	06/10	07/10
Record feature importance and com...	08/10	08/10
Validate & Test	**09/10/25**	**14/10/25**
Split data into training/test sets	09/10	10/10
Evaluate logistic regression model	11/10	12/10
Evaluate decision tree model & com...	13/10	14/10
Visualise & Report	**15/10/25**	**22/10/25**
Build dashboards	15/10	17/10
Draft summary report	18/10	20/10
Tailor outputs for executives, market...	21/10	22/10
Review & Refine	**23/10/25**	**25/10/25**
Review outcomes & revise report	23/10	25/10
Submission	-	-

Figure 1: Project timeline represented as a Gantt chart

3.4 Project Milestones

Milestone	Tasks	Estimated hours	Due date
Identify problem & plan	Define KPIs, confirm requirements, scope	4-6	2025-09-15
Collect data	Source dataset, prepare inventory	4-6	2025-09-22
Prepare & explore	Clean data, handle missing values, exploratory data analysis (EDA)	10-12	2025-10-02
Model	Build logistic regression & decision tree	12-14	2025-10-14
Validate & test	Evaluate accuracy, precision, recall, F1	5-7	2025-10-20
Visualize & present	Create dashboards, write final report	8-10	2025-10-31

4. Collect the Data

The success of this project depends on the availability and quality of data. An open dataset commonly used in churn prediction studies will be used. The selected dataset is the Telco Customer Churn dataset available on Kaggle. This dataset contains information about a telecommunications company's customers. It includes a column showing whether each customer has churned, which makes it suitable for supervised modelling.

The dataset has more than 7,000 records and over 20 variables. These include age, gender, type of contract, payment method, monthly charges, tenure, internet services, and customer support features. It also provides information about customer support interactions. These fields are important because they showcase common factors that influence churn.

The dataset will be downloaded in CSV format directly from Kaggle. The dataset is anonymized and public available so there should not be any issues with regard to legal or ethical responsibilities with personal data. This also ensures compliance with GDPR.

Once collected, the data will go through a preparation stage. This includes handling missing values, removing duplicates, and standardizing categories such as payment methods. Exploratory Data Analysis (EDA) will then be carried out to understand patterns and distributions. For example, churn rates will be compared by contract type, gender, or payment method. These insights will guide the selection of features for predictive modelling.

An inventory of the data sources is provided below:

#	Dataset name	Source	License / terms	Date accessed	Access method	Key fields used
1	Telco Customer Churn	https://www.kaggle.com/datasets/blastchar/telco-customer-churn	Open dataset, anonymized, free	2025-09-15	Download CSV	CustomerID, Gender, Age, Tenure, Contract, PaymentMethod, MonthlyCharges, TotalCharges, Churn

5. Prepare & explore the data

5.1 Data profiling / EDA summary

5.1.1 Dataset size and basic profile

- Rows: 7,043 customers.
- Columns: about 40 after encoding and feature engineering.
- Main target: Churn (Yes / No).

5.1.2 Missing values and duplicates

- TotalCharges had some non-numeric values and a few missing entries. These were converted to numeric and filled with the median.
- No duplicate customerID rows were found after initial checks.

5.1.3 Key variable distributions

- Churn distribution: ~25–30% churners
- Tenure: Most customers have been with the company for a short time, and they are more likely to leave.
- MonthlyCharges: Charges vary widely, and customers with higher monthly charges are more likely to churn.
- Contract: Customers with month to month contracts leave more often than those with 1 or 2 year contracts.
- Payment Method: Customers using electronic checks tend to leave more often.

