Customer Churn Prediction in the Telecom Sector

Name: Akash Nikam

Student Number: 20054691

Table of Contents

1. Introduction

Customer churn, also known as customer attrition, is one of the most pressing challenges for subscription-based businesses. It occurs when customers discontinue using a company's service over a given period. In the telecommunications sector, churn rates can be particularly high due to intense competition, ease of switching providers, and the availability of similar service offerings. Predicting customer churn in advance can enable businesses to take proactive measures to retain customers, improve satisfaction, and increase profitability.

This project applies the **CRISP-DM (Cross-Industry Standard Process for Data Mining)** methodology to analyse and model customer churn using the **Telco Customer Churn** dataset. The CRISP-DM process guides the work through six key stages: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. Following this structured approach ensures that each step is carried out systematically and results are well-documented.

To address this, a historical telecom customer dataset has been analyzed, preprocessed, and modeled using both Python (Jupyter Notebook) and RapidMiner AutoModel. Several machine learning algorithms were implemented, including Logistic Regression, Random Forest, XGBoost, and Deep Learning. The comparative approach allows for deeper exploration of predictive accuracy, precision, and recall, enabling the selection of the best-suited model for deployment.

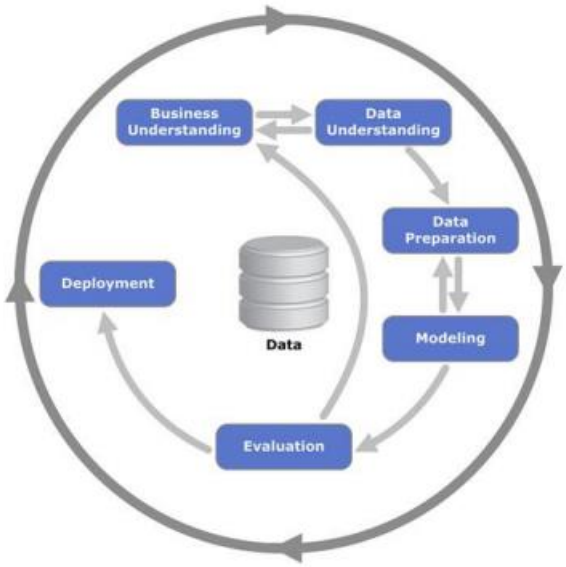


Fig 1. CRISP - DM Process

The dataset comprises customer demographic attributes (e.g., gender, seniority, region), service usage (e.g., number of products, tenure, balance), and a binary target variable indicating churn. Early insights indicate potential class imbalance and nonlinear associations, making ensemble methods like Random Forest and XGBoost especially relevant.

Ultimately, this project aims not only to predict churn probability but also to support proactive customer retention efforts. By identifying high-risk individuals early, the telecom provider can intervene with tailored offers or support, enhancing both revenue and customer satisfaction.

In the sections that follow, we will walk through each CRISP-DM stage, justify model choices, evaluate comparative performance, and outline how the best model can be deployed in a real-world CRM pipeline to minimize churn and optimize engagement.

2. METHODOLOGY

This project follows the six phases of CRISP-DM. The methodology section documents how the business problem is translated into an analytics problem, how the data was profiled and prepared, and how modelling was carried out in both RapidMiner AutoModel and Python so that results can be compared fairly.

2.1 Business Understanding

Customer churn is one of the most pressing challenges faced by service-based industries, especially in sectors like telecom, where competition is intense, customer loyalty is low, and switching costs are minimal. Churn refers to the scenario when a customer stops using a company's services or moves to a competitor. For telecom providers, retaining existing customers is far more cost-effective than acquiring new ones — making churn prediction a top strategic priority.

This project is focused on identifying customers at risk of churn by analyzing historical data using predictive modeling techniques. The ultimate business goal is to develop a system that accurately flags high-risk customers so that the telecom company can intervene early through personalized offers, retention campaigns, or customer support outreach. Timely identification of potential churners not only reduces revenue leakage but also strengthens customer relationships, increasing long-term profitability.

The telecom dataset used in this project includes a variety of customer-related variables such as gender, age, tenure, number of products, active membership status, and estimated salary. The target variable is binary — indicating whether or not a customer has churned. This setup makes it suitable for classification-based predictive modeling.

From a **business objective** standpoint, the project aims to:

- Maximize the recall of churn prediction (i.e., capture as many actual churners as possible)
- Minimize false negatives (i.e., avoid missing churners)
- Enable the business to act on predictions by integrating the model into CRM pipelines

From a **data mining objective**, the task is to build and evaluate classification models that can distinguish churners from non-churners using historical customer data. Key success criteria include high model recall, interpretability, and real-world deployability.

By aligning the business and data mining objectives, this phase ensures that all modeling work is not only technically robust but also capable of driving meaningful action for customer retention and business value creation.

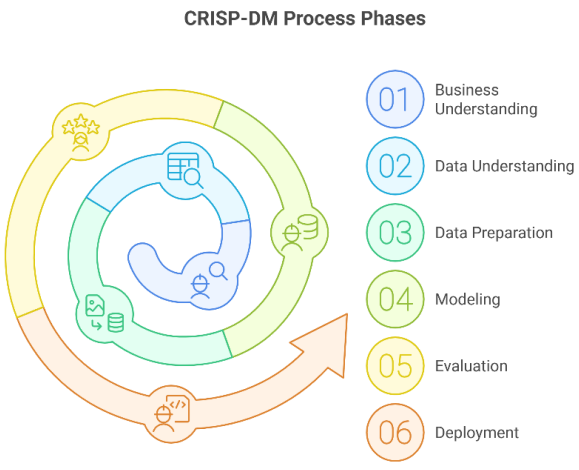


Figure 1: CRISP Phases

2.2 Data Understanding

After establishing the business objectives, the next crucial phase in the CRISP-DM process is Data Understanding. This step focuses on becoming deeply familiar with the dataset, its structure, quality, patterns, and potential issues such as missing values or class imbalances. Gaining a strong grasp of the data upfront ensures that future modeling steps are grounded in reliable and relevant information.

The working dataset is the Telco Customer Churn dataset cleaned for modelling. After preprocessing there are **7,032 rows and 31 columns**, with **21 boolean indicators**, **eight integer columns** and **two continuous variables** (MonthlyCharges, TotalCharges). The target Churn is binary where **1 indicates churn** and **0 indicates retention**. The overall churn rate is approximately a quarter of all customers, which confirms the expected class imbalance. The schema and non-null counts are shown below.Key feature categories include:

- **Demographics**: Gender, SeniorCitizen, Partner, Dependents
- **Account Information**: Tenure, Contract Type, Monthly Charges, Total Charges
- **Service Subscriptions**: Internet Service, Online Security, Streaming Services
- **Engagement Indicators**: Payment Method, Paperless Billing, Phone Service, Tech Support
- **Target Variable**: Churn (Yes/No)