5.1.4 Visualizations

5.1.4.1 Bar chart for Churn distribution

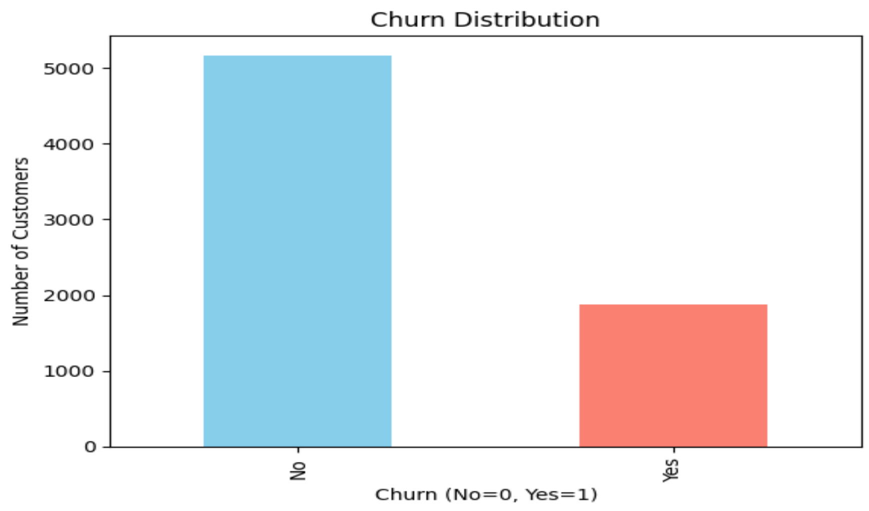


Figure 2: Bar chart showing churn distribution among customers

5.1.4.2 Histogram of tenure split by churn

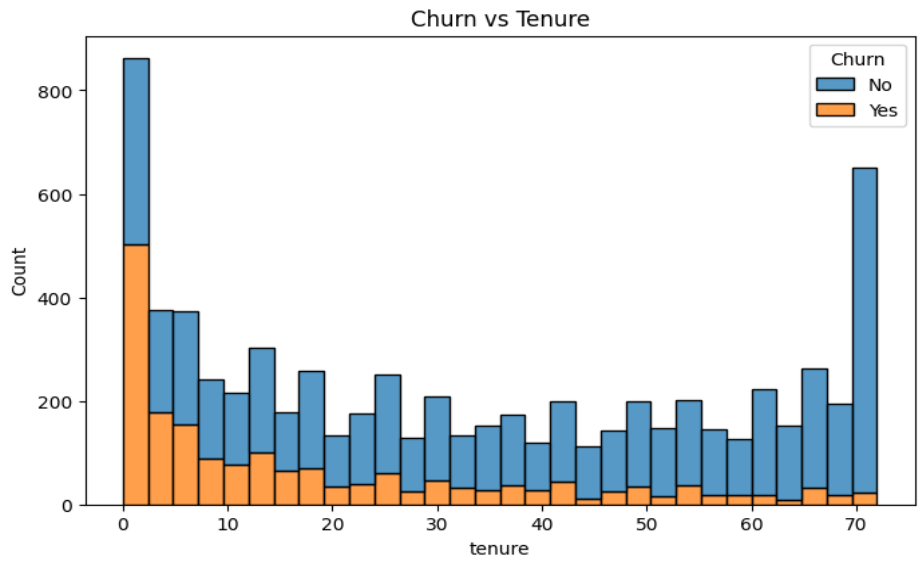


Figure 3: Histogram showing tenure distribution split by churn status

5.1.4.3 Count plot of Contract vs Churn

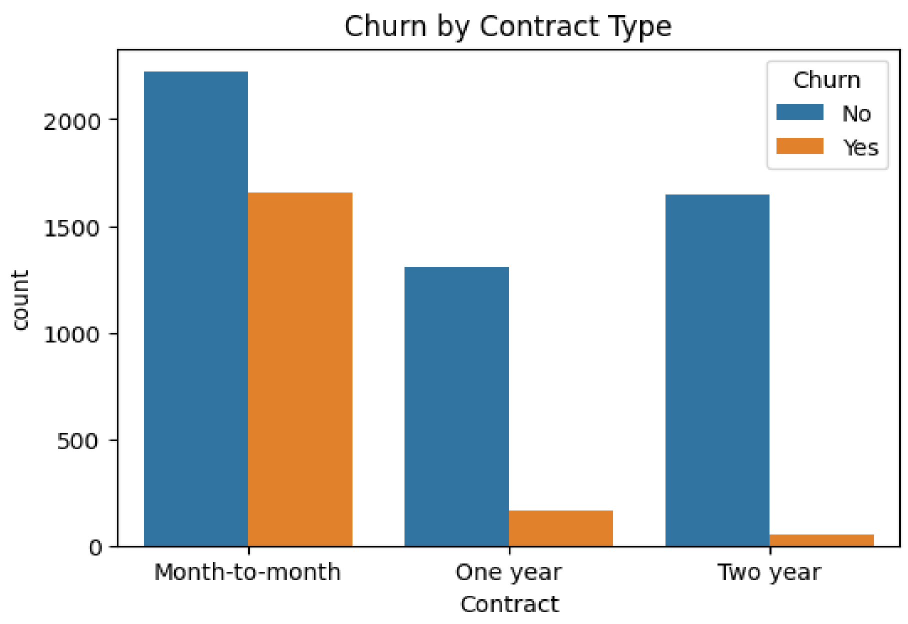


Figure 4: Count plot comparing contract type with churn rate

5.2 Data model / structure

5.2.1 Logical Entity Relationship Diagram (ERD)

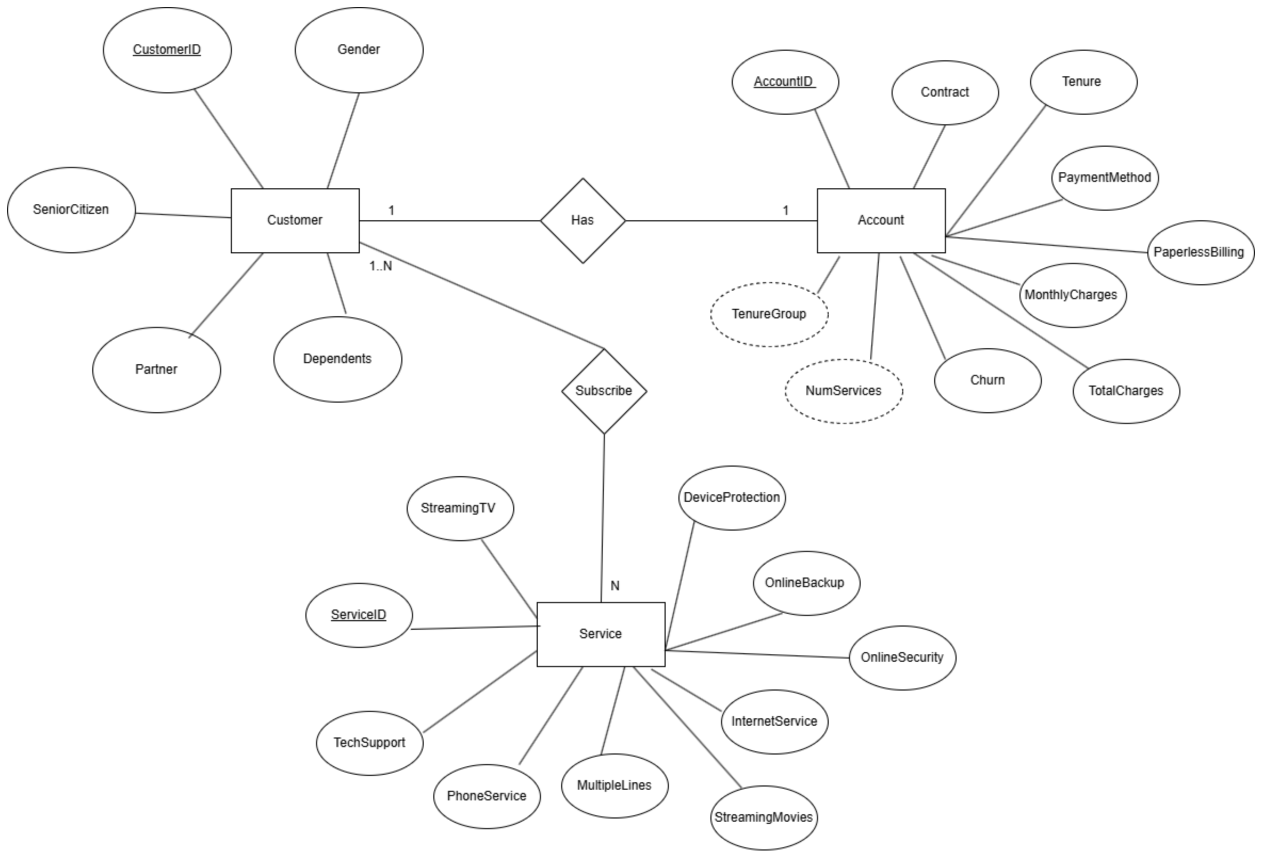


Figure 5: Logical Entity Relationship Diagram (ERD) of the dataset

The above ER diagram shows the logical structure of the data used for the customer churn prediction.

The Customer entity represents individual customers and stores their demographic information such as gender, senior citizen status, and tenure. Each customer is uniquely identified by the CustomerID, which acts as the primary key.

The Account entity contains details related to the customer's contract, billing, and payment information. The CustomerID in this table functions as a foreign key, linking each account record to its corresponding customer.

The Service entity records information about the services each customer subscribes to, such as phone, internet, and streaming services. It also uses CustomerID as a foreign key to maintain a relationship with the Customer entity.

These relationships together maintain referential integrity across entities, allowing all relevant details about a customer, their account, and the services they use to be connected logically. This combine view supports accurate analysis of churn behaviour and helps in identifying the key factors influencing customer retention.

5.3 Cleaning Log

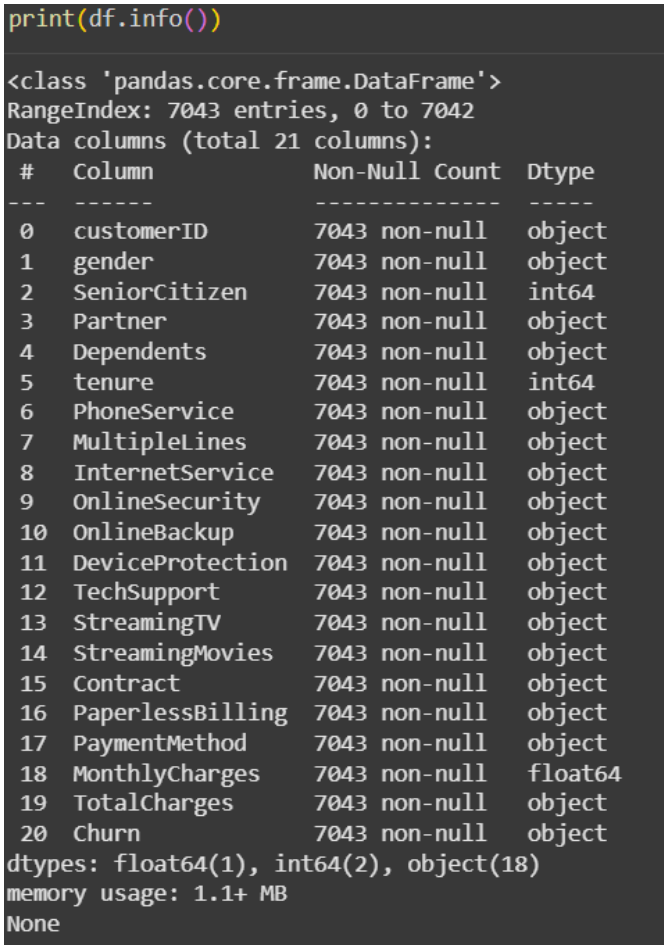
The raw dataset was first explored to identify data quality issues such as incorrect data types, duplicate records, and inconsistent values.

Each issue was identified, corrected, and verified using Python in Google Colab.

5.3.1 Data Inspection

The structure of the dataset was checked using the info() and duplicated() functions to understand its data types and potential issues.

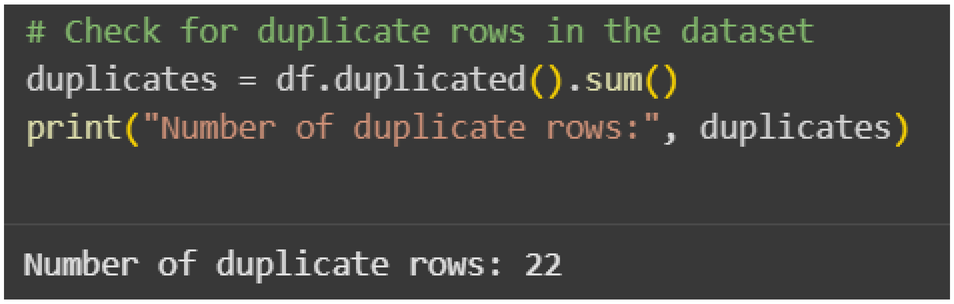
Dataset structure before cleaning:



```
print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
None
```

Figure 6: Dataset structure before cleaning and preprocessing

Duplicate count before cleaning:



```
# Check for duplicate rows in the dataset
duplicates = df.duplicated().sum()
print("Number of duplicate rows:", duplicates)

Number of duplicate rows: 22
```

Figure 7: Duplicate record count before data cleaning

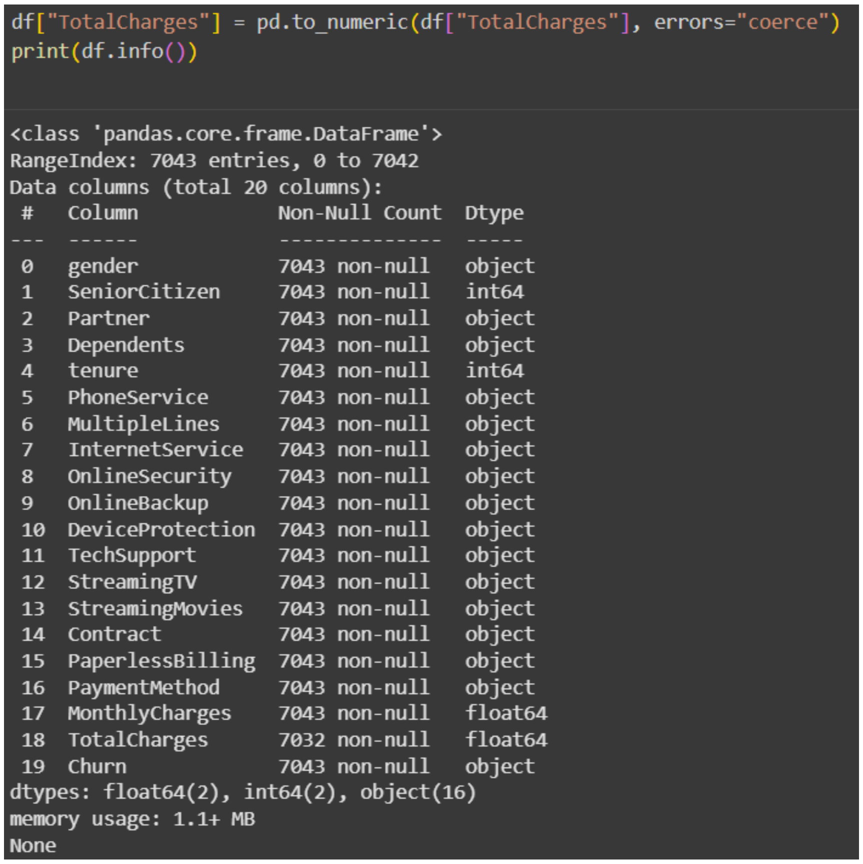
5.3.2 Issues Identified

Issue 1: "TotalCharges" column stored as text

The TotalCharges column was found to be stored as text, which prevents numeric calculations.

It was converted into numeric format.

Data type of TotalCharges changed from object to float.



```
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7043 non-null   object
 1   SeniorCitizen     7043 non-null   int64
 2   Partner           7043 non-null   object
 3   Dependents        7043 non-null   object
 4   tenure            7043 non-null   int64
 5   PhoneService      7043 non-null   object
 6   MultipleLines     7043 non-null   object
 7   InternetService   7043 non-null   object
 8   OnlineSecurity    7043 non-null   object
 9   OnlineBackup      7043 non-null   object
 10  DeviceProtection  7043 non-null   object
 11  TechSupport       7043 non-null   object
 12  StreamingTV       7043 non-null   object
 13  StreamingMovies   7043 non-null   object
 14  Contract          7043 non-null   object
 15  PaperlessBilling  7043 non-null   object
 16  PaymentMethod     7043 non-null   object
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7032 non-null   float64
 19  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
None
```

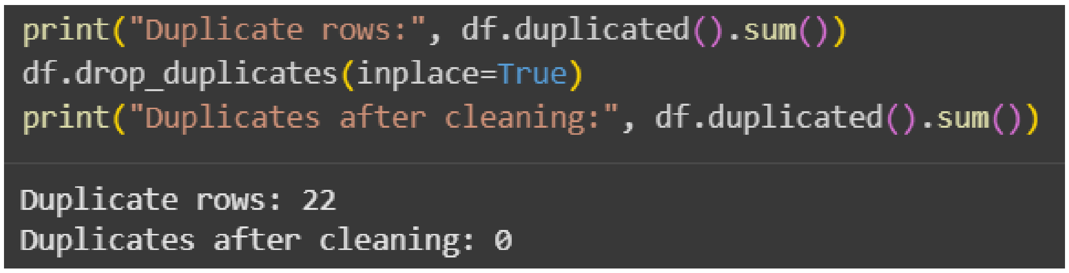
Figure 8: Data structure after converting TotalCharges from object to float

Issue 2: Duplicate records found

The duplicate check was performed after removing the customerID column, since each ID was unique and would prevent detection of true duplicates.

A total of 22 duplicate rows were identified, meaning multiple customers shared identical attribute values.

These duplicates were removed to ensure that each record represented a unique customer profile and to avoid bias in the predictive model.



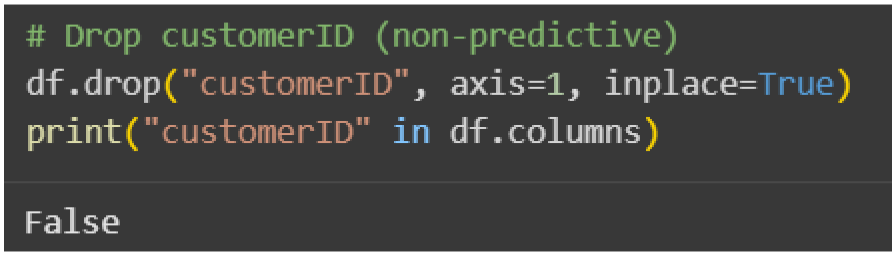
```
print("Duplicate rows:", df.duplicated().sum())
df.drop_duplicates(inplace=True)
print("Duplicates after cleaning:", df.duplicated().sum())
```

```
Duplicate rows: 22
Duplicates after cleaning: 0
```

Figure 9: Output showing the number of duplicate records found & removed

Issue 3: Removing Unnecessary Column

The customerID column was removed as it does not provide any useful information for prediction.



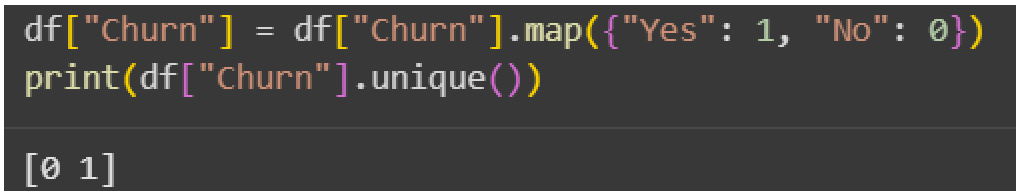
```
# Drop customerID (non-predictive)
df.drop("customerID", axis=1, inplace=True)
print("customerID" in df.columns)
```

```
False
```

Figure 10: Confirmation of removal of the customerID column

Issue 4: Encoding the Target Variable

The Churn column was encoded from text values (“Yes”, “No”) into binary (1, 0) to prepare it for model training.



```
df["Churn"] = df["Churn"].map({"Yes": 1, "No": 0})
print(df["Churn"].unique())
```