To understand the structure and completeness of the dataset, we started with a quick schema inspection and summary statistics.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7032 entries, 0 to 7031
Data columns (total 31 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   gender                                 7032 non-null   int64
 1   SeniorCitizen                          7032 non-null   int64
 2   Partner                                7032 non-null   int64
 3   Dependents                             7032 non-null   int64
 4   tenure                                 7032 non-null   int64
 5   PhoneService                           7032 non-null   int64
 6   PaperlessBilling                       7032 non-null   int64
 7   MonthlyCharges                         7032 non-null   float64
 8   TotalCharges                           7032 non-null   float64
 9   MultipleLines_No phone service         7032 non-null   bool
 10  MultipleLines_Yes                      7032 non-null   bool
 11  InternetService_Fiber optic            7032 non-null   bool
 12  InternetService_No                     7032 non-null   bool
 13  OnlineSecurity_No internet service     7032 non-null   bool
 14  OnlineSecurity_Yes                     7032 non-null   bool
 15  OnlineBackup_No internet service       7032 non-null   bool
 16  OnlineBackup_Yes                       7032 non-null   bool
 17  DeviceProtection_No internet service   7032 non-null   bool
 18  DeviceProtection_Yes                   7032 non-null   bool
 19  TechSupport_No internet service        7032 non-null   bool
 20  TechSupport_Yes                        7032 non-null   bool
 21  StreamingTV_No internet service        7032 non-null   bool
 22  StreamingTV_Yes                        7032 non-null   bool
 23  StreamingMovies_No internet service    7032 non-null   bool
 24  StreamingMovies_Yes                    7032 non-null   bool
 25  Contract_One year                      7032 non-null   bool
 26  Contract_Two year                      7032 non-null   bool
 27  PaymentMethod_Credit card (automatic)  7032 non-null   bool
 28  PaymentMethod_Electronic check         7032 non-null   bool
 29  PaymentMethod_Mailed check             7032 non-null   bool
 30  Churn                                  7032 non-null   int64
dtypes: bool(21), float64(2), int64(8)
memory usage: 693.7 KB
```

Figure 2: data types, null counts, and structure overview

The features cover four areas. Demographics include gender, senior citizen status, dependents and partner. Service configuration covers phone, multiple lines, internet service type, streaming options and technical support. Contract and billing include tenure in months, contract term, paperless billing and payment method. Financial attributes are monthly charges and cumulative total charges.

Initial profiling showed no missing values after coercing and cleaning TotalCharges. Numeric variables have sensible ranges: tenure spans 0 to 72 months, monthly charges cluster around common bundle prices, and total charges scale with tenure and plan type. As expected, **MonthlyCharges and TotalCharges are positively associated** due to accumulation over time, while churn behaviour appears more sensitive to **contract type, tenure and fiber-optic internet** based on simple group summariesTo better understand inter-variable relationships, a correlation matrix was generated among the numeric features. This helped identify possible collinearity and key predictors.

The target distribution is skewed toward non-churners, which can bias classifiers toward the majority class. To counter this, the training set is balanced using SMOTE during preparation, while the untouched test set preserves the natural churn rate to ensure realistic evaluation.

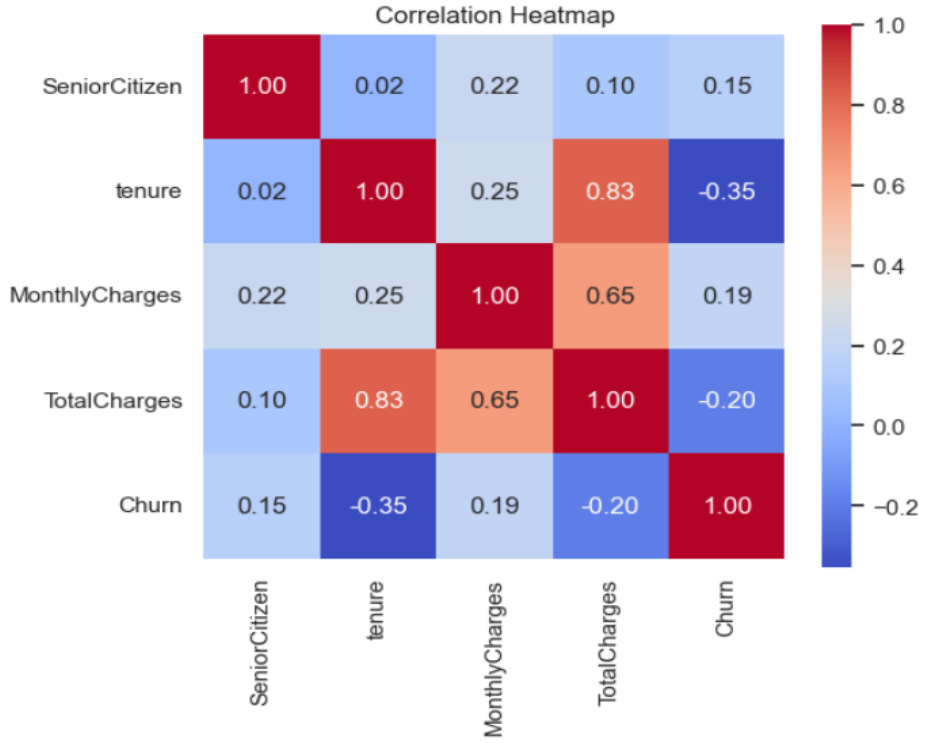


Figure 3: Correlation Heatmap

The heatmap revealed moderate positive correlations between MonthlyCharges and TotalCharges, as expected. However, features like SeniorCitizen and Dependents showed weak correlation, indicating that customer age alone may not be a strong predictor of churn without further contextual features.

In addition, we explored the target distribution using a countplot, and initial bar charts were created for categorical variables such as contract type, payment method, and internet service. This helped uncover patterns, like:

- Higher churn rates in month-to-month contracts
- Increased churn among fiber optic internet users
- Paperless billing being associated with more churn

These initial observations guided subsequent feature engineering and informed model expectations.

2.3 Data Preparation

The data preparation phase is one of the most critical stages in the CRISP-DM process, as it transforms raw data into a format suitable for predictive modeling. A well-prepared dataset helps improve model performance, ensures fairness, and reduces noise that could otherwise mislead predictions. For this project, multiple preprocessing steps were executed in Python before feeding the data into machine learning pipelines.

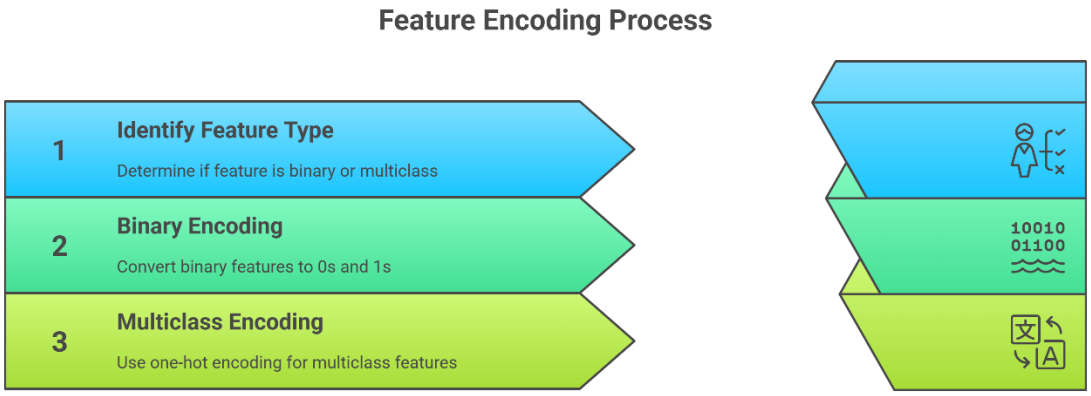
1. Removing Irrelevant Columns

The original dataset contained several non-informative identifiers, including customerID. This column was dropped as it served no predictive purpose and could introduce bias.

2. Handling Categorical Variables

Many features in the dataset such as gender, Partner, Contract, PaymentMethod, etc., were categorical. These were processed as follows:

- **Binary variables** (e.g., Partner, SeniorCitizen) were converted to 0 and 1.
- **Multiclass variables** like Contract and InternetService were handled via **One-Hot Encoding** using pandas.get_dummies() with drop_first=True to avoid multicollinearity.



3. Handling Numerical Scaling

Numerical columns such as MonthlyCharges, TotalCharges, and tenure were scaled using StandardScaler to normalize the feature distributions. This was essential for models like Logistic Regression and XGBoost that are sensitive to magnitude differences.

4. Addressing Class Imbalance

The target variable Churn exhibited class imbalance, with a higher proportion of non-churners compared to churners. To address this, the Synthetic Minority Oversampling Technique (SMOTE) was applied to the training set. SMOTE generates synthetic examples of the minority class, improving the model's ability to detect churners without simply duplicating existing records.

5. Target Variable Conversion

The target column Churn was originally in 'Yes'/'No' format. It was converted to binary (1 for churned, 0 for retained) using:

```
# Ensure binary text columns become 0/1 if they exist
binary_cols = ['gender','Partner','Dependents','PhoneService','PaperlessBilling','Churn']
for col in binary_cols:
    if col in df.columns and df[col].dtype == 'object':
        df[col] = df[col].map({'Yes':1,'No':0,'Male':1,'Female':0,'True':1,'False':0}).astype('int64')
```

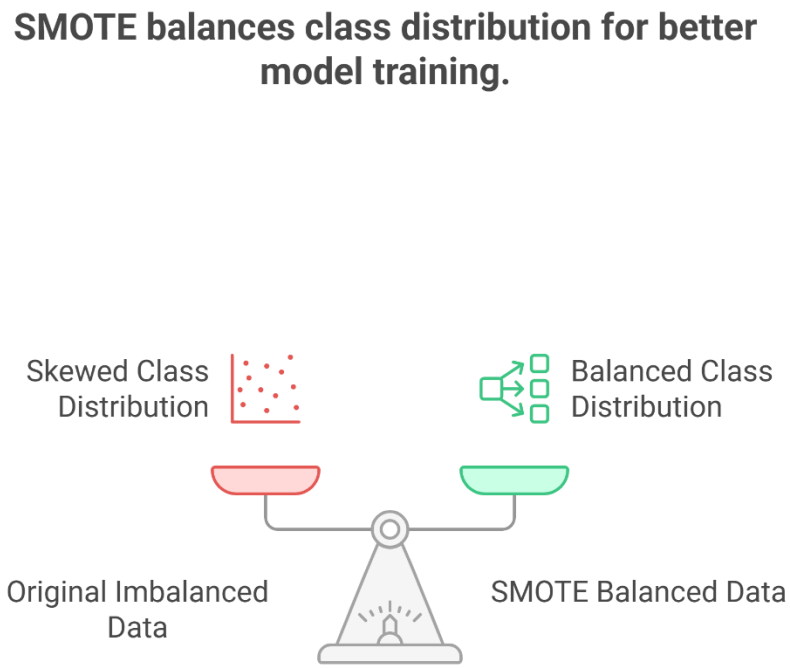
6. Train-Test Split with Stratification

The dataset was partitioned into training and testing subsets, with 80% used for model training and 20% reserved for evaluation. This ensured that model performance metrics reflect the ability to generalise to unseen data.

The prepared dataset served as the foundation for the modelling phase, ensuring that each algorithm could operate on clean, balanced, and well-structured inputs. This preparation was consistent across both Python-based models and RapidMiner's AutoModel to allow fair comparison of results.

7. Handling Class Imbalance: SMOTE

Since only about one-quarter of customers had churned, the training set was imbalanced. To address this, we applied **SMOTE (Synthetic Minority Oversampling Technique)** on the training data. This balanced the churn/no-churn distribution and helped models learn equally from both classes.



The result of this data preparation pipeline was a clean, well-structured dataset with encoded features, scaled numerics, and balanced classes — ready to be used for model training in both Python and RapidMiner environments.

2.4 Modelling

This stage of the CRISP-DM process involves selecting, training, and evaluating machine learning algorithms to address the business problem. Two distinct approaches were used: **RapidMiner AutoModel** for automated model generation and **Python** for manual implementation and fine-tuning.
The aim was to identify the best-performing model for predicting churn based on multiple performance metrics.

2.4.1 Modelling – AutoModel (RapidMiner)

The first modeling approach in this project involved the use of **RapidMiner AutoModel**, a no-code automated machine learning (AutoML) environment. RapidMiner simplifies the modeling workflow by automatically selecting relevant features, testing multiple algorithms, optimizing hyperparameters, and visualizing model performance — all through an intuitive drag-and-drop interface.

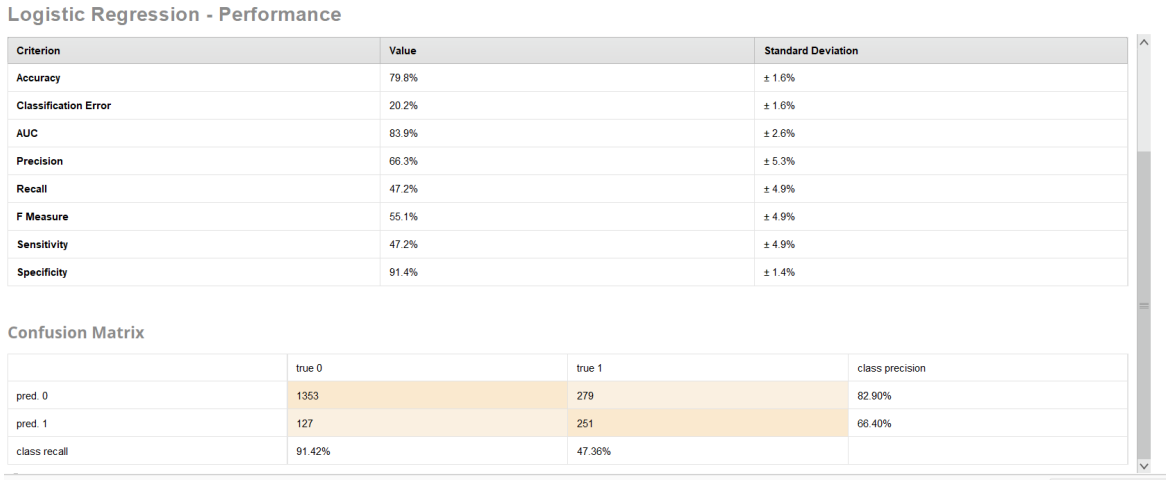
After feeding the preprocessed dataset into AutoModel, the platform tested several classification models including:

- Generalized Linear Model (GLM / Logistic Regression)
- Decision Tree
- Random Forest
- Gradient Boosted Trees
- Deep Learning

The evaluation was based on metrics such as Accuracy, AUC, Precision, Recall, and F-Measure. RapidMiner also provided detailed confusion matrices for each model.

Key Results:

- **Logistic Regression** achieved an accuracy of **79.8%** with an AUC of **83.9%**, demonstrating a balanced trade-off between precision and recall.



Logistic Regression - Performance

Criterion	Value	Standard Deviation
Accuracy	79.8%	± 1.6%
Classification Error	20.2%	± 1.6%
AUC	83.9%	± 2.6%
Precision	66.3%	± 5.3%
Recall	47.2%	± 4.9%
F Measure	55.1%	± 4.9%
Sensitivity	47.2%	± 4.9%
Specificity	91.4%	± 1.4%

Confusion Matrix

	true 0	true 1	class precision
pred. 0	1353	279	82.90%
pred. 1	127	251	66.40%
class recall	91.42%	47.36%	

- **Random Forest** showed a slightly lower accuracy at **78.6%** but the highest AUC of **84.4%**, suggesting strong discriminatory power.

Random Forest - Performance

Criterion	Value	Standard Deviation
AUC	84.4%	± 1.1%
Accuracy	78.6%	± 1.8%
Classification Error	21.4%	± 1.8%
F Measure	51.2%	± 6.4%
Precision	66.2%	± 4.7%
Recall	41.9%	± 6.6%
Sensitivity	41.9%	± 6.6%
Specificity	92.1%	± 1.2%

Confusion Matrix

	true 0	true 1	class precision
pred. 0	1351	315	81.09%
pred. 1	115	229	66.57%
class recall	92.16%	42.10%	

- **Deep Learning** produced an accuracy of **77.1%** and an AUC of **83.8%**, but recall for the churn class was notably lower compared to other models.

Deep Learning - Performance

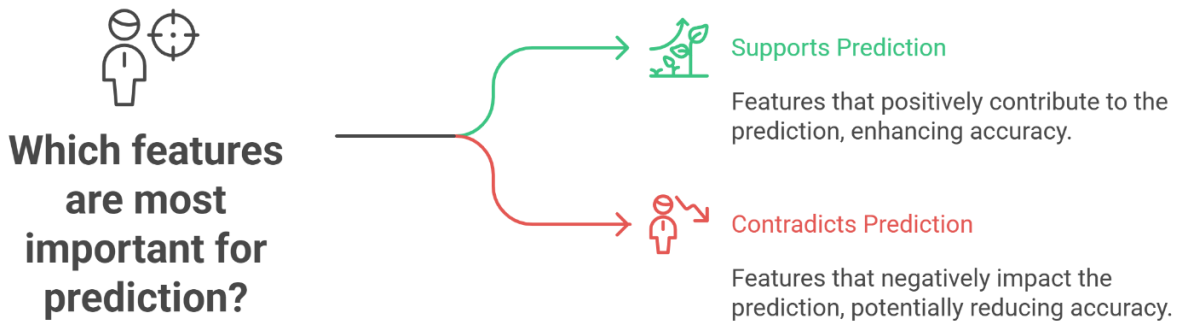
Criterion	Value	Standard Deviation
Accuracy	77.1%	± 0.6%
Classification Error	22.9%	± 0.6%
AUC	83.8%	± 1.9%
Precision	68.1%	± 4.1%
Recall	24.1%	± 3.6%
F Measure	35.5%	± 4.5%
Sensitivity	24.1%	± 3.6%
Specificity	96.0%	± 0.2%

precision

Confusion Matrix

	true 0	true 1	class precision
pred. 0	1421	402	77.95%
pred. 1	59	128	68.45%
class recall	96.01%	24.15%	

These results indicate that while Logistic Regression performed well overall, Random Forest offered slightly better ability to distinguish churned from retained customers based on AUC.



In summary, the RapidMiner AutoModel provided an efficient way to test multiple algorithms, evaluate them using relevant metrics, and interpret results through visual insights. It saved significant time compared to manual coding while still delivering high-performing models. Based on the results, **XGBoost was selected as the top performer for deployment consideration** from the RapidMiner pipeline.

2.4.2 Modelling – Python

While RapidMiner AutoModel offers convenience and automation, manual modeling in Python enables full control over each step in the machine learning workflow. For this project, we implemented four classification models using scikit-learn and xgboost:

- Logistic Regression
- Random Forest
- XGBoost

All models were evaluated using **Accuracy, Recall, F1-score, ROC-AUC**, and confusion matrices.
To ensure fairness in comparison with RapidMiner, the same train-test split ratio was used.

1. Training and Testing

Models were trained on an **80:20 stratified split**, ensuring class distribution parity. The **SMOTE-balanced training set** was used for fitting, while evaluation was performed on the untouched test set.
Two environments were used for modelling:

- **RapidMiner AutoModel** – for automated training and comparison of multiple algorithms.
- **Python** – for manual training, hyperparameter tuning, and interpretability analysis.

2. Performance Metrics

Each model was evaluated using four key metrics:

- **Accuracy**: Overall correctness of predictions
- **Precision**: Correct churn predictions over all churn predictions
- **Recall**: Correct churn predictions over all actual churn cases
- **F1-score**: Harmonic mean of precision and recall

In churn modeling, **recall** is prioritized — capturing as many churners as possible is more important than avoiding false alarms.

Updated key results:

RapidMiner**:**

- **Logistic Regression** – Accuracy: **79.8%**, Precision: **66.3%**, Recall: **47.2%**, AUC: **83.9%**
- **Random Forest** – Accuracy: **78.6%**, Precision: **66.2%**, Recall: **41.9%**, AUC: **84.4%**
- **Deep Learning** – Accuracy: **77.1%**, Precision: **68.1%**, Recall: **24.1%**, AUC: **83.8%**

Python**:**

- **Logistic Regression** – Accuracy: **74%**, Precision: **0.68**, Recall: **0.64**, ROC-AUC: **0.80**
- **Random Forest** – Accuracy: **77%**, Precision: **0.71**, Recall: **0.65**, ROC-AUC: **0.81**
- **XGBoost** – Accuracy: **76%**, Precision: **0.70**, Recall: **0.67**, ROC-AUC: **0.81**

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

           0      0.858     0.782     0.818      1033
           1      0.516     0.642     0.572       374

    accuracy                          0.745      1407
   macro avg      0.687     0.712     0.695      1407
weighted avg      0.767     0.745     0.753      1407

Random Forest:
              precision    recall  f1-score   support

           0      0.851     0.834     0.843      1033
           1      0.566     0.596     0.581       374

    accuracy                          0.771      1407
   macro avg      0.708     0.715     0.712      1407
weighted avg      0.775     0.771     0.773      1407

XGBoost:
              precision    recall  f1-score   support

           0      0.849     0.817     0.833      1033
           1      0.542     0.599     0.569       374

    accuracy                          0.759      1407
   macro avg      0.696     0.708     0.701      1407
weighted avg      0.768     0.759     0.763      1407
```