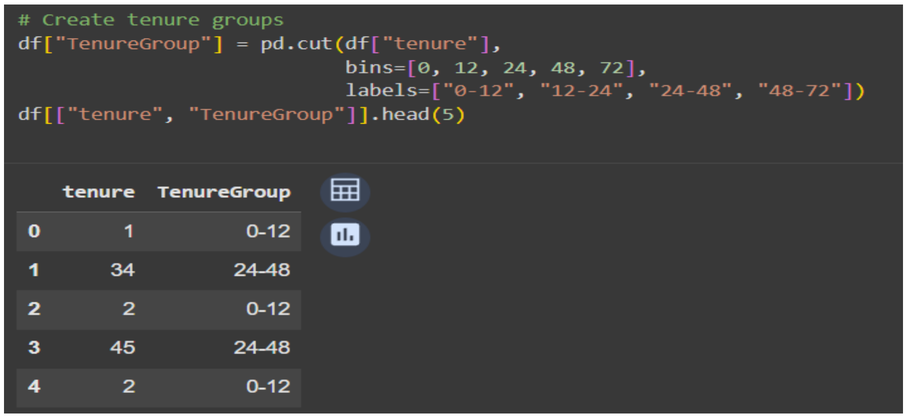
```
[0 1]
```

Figure 11: Output showing that the Churn column was encoded from text (“Yes”, “No”) to binary (1, 0)

5.4 Feature Engineering

5.4.1 Tenure Group

A new feature, TenureGroup, was created by categorizing customer tenure into four groups. This transformation helps identify whether churn is higher among new or long-term customers.



```
# Create tenure groups
df["TenureGroup"] = pd.cut(df["tenure"],
                           bins=[0, 12, 24, 48, 72],
                           labels=["0-12", "12-24", "24-48", "48-72"])
df[["tenure", "TenureGroup"]].head(5)
```

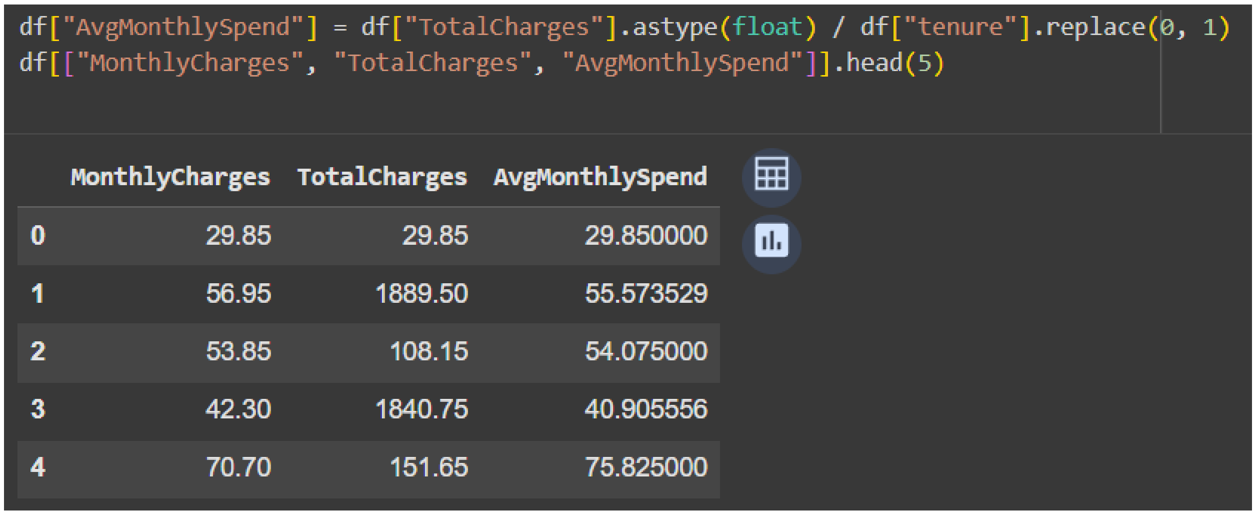
	tenure	TenureGroup
0	1	0-12
1	34	24-48
2	2	0-12
3	45	24-48
4	2	0-12

Figure 12: Output showing creation of the TenureGroup column by categorising customer tenure into four ranges

5.4.2 Average Monthly Spend

A new feature AvgMonthlySpend was derived by dividing total charges by tenure.

This shows each customer’s average monthly spend and helps identify whether spending behaviour relates to churn.



```python
df["AvgMonthlySpend"] = df["TotalCharges"].astype(float) / df["tenure"].replace(0, 1)
df[["MonthlyCharges", "TotalCharges", "AvgMonthlySpend"]].head(5)
```

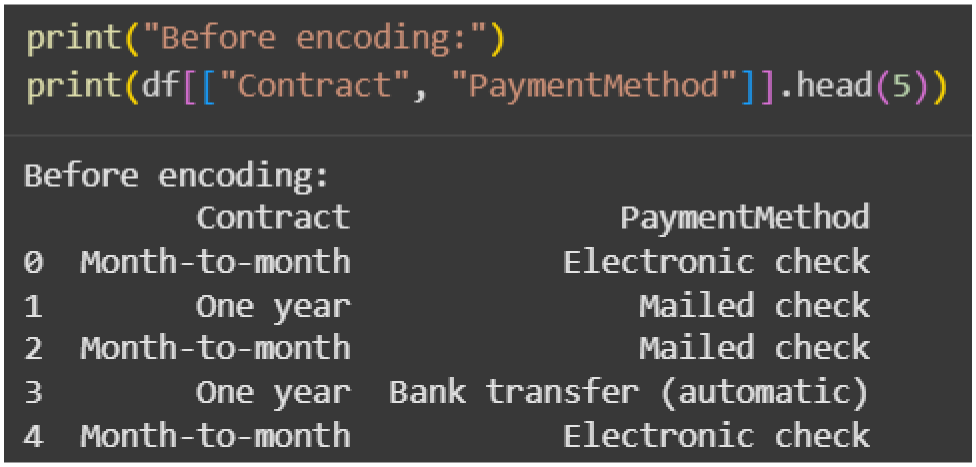
	MonthlyCharges	TotalCharges	AvgMonthlySpend
0	29.85	29.85	29.850000
1	56.95	1889.50	55.573529
2	53.85	108.15	54.075000
3	42.30	1840.75	40.905556
4	70.70	151.65	75.825000

Figure 13: Output showing calculation of the AvgMonthlySpend column derived from TotalCharges divided by tenure.

5.4.3 One-Hot Encode Categorical Variables

All categorical columns were transformed into numerical form using one-hot encoding.

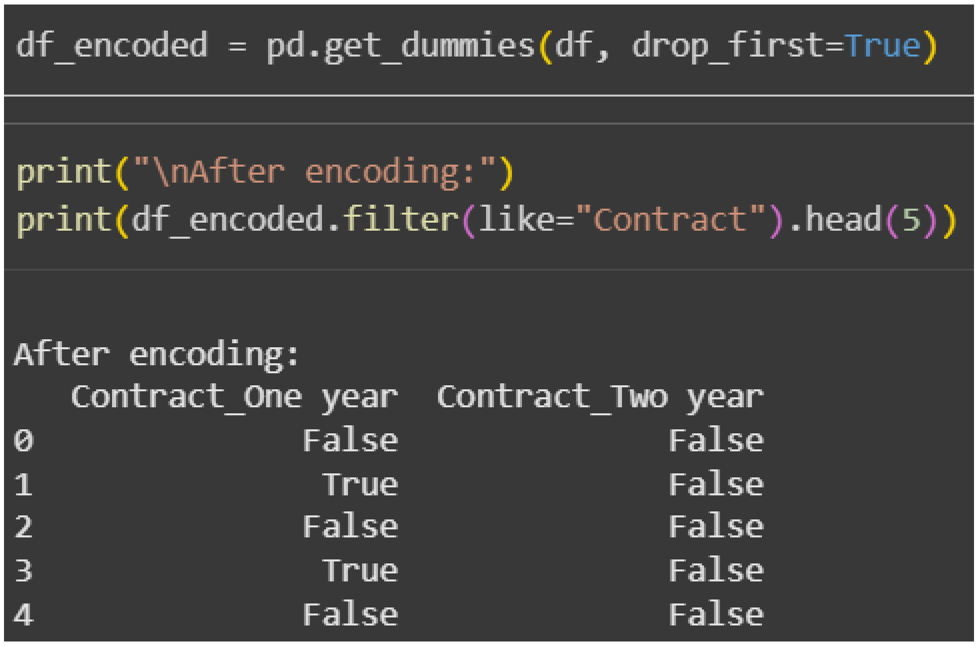
This prepares the dataset for machine learning algorithms such as logistic regression and decision trees.



```
print("Before encoding:")
print(df[["Contract", "PaymentMethod"]].head(5))
```

```
Before encoding:
         Contract              PaymentMethod
0  Month-to-month           Electronic check
1        One year               Mailed check
2  Month-to-month               Mailed check
3        One year  Bank transfer (automatic)
4  Month-to-month           Electronic check
```

Figure 14: Before encoding



```
df_encoded = pd.get_dummies(df, drop_first=True)
```

```
print("\nAfter encoding:")
print(df_encoded.filter(like="Contract").head(5))
```

```
After encoding:
   Contract_One year  Contract_Two year
0              False              False
1               True              False
2              False              False
3               True              False
4              False              False
```

Figure 15: One-hot encoding applied, converting contract type values from text into boolean numeric form (True/False)

6. Model the Data

6.1 Problem Framing

The primary purpose of this project is to predict churn (customers who are going to leave the network).

This is the case of a binary classification problem, with a target variable Churn (1 = churned, 0 = retained).

The idea behind modelling is to learn about any type of patterns in our customers activities that can be a pre-symptom for churn.

By predicting which customers are most likely to leave, the business can take preventive steps such as offering retention discounts or improving customer service.

The features used for modelling include both numerical and categorical attributes, such as:

- Tenure
- Monthly and total charges
- Contract type
- Internet and phone services used
- Payment method
- Demographic information

These features represent customer behavior, service engagement, and payment preferences of which can influence churn.

6.2 Chosen Techniques

Two machine learning techniques were used to build and compare predictive models:

1. Logistic Regression

Logistic Regression is good for binary problems like churn.

It is simple and gives probabilities that show the risk of churn.

It also makes it easier to explain which features increase or decrease churn, which is important for business users.

2. Decision Tree

The Decision Tree was used as the second model because it can handle more complex patterns.

It makes rules such as “if tenure is low and contract type is month to month, then churn is high.”

Such rules are simple for managers to understand and can be helpful when presenting results to non-technical people.

Also, the usage of both models enable to compare a statistical model and rule based model to see which has better performance in overall for predicting customer churn.

6.3 Model Specification

The features used for modelling included all key customer details that were likely to influence churn. These were a mix of numerical and categorical variables, such as:

- Numerical: tenure, MonthlyCharges, TotalCharges, AvgMonthlySpend
- Categorical(encoded): Contract, PaymentMethod, InternetService, OnlineSecurity, TechSupport

The target variable was Churn, encoded as 1 = churned and 0 = not churned.

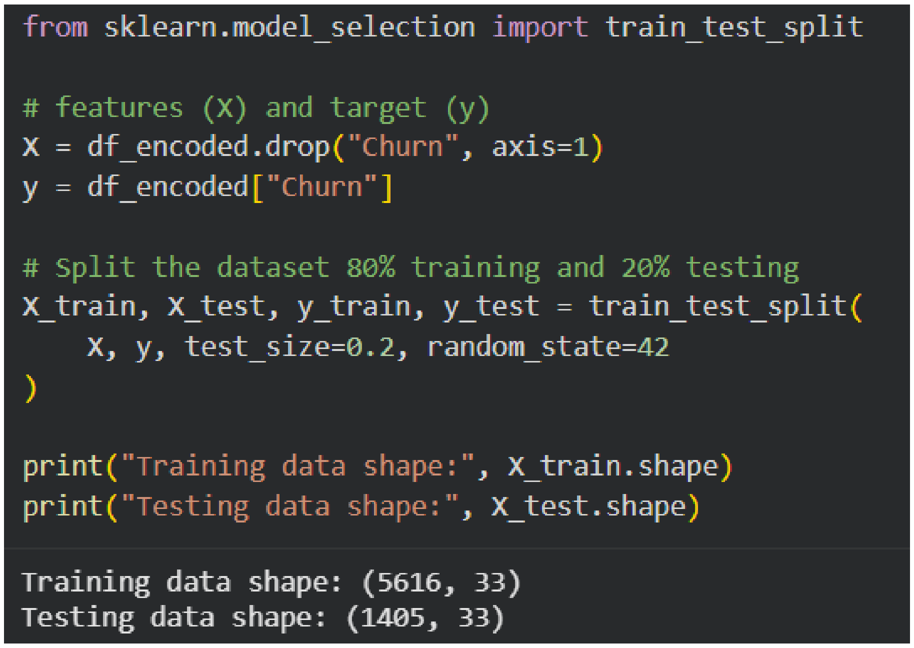
All categorical columns were one-hot encoded before training to make them suitable for machine learning models.

6.3.1 Training and Test Split

To train and evaluate the model fairly, the dataset was divided into two parts:

- 80% for training
- 20% for testing

The training data was used to teach the model the relationship between features and churn, while the test data was used to check how well the model performs on unseen data.



```python
from sklearn.model_selection import train_test_split

# features (X) and target (y)
X = df_encoded.drop("Churn", axis=1)
y = df_encoded["Churn"]

# Split the dataset 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)
```

```
Training data shape: (5616, 33)
Testing data shape: (1405, 33)
```

Figure 16: Output showing 80/20 split of the dataset into training and testing sets.

6.3.2 Import Models and Train Them

Two models were selected:

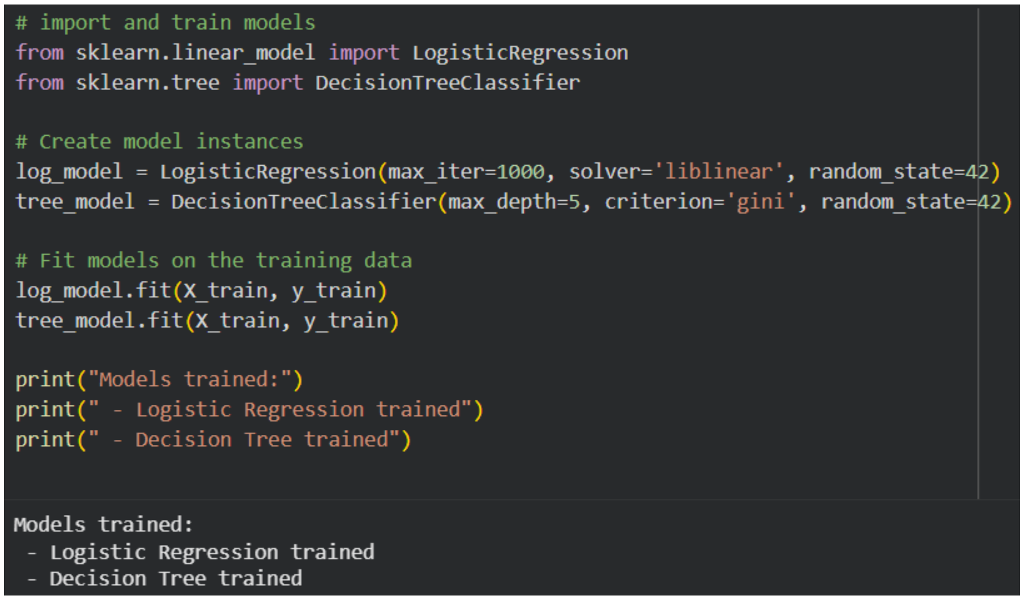
1. Logistic Regression

- Works well for binary outcomes (such as churn = yes/no)
- Produces probabilities, which help in ranking customers by risk

2. Decision Tree Classifier

- Can model non-linear relationships
- Produces simple if-then rules that are easy for managers to understand

Both models were trained using the training dataset (X_train, y_train).



```
# import and train models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

# Create model instances
log_model = LogisticRegression(max_iter=1000, solver='liblinear', random_state=42)
tree_model = DecisionTreeClassifier(max_depth=5, criterion='gini', random_state=42)

# Fit models on the training data
log_model.fit(X_train, y_train)
tree_model.fit(X_train, y_train)

print("Models trained:")
print(" - Logistic Regression trained")
print(" - Decision Tree trained")
```