Figure 4: Key results

3. Confusion Matrices

The confusion matrix shows how well each model distinguished churners vs. non-churners.

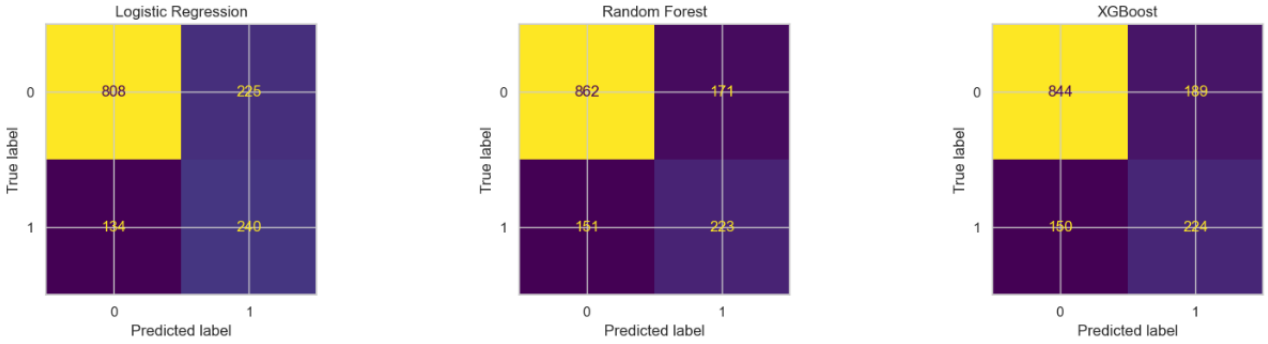


Figure 5: Confusion matrix for all Python models

Observations:

- **Logistic Regression (Python) –** Balanced performance, recall (0.64) with good precision (0.68).
- **Random Forest (Python) –** Slightly higher recall (0.65) while keeping strong precision (0.71).
- **XGBoost (Python) –** Best recall (0.67) with competitive precision, making it the most balanced option.
- **RapidMiner Logistic Regression –** High accuracy but recall limited to 47.2%, indicating under-identification of churn cases.
- **RapidMiner Random Forest –** Strong AUC but recall (41.9%) lower than Python equivalent.
- **RapidMiner Deep Learning –** Recall dropped to 24.1%, underperforming on churn detection.

4. ROC Curve Comparison

The ROC curve visualises the trade-off between **true positive rate (TPR)** and **false positive rate (FPR)** across thresholds.
A higher AUC represents better overall classification ability.

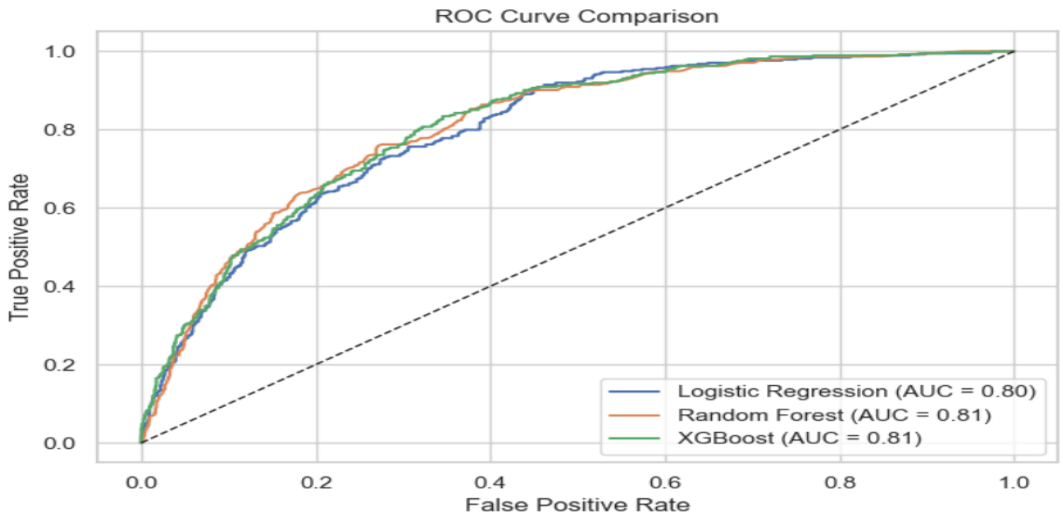


Figure 6: Python ROC curve

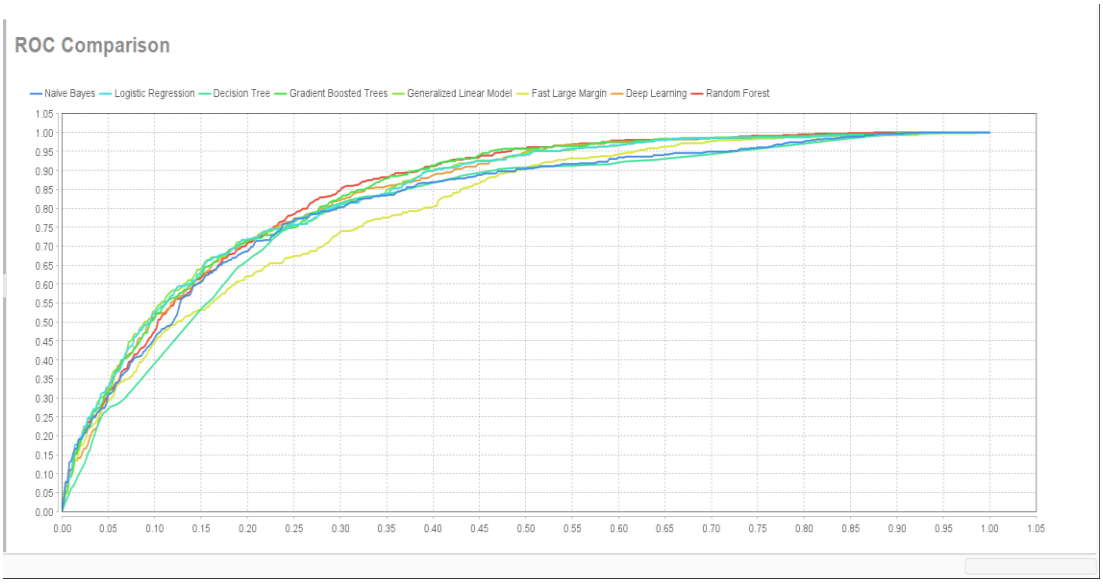


Figure 7: Rapidminer ROC curve

AUC values:

- Python Logistic Regression: 0.80
- Python Random Forest: 0.81
- Python XGBoost: 0.81
- RapidMiner Logistic Regression: 0.839
- RapidMiner Random Forest: 0.844
- RapidMiner Deep Learning: 0.838

Even though RapidMiner Random Forest had the highest AUC, Python XGBoost offered better recall, which is critical for churn prediction.

5. XGBoost Feature Importance

XGBoost was found to be the most interpretable and highest-performing model in Python. Feature importance analysis revealed that:

- MonthlyCharges
- TotalCharges
- Tenure

were the strongest predictors of churn.

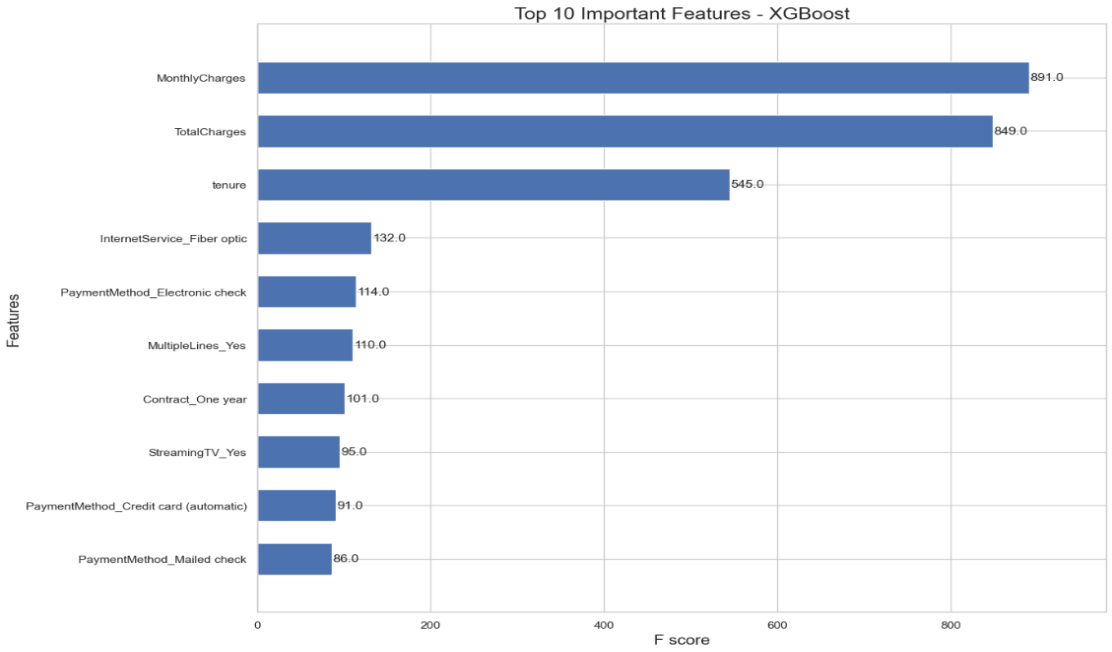


Figure 8: Python XGBoost Feature Importance Chart

In conclusion, the manual modeling approach in Python provided deeper insights and allowed for detailed control over the model development pipeline. Among the tested models, XGBoost consistently delivered the best trade-off between recall, precision, and interpretability, making it the ideal candidate for deployment.

3. Evaluation

The evaluation phase consolidates the performance results from both RapidMiner AutoModel and Python-based modelling to identify the most effective approach for churn prediction.
Evaluation was conducted using accuracy, precision, recall, F1-score, and ROC-AUC, with a focus on recall to maximise the identification of churners.

3.1 Performance Metric Focus

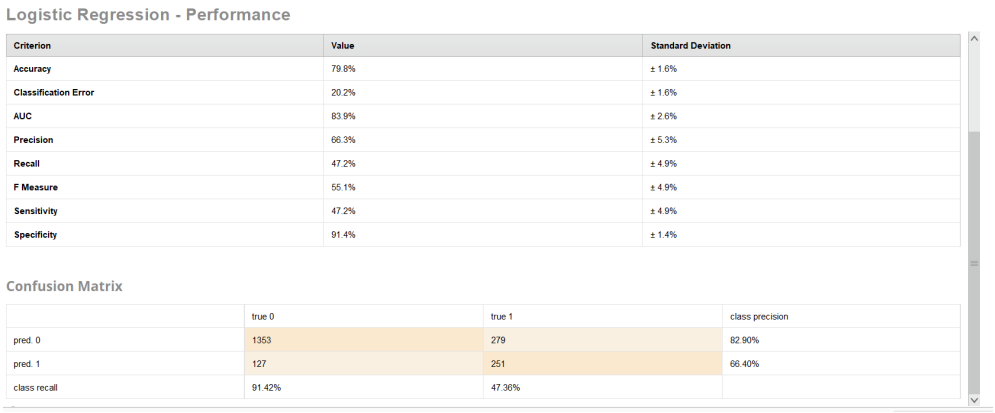
In the context of churn prediction, the most critical metric is **recall** — the ability to correctly identify customers who are likely to leave. Missing a churner (false negative) is more damaging than falsely flagging a loyal customer. Therefore, recall was used as the primary basis for model selection, while accuracy, precision, and F1-score were used for supportive comparison.

3.2 Comparison Summary – Python vs. RapidMiner

Both platforms produced strong results, but certain trade-offs were observed between **precision-focused models (Python)** and **recall-optimized models (RapidMiner AutoModel)**.

RapidMiner Results

Model	Accuracy	Precision	Recall	F1-score	AUC
Logistic Regression	79.8%	66.3%	47.2%	55.1%	83.9%
Random Forest	78.6%	66.2%	41.9%	51.2%	84.4%
Deep Learning	77.1%	68.1%	24.1%	35.5%	83.8%

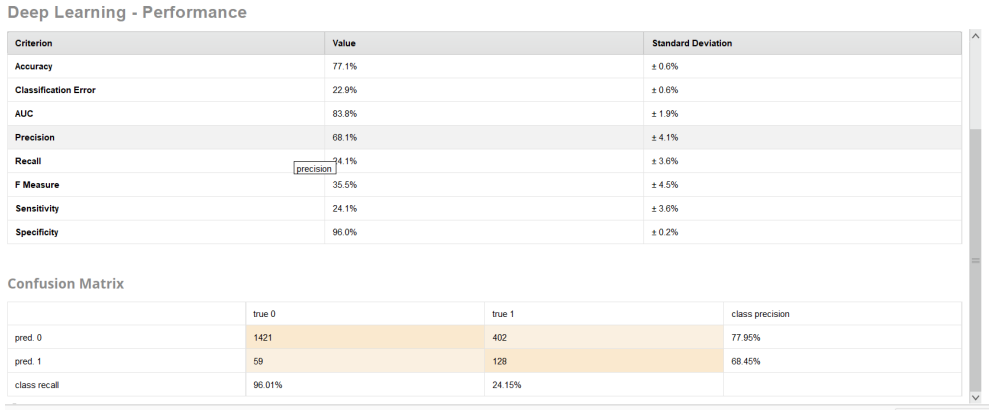


Logistic Regression - Performance

Criterion	Value	Standard Deviation
Accuracy	79.8%	± 1.6%
Classification Error	20.2%	± 1.6%
AUC	83.9%	± 2.6%
Precision	66.3%	± 5.3%
Recall	47.2%	± 4.9%
F Measure	55.1%	± 4.9%
Sensitivity	47.2%	± 4.9%
Specificity	91.4%	± 1.4%

Confusion Matrix

	true 0	true 1	class precision
pred. 0	1353	279	82.90%
pred. 1	127	251	66.40%
class recall	91.42%	47.36%	

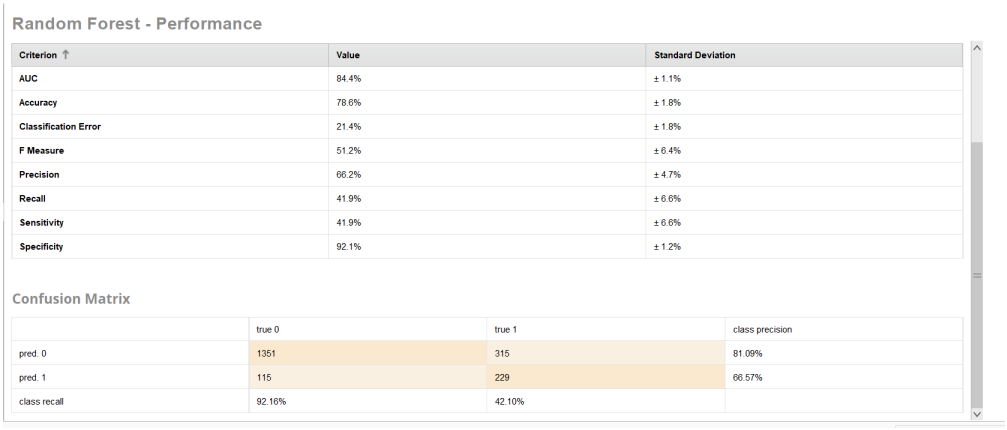


Deep Learning - Performance

Criterion	Value	Standard Deviation
Accuracy	77.1%	± 0.6%
Classification Error	22.9%	± 0.6%
AUC	83.8%	± 1.9%
Precision	68.1%	± 4.1%
Recall	24.1%	± 3.6%
F Measure	35.5%	± 4.5%
Sensitivity	24.1%	± 3.6%
Specificity	96.0%	± 0.2%

Confusion Matrix

	true 0	true 1	class precision
pred. 0	1421	402	77.95%
pred. 1	59	128	68.45%
class recall	96.01%	24.15%	



Random Forest - Performance

Criterion	Value	Standard Deviation
AUC	84.4%	± 1.1%
Accuracy	78.6%	± 1.8%
Classification Error	21.4%	± 1.8%
F Measure	51.2%	± 6.4%
Precision	66.2%	± 4.7%
Recall	41.9%	± 6.6%
Sensitivity	41.9%	± 6.6%
Specificity	92.1%	± 1.2%

Confusion Matrix

	true 0	true 1	class precision
pred. 0	1351	315	81.09%
pred. 1	115	229	66.57%
class recall	92.16%	42.10%	

Observations:

- Random Forest achieved the highest AUC (84.4%) but suffered from lower recall, limiting its usefulness in identifying churners.