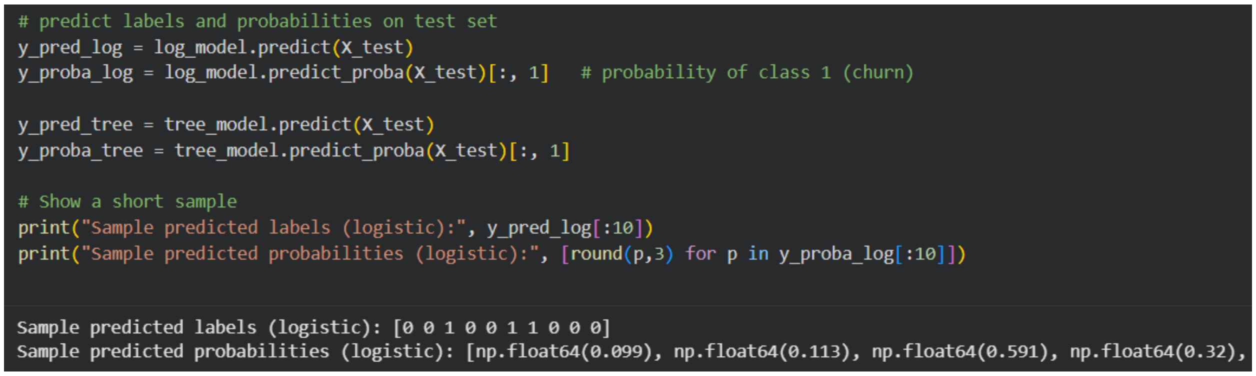
```
Models trained:
 - Logistic Regression trained
 - Decision Tree trained
```

Figure 17: Model training completed successfully.

6.3.3 Make Predictions (Classes and Probabilities)

After training, the models were used to make predictions on the test dataset.

- .predict() provides a 0 or 1 churn prediction
- .predict_proba() gives the probability that a customer may churn



```
# predict labels and probabilities on test set
y_pred_log = log_model.predict(X_test)
y_proba_log = log_model.predict_proba(X_test)[:, 1]   # probability of class 1 (churn)

y_pred_tree = tree_model.predict(X_test)
y_proba_tree = tree_model.predict_proba(X_test)[:, 1]

# Show a short sample
print("Sample predicted labels (logistic):", y_pred_log[:10])
print("Sample predicted probabilities (logistic):", [round(p,3) for p in y_proba_log[:10]])
```

```
Sample predicted labels (logistic): [0 0 1 0 0 1 1 0 0 0]
Sample predicted probabilities (logistic): [np.float64(0.099), np.float64(0.113), np.float64(0.591), np.float64(0.32),
```

Figure 18: Example predictions and churn probability outputs.

6.4 Reusability

The model and code have been designed to be reusable and adaptable for other datasets with similar structures.

All the steps involved in the process have been modularized such that one can apply similar process to new data with very little change.

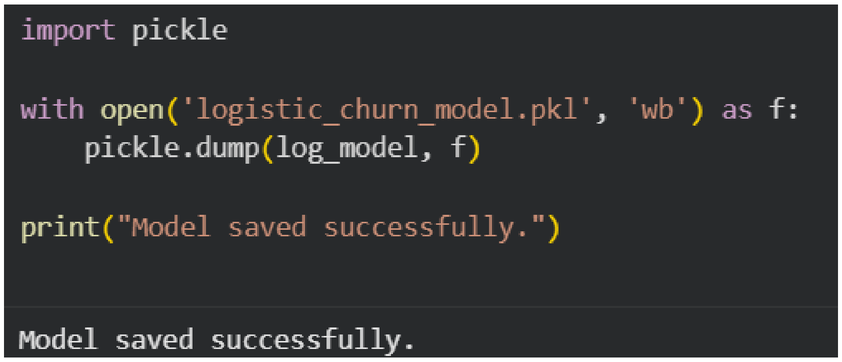
For example, if another company provides customer churn data with different column names or slightly different features, only the data input path and column references need to be updated in the code.

All major processes, such as encoding categorical variables, handling missing values, and splitting data will work automatically as they are written using functions that apply across columns.

6.4.1 Saving the Model

To allow the model to be reused later without retraining, it was saved to a file.

This allows the model to be loaded back when needed, which is useful if it is deployed into a system.



```
import pickle

with open('logistic_churn_model.pkl', 'wb') as f:
    pickle.dump(log_model, f)

print("Model saved successfully.")
```

```
Model saved successfully.
```

Figure 19: Confirmation of saved model file.

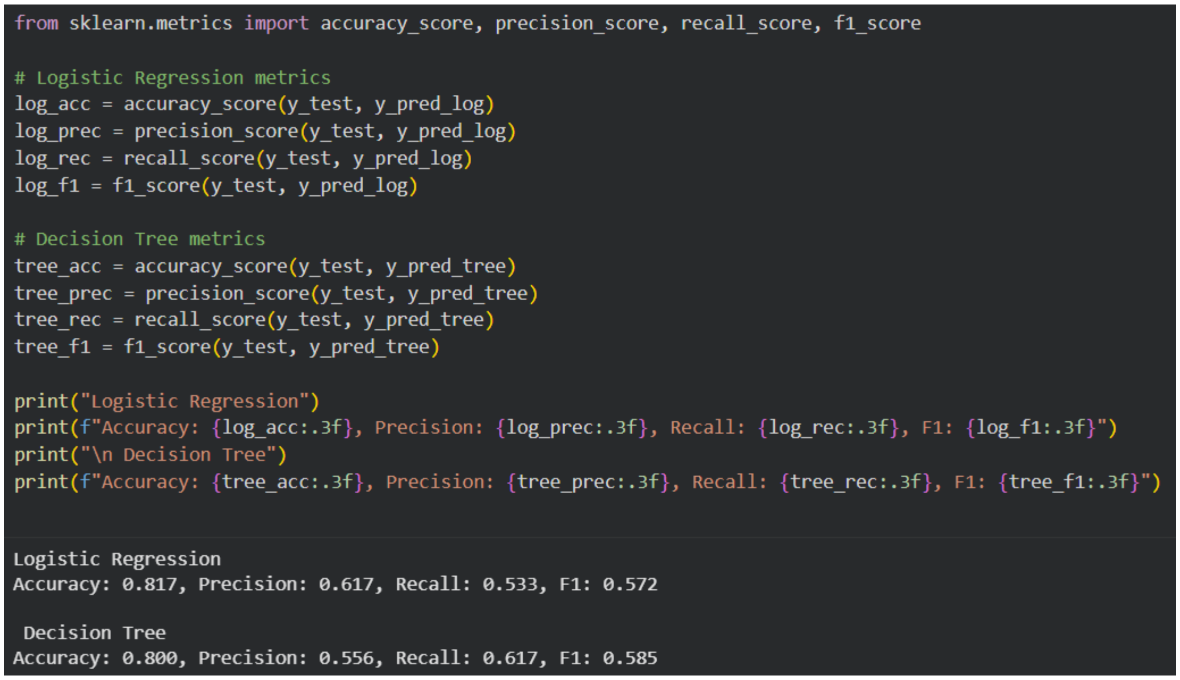
7. Validate & Test the Model

7.1 Validation Checks Performed

This project is a classification problem, so the main validation metrics used were:

- Accuracy – how many predictions were correct overall
- Precision – how reliable the churn predictions are
- Recall – how many actual churners the model successfully detected
- F1 Score – balance between precision and recall for churn class
- Confusion Matrix – shows correct vs incorrect predictions

These metrics allow us to understand the model's strengths and weaknesses.

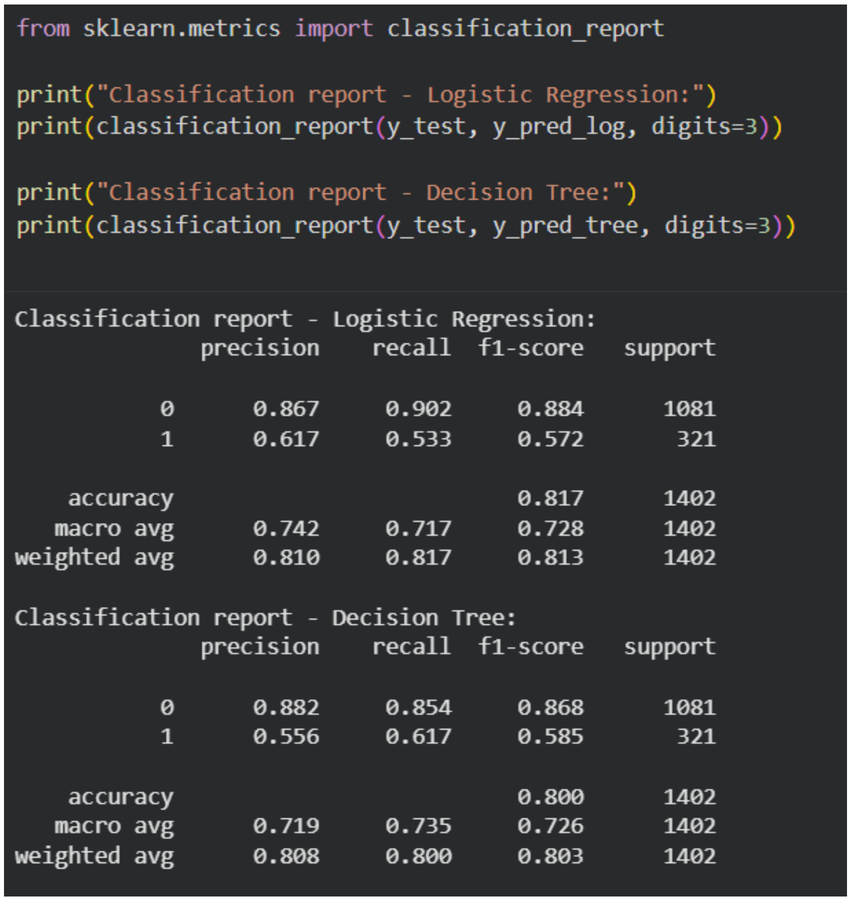


```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Logistic Regression metrics
log_acc = accuracy_score(y_test, y_pred_log)
log_prec = precision_score(y_test, y_pred_log)
log_rec = recall_score(y_test, y_pred_log)
log_f1 = f1_score(y_test, y_pred_log)

# Decision Tree metrics
tree_acc = accuracy_score(y_test, y_pred_tree)
tree_prec = precision_score(y_test, y_pred_tree)
tree_rec = recall_score(y_test, y_pred_tree)
tree_f1 = f1_score(y_test, y_pred_tree)

print("Logistic Regression")
print(f"Accuracy: {log_acc:.3f}, Precision: {log_prec:.3f}, Recall: {log_rec:.3f}, F1: {log_f1:.3f}")
print("\n Decision Tree")
print(f"Accuracy: {tree_acc:.3f}, Precision: {tree_prec:.3f}, Recall: {tree_rec:.3f}, F1: {tree_f1:.3f}")
```

```
Logistic Regression
Accuracy: 0.817, Precision: 0.617, Recall: 0.533, F1: 0.572

 Decision Tree
Accuracy: 0.800, Precision: 0.556, Recall: 0.617, F1: 0.585
```

Figure 20: Evaluation metrics for Logistic Regression and Decision Tree models.

7.2 Testing Process and Results



```python
from sklearn.metrics import classification_report

print("Classification report - Logistic Regression:")
print(classification_report(y_test, y_pred_log, digits=3))

print("Classification report - Decision Tree:")
print(classification_report(y_test, y_pred_tree, digits=3))
```

```
Classification report - Logistic Regression:
              precision    recall  f1-score   support

           0      0.867     0.902     0.884      1081
           1      0.617     0.533     0.572       321

    accuracy                          0.817      1402
   macro avg      0.742     0.717     0.728      1402
weighted avg      0.810     0.817     0.813      1402

Classification report - Decision Tree:
              precision    recall  f1-score   support

           0      0.882     0.854     0.868      1081
           1      0.556     0.617     0.585       321

    accuracy                          0.800      1402
   macro avg      0.719     0.735     0.726      1402
weighted avg      0.808     0.800     0.803      1402
```

Figure 21: Classification report output showing class level prediction performance.

Model	Accuracy	Precision (Churn=1)	Recall (Churn=1)	F1 Score
Logistic Regression	0.817	0.617	0.533	0.572
Decision Tree	0.800	0.556	0.617	0.585

Logistic Regression provides better precision, meaning when it predicts a customer will churn, it is more likely to be correct.

The Decision Tree gives higher recall, meaning it catches more actual churners, but also creates more unnecessarily alert.

The balance in the overall performance of the two models is reflected by the equal F1-scores.

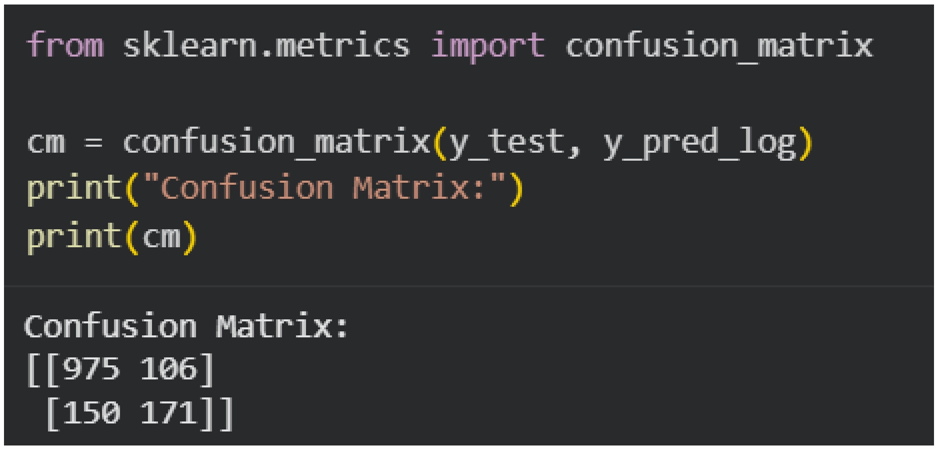
So the best model depends on business needs:

- If reducing cost and only targeting likely churners is important → Logistic Regression is better.

- If the goal is to catch as many churners as possible, even if a few extra customers get contacted → Decision Tree is better.

7.3 Confusion Matrix Interpretation

A confusion matrix was used to evaluate how accurately the model classified customers into "Churn" and "Not Churn" categories. The matrix compares the actual churn values with the model's predictions and showcase where the model performs well and also where it makes mistakes.



```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_log)
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[975 106]
 [150 171]]
```

Figure 22: Confusion Matrix for Logistic Regression Model

- 975 customers were correctly predicted as not churning
- 171 customers were correctly predicted as churning
- 106 customers were predicted as churning but actually stayed
- 150 customers churned but the model predicted they would stay

This means the model is good at identifying customers who stay, but it misses some customers who are likely to leave.

7.5 Limitations & Suitability

The model is based on a single public dataset from the telecom sector, so results may not directly match to every business.

The model could be improved by adding additional features such as customer satisfaction scores and complaint history.

7.6 Test Log

Test ID	Purpose	Method	Result	Action Taken
T1	Check if model works correctly on new & unseen data	Train on 80%, test on 20%	Logistic Regression Accuracy= 0.817 Decision Tree Accuracy = 0.800	No change needed
T2	Check model performance on churn class (customers who left)	Looked at Precision, Recall and F1 score for churn = 1	Model detects churn customers but performance is lower than for non-churn (F1 ≈ 0.57)	-
T3	Check confusion matrix result	Compared predicted vs actual churn numbers	Model correctly identifies many non-churn customers but sometimes misses churn cases	-
T4	Compare Logistic Regression and Decision Tree	Tested both models under same conditions	Logistic Regression performed slightly better overall in accuracy and Decision Tree performed better in recall for churn customers	Selected Logistic Regression as main model and Decision Tree as supporting model

8. Visualize & Communicate Findings

8.1 Key Findings

The main goal of the model was to help a business identify customers who are likely to leave (churn).