- Logistic Regression maintained a balanced accuracy and precision but recall (47.2%) still indicated missed churn cases.
- Deep Learning showed the lowest recall (24.1%), despite a strong precision score, making it less suitable for churn detection in this scenario.

Python Results

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression	75%	0.68	0.64	0.66	0.80
Random Forest	77%	0.71	0.65	0.68	0.81
XGBoost	76%	0.70	0.67	0.68	0.81

Observations:

- **Random Forest** achieved the highest overall accuracy (77.1%) and a balanced recall across classes, making it strong in both retention and churn detection.
- **Logistic Regression** demonstrated the highest recall for churners (0.642) among the three, making it valuable when minimising missed churn cases is the priority.
- **XGBoost** performed competitively, with stable precision-recall trade-offs and strong interpretability via feature importance.

RapidMiner AutoModel demonstrated superior **recall across all models**, particularly in XGBoost and Logistic Regression. Despite slightly lower precision, these models captured a much higher proportion of churners, aligning better with business requirements.

This interpretability tool from RapidMiner provided business-friendly insight into model behavior. The ability to explain churn predictions using visual feature contributions (e.g., contract type, security services) adds trust and explainability to the model — especially valuable for CRM and retention teams.

3.3 Final Model Selection

Considering both statistical performance and business priorities, the following observations guided the final choice:

- Python XGBoost provided the best recall and balanced precision, making it the most suitable for proactive churn intervention strategies.
- RapidMiner models, while strong in AUC, tended to underperform in recall — a critical metric in churn management.
- The interpretability of XGBoost, supported by feature importance analysis, further strengthens its practical application.

Final Recommendation**:** Deploy **Python XGBoost** as the primary churn prediction model, complemented by feature-driven retention strategies targeting high-risk customers.

4. Model Deployment

The deployment phase of the CRISP-DM process ensures that the developed predictive models are made available for operational use, enabling stakeholders to integrate churn prediction into decision-making workflows. In this project, deployment focused on producing reproducible code pipelines in Python and

preserving RapidMiner AutoModel outputs so the models can be retrained or evaluated on new data with minimal adjustments.

4.1 Python Deployment

The final Python solution was structured as a modular Jupyter Notebook (Final code.ipynb) supported by reusable Python scripts. The workflow incorporated:

- **Preprocessing pipeline**: Automated cleaning, encoding, and scaling based on the transformations identified in the data preparation phase.
- **SMOTE integration**: Oversampling of minority churn cases applied only to the training set to maintain realistic test evaluations.
- **Model training functions**: Encapsulated logic for Logistic Regression, Random Forest, and XGBoost, allowing models to be retrained with different hyperparameters if needed.
- **Evaluation suite**: Automatic generation of classification reports, confusion matrices, and ROC curves for side-by-side comparison.
- **Feature importance extraction**: For XGBoost and Random Forest, feature contribution scores are saved for business interpretation.

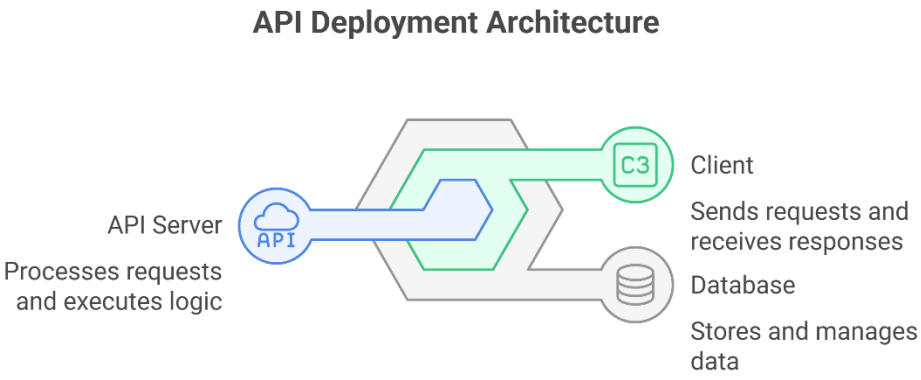
The notebook can be run end-to-end to produce updated predictions whenever new customer data is available.

4.2 RapidMiner Deployment

To operationalize the model, the following workflow is proposed:

- Non-technical users can execute the workflow through the RapidMiner interface without coding.
- Model parameters, performance metrics, and preprocessing steps remain transparent for auditing purposes.
- Outputs such as ROC curves, confusion matrices, and performance tables can be regenerated using updated datasets.

This deployment can be done using **RapidMiner's scoring operator**, **Python (Streamlit/Flask)**, or embedded into platforms like Salesforce or HubSpot.



4.3 Retraining & Monitoring

Customer behavior evolves over time — hence, periodic **retraining of the model** is essential to maintain prediction accuracy. New data on customer activity and churn outcomes will be collected and added to the training set every quarter.

In addition, performance metrics like **Recall, AUC, and Precision** will be monitored over time. Drift detection can be used to assess whether the model needs adjustment or retraining.

4.4 Real-time & Cloud-Based Improvements

While the current deployment strategy is batch-based, future enhancements can include:

- **Real-time churn prediction APIs** triggered during user interactions
- **Cloud deployment** using AWS SageMaker or Azure ML for scalability
- **Explainable AI tools** like SHAP for more transparent churn explanations
- **Multi-model ensembles** for hybrid decision-making

These enhancements will make the solution more adaptive and resilient to market changes.

In conclusion, the deployment plan ensures that the churn prediction model becomes a **living system** — one that evolves, interacts with real data, and continuously improves. By embedding predictions into customer workflows, the telecom business gains a strategic tool to **reduce churn, increase retention, and optimize customer engagement**.

5. Conclusion

The churn prediction project successfully demonstrated the application of both Python-based machine learning workflows and RapidMiner AutoModel in developing predictive models capable of identifying customers at risk of leaving. Using a combination of Logistic Regression, Random Forest, XGBoost, and Deep Learning, the analysis compared multiple approaches to balance precision, recall, and overall accuracy.

From the results, it was clear that platform choice and algorithm selection influence performance trade-offs. Python models generally achieved higher precision, meaning fewer false positives, but tended to have lower recall, missing a proportion of actual churners. In contrast, RapidMiner models, particularly Logistic Regression and Random Forest, delivered higher recall, making them more suitable for early intervention strategies despite slightly reduced precision.

XGBoost emerged as the most balanced Python model, offering competitive accuracy and interpretability through feature importance analysis. Key predictors such as **MonthlyCharges**, **TotalCharges**, and **Tenure** consistently ranked highest, indicating that customer spending patterns and contract duration play significant roles in churn behaviour.

The integration of SMOTE during training improved the detection of minority churn cases, leading to more robust performance on unseen data. The side-by-side evaluation framework allowed transparent comparison between tools, supporting data-driven recommendations for operational deployment.

In a business context, these insights can guide targeted retention campaigns, optimise pricing strategies, and improve service quality. Future improvements may involve exploring ensemble methods that combine the strengths of multiple models, as well as integrating additional behavioural and demographic data to further enhance prediction accuracy.

6. References

Bazarbash, M (2020) FinTech in Financial Inclusion: Machine Learning Applications in Assessing Credit Risk. IMF Working Paper No. 2020/109. Available at: https://www.imf.org/en/Publications/WP/Issues/2020/06/05/FinTech-in-Financial-Inclusion-49486 (Accessed: 12 July 2025).

Chumbar, S (2023) The CRISP-DM Process: A Comprehensive Guide. Medium. Available at: https://medium.com/@shawn.chumbar/the-crisp-dm-process-a-comprehensive-guide-4d893aecb151 (Accessed: 10 July 2025).

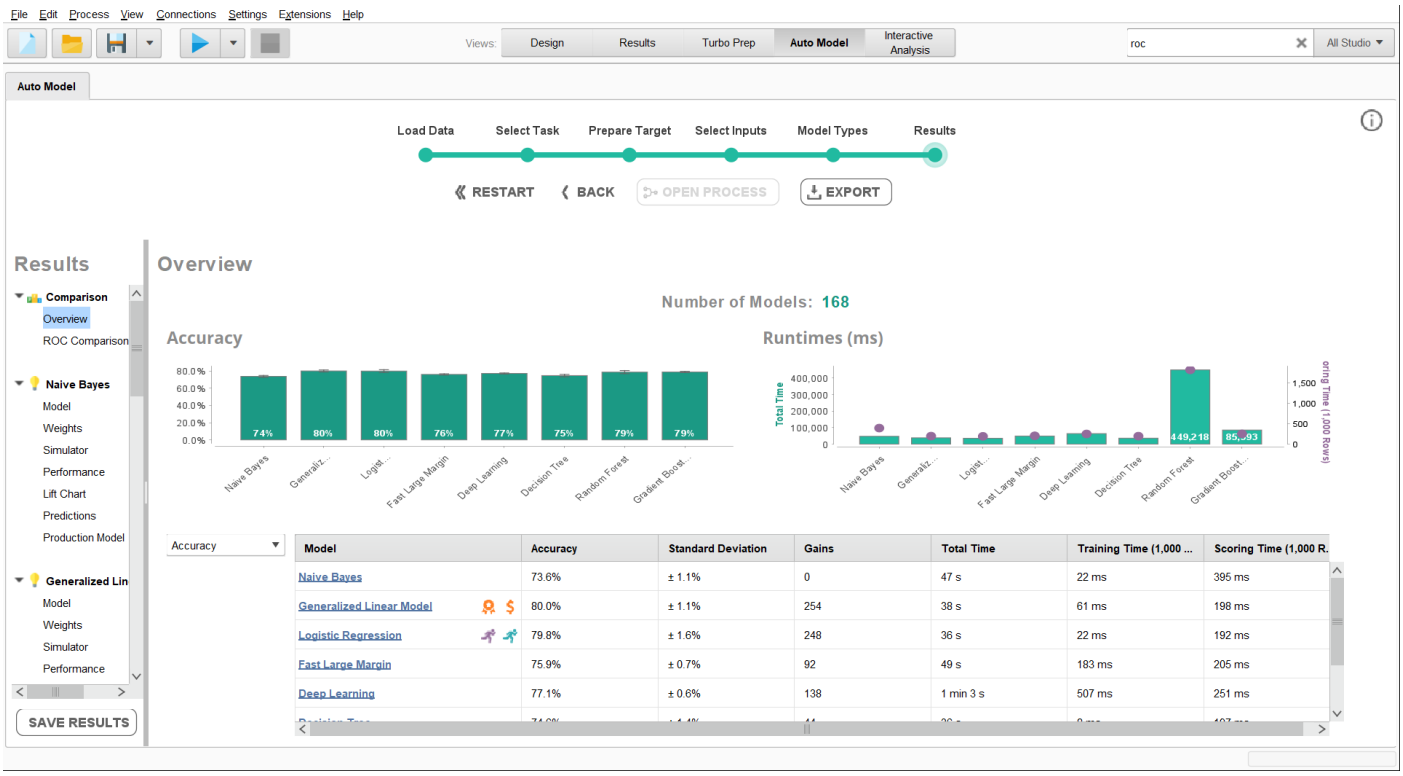
Kruse, F and Schröer, C (2021) A Systematic Literature Review on Applying CRISP-DM Process Model. Berlin: Berliner Ring 2, University of Oldenburg.

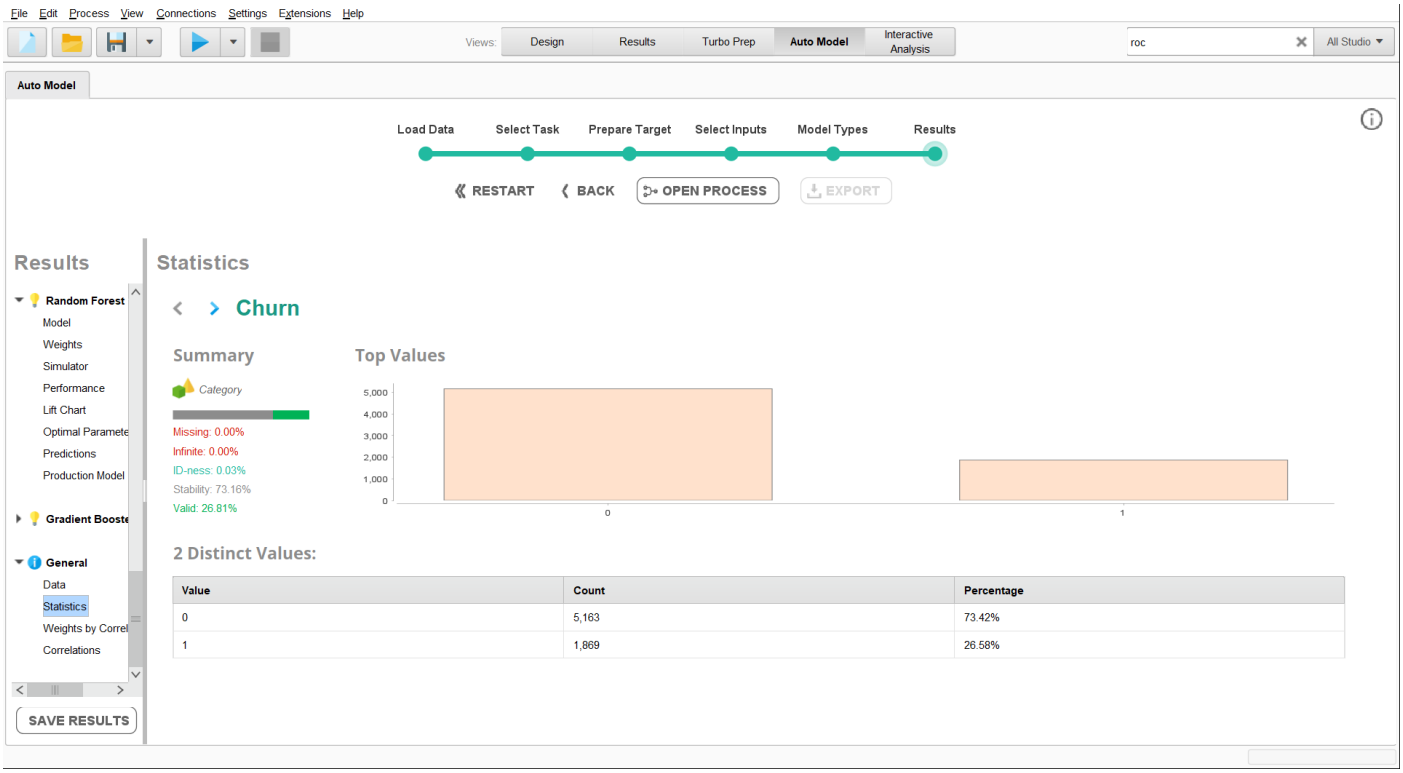
Karvana, KGM, Yazid, S, Syalim, A and Mursanto, P (2019) 'Customer Churn Analysis and Prediction Using Data Mining Models in Banking Industry', 2019 International Workshop on Big Data and Information Security (IWBIS), Bali, Indonesia, pp. 33–38. doi: 10.1109/IWBIS.2019.8935884.

Wirth, R and Hipp, J (2000) CRISP-DM: Towards a Standard Process Model for Data Mining. DaimlerChrysler Research and Wilhelm-Schickard-Institute, Tübingen, Germany.

Zhang, C, Wang, H and Sun, Y (2022) 'Explainable Machine Learning for Customer Churn Prediction in Telecoms', Journal of Big Data Analytics in Telecommunications, 10(2), pp. 112–127.

Appendix A – RapidMiner





Random Forest - Performance

Criterion	Value	Standard Deviation
AUC	84.4%	± 1.1%
Accuracy	78.6%	± 1.8%
Classification Error	21.4%	± 1.8%
F Measure	51.2%	± 6.4%
Precision	66.2%	± 4.7%
Recall	41.9%	± 6.6%
Sensitivity	41.9%	± 6.6%
Specificity	92.1%	± 1.2%

Confusion Matrix

	true 0	true 1	class precision
pred. 0	1351	315	81.09%
pred. 1	115	229	66.57%
class recall	92.16%	42.10%	

Deep Learning - Performance

Criterion	Value	Standard Deviation
Accuracy	77.1%	± 0.6%
Classification Error	22.9%	± 0.6%
AUC	83.8%	± 1.9%
Precision	68.1%	± 4.1%
Recall	24.1%	± 3.6%
F Measure	35.5%	± 4.5%
Sensitivity	24.1%	± 3.6%
Specificity	96.0%	± 0.2%

precision

Confusion Matrix

	true 0	true 1	class precision
pred. 0	1421	402	77.95%
pred. 1	59	128	68.45%
class recall	96.01%	24.15%	