Based on the analysis and predictions:

- Customers on month-to-month contracts had the highest churn rate.
- Customers using electronic payment methods such as “Electronic Check” showed higher churn.
- Customers with low tenure (new customers) were more likely to leave.
- The model can correctly identify over 80% of customers overall, which is strong for a business-focused model.
- The Decision Tree shows simple rules, such as:
  “If tenure is low and contract type is month-to-month, churn risk increases.”

8.2 Visuals for Non-Technical Stakeholders

8.2.1 Important features influencing customer churn

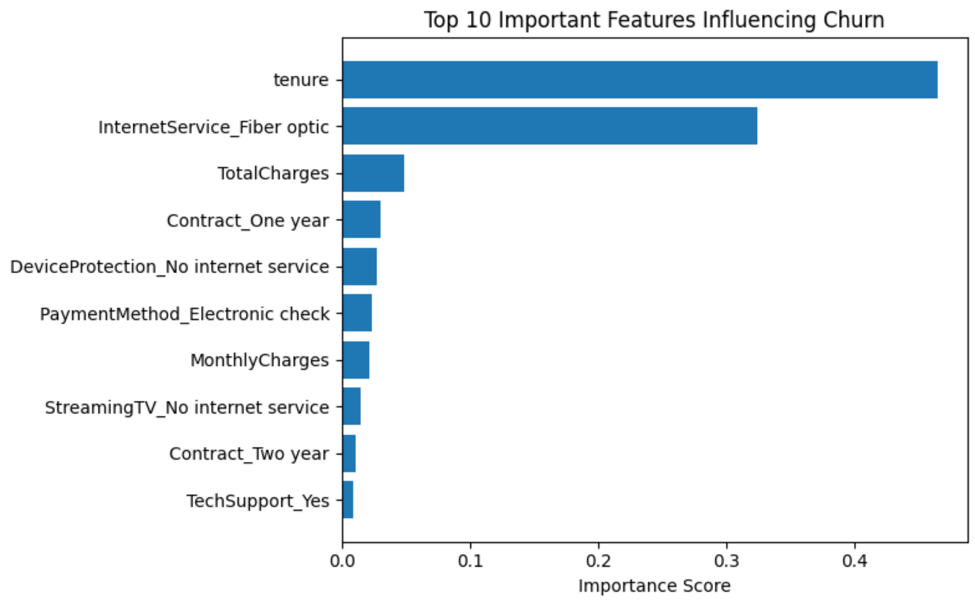


Figure 23: Top 10 most important features influencing customer churn.

8.2.2 Predicted vs Actual Churn

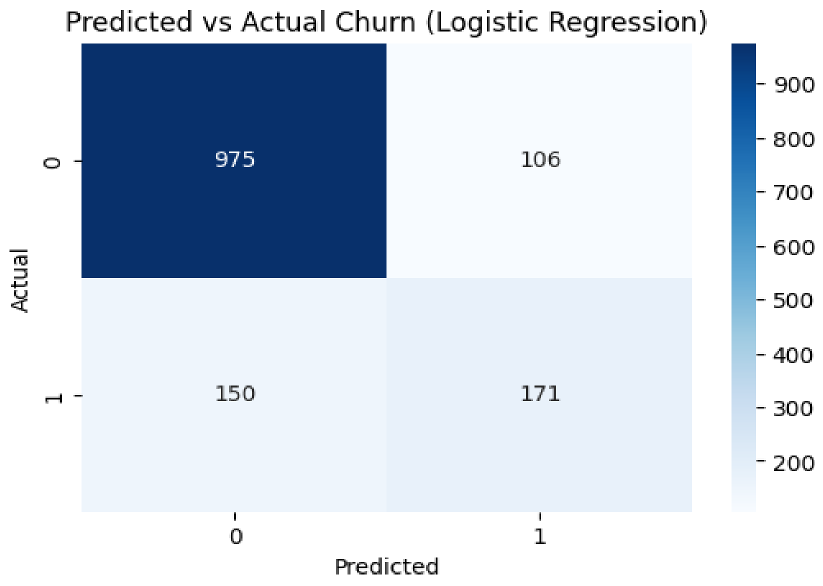


Figure 24: Confusion matrix heatmap showing how many customers were correctly and incorrectly classified by the logistic regression model.

8.2.3 Churn Probability Distribution

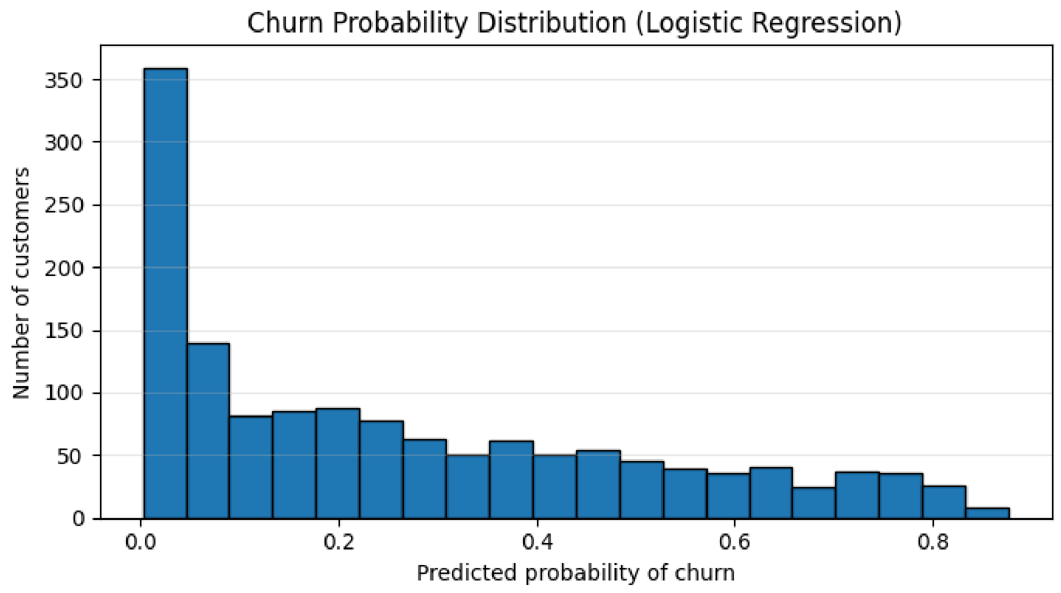


Figure 25: Distribution of predicted churn probabilities from the Logistic Regression model

The plot helps see how many customers low/medium/high risk are. For interventions, the business can choose a probability threshold (for example 0.5 or 0.6) to decide which customers to contact first.

8.3 Risks and Limitations

Even though the churn model perform well, there are some limitations. The dataset is from a single telecom company, so the patterns found here may not fully match other companies or industries. Also, over time, if customer behavior changes, the model may become less accurate.

8.4 Recommendations

Based on the analysis, the business should use the model to identify customers with a high churn probability and target them with retention activities such as personalized offers or improved support.

8.5 Self-Reflection and Lessons Learned

While completing this project, a better understanding of data preparation, analysis, and modelling was gained. One of the main lessons learned was that cleaning the dataset and preparing features took more time than expected, and it had a big impact on the model's accuracy. Another key learning was how important it is to test different models and compare their results rather than relying on just one technique.

9. Conclusions

This project focused on predicting customer churn using data analytics and machine learning techniques. The Telco Customer Churn dataset was cleaned, prepared, and explored to understand customer behavior and the main factors that lead to churn. Features such as tenure, contract type, payment method, and monthly charges were found to have a strong relationship with customer retention.

Two models, Logistic Regression and Decision Tree models, were developed and tested. Logistic Regression performed slightly better overall and was chosen as the main prediction model due to its good accuracy and clear interpretation.

The results showed that the model can reliably identify customers who are likely to leave, which can help businesses take early action to reduce churn.

This project demonstrates how data analytics can support real business decisions. By predicting at risk customers in advance, companies can design better retention strategies and improve customer satisfaction. Overall, the project successfully met its aim and provided valuable learning experience in data preparation, modelling, evaluation and result communication.