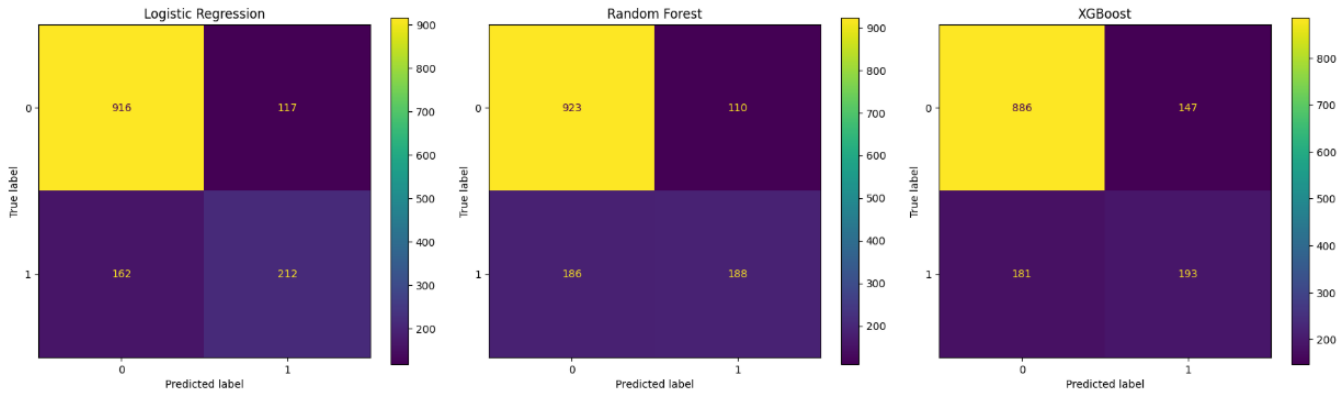
Logistic Regression - Performance

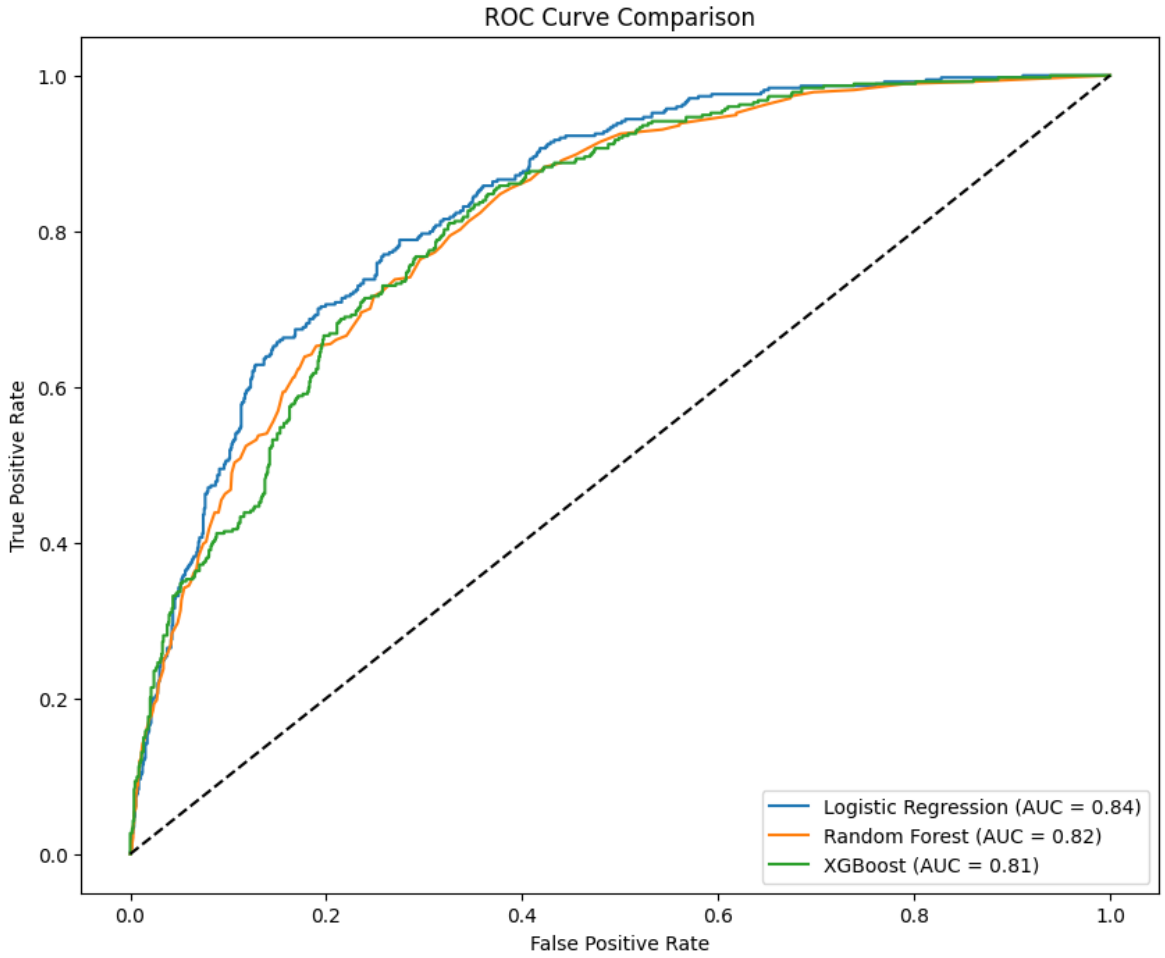
Criterion	Value	Standard Deviation
Accuracy	79.8%	± 1.6%
Classification Error	20.2%	± 1.6%
AUC	83.9%	± 2.6%
Precision	66.3%	± 5.3%
Recall	47.2%	± 4.9%
F Measure	55.1%	± 4.9%
Sensitivity	47.2%	± 4.9%
Specificity	91.4%	± 1.4%

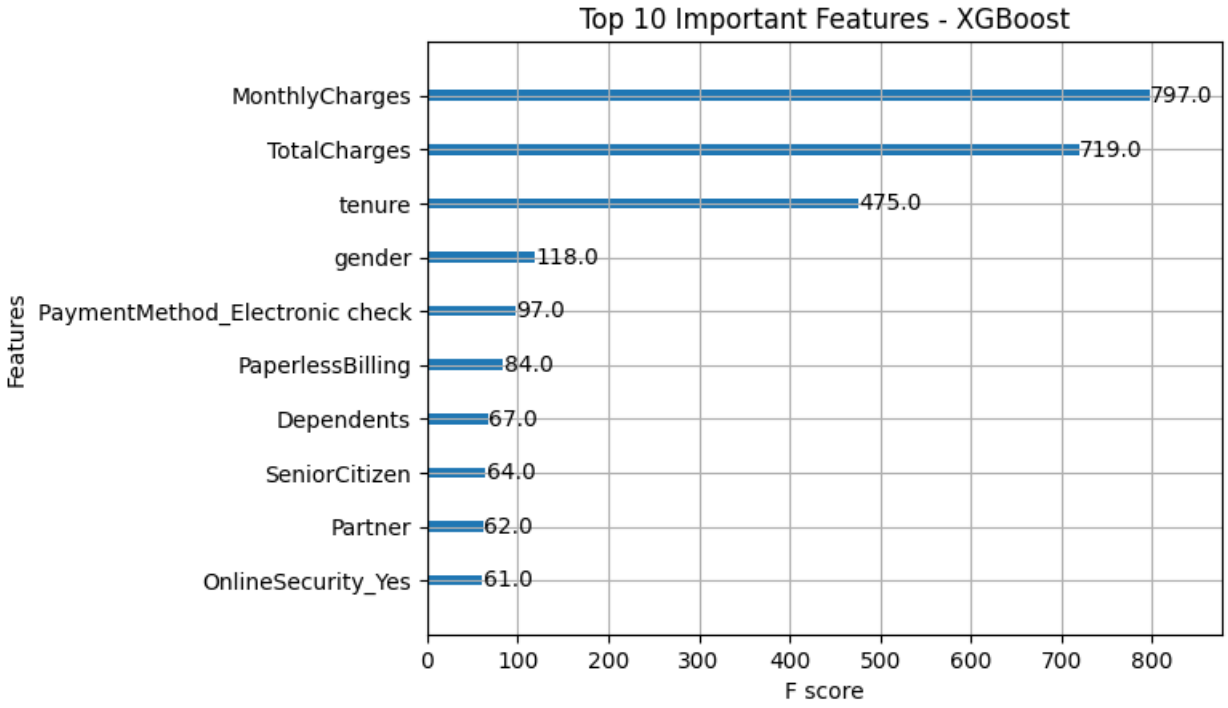
Confusion Matrix

	true 0	true 1	class precision
pred. 0	1353	279	82.90%
pred. 1	127	251	66.40%
class recall	91.42%	47.36%	

Appendix B – Python






Top 10 Important Features - XGBoost
MonthlyCharges
TotalCharges
tenure
gender
PaymentMethod_Electronic check
PaperlessBilling
Dependents
SeniorCitizen
Partner
OnlineSecurity_Yes
797.0
719.0
475.0
118.0
97.0
84.0
67.0
64.0
62.0
61.0
Features
0
100
200
300
400
500
600
700
800
F score