

INTERNSHIP PROJECT REPORT

“Customer Churn Prediction for Telecom Industry Using Machine Learning”

*A Report submitted in partial fulfillment of the requirements for the Internship at **InLighnx Global** (InLighn Tech)*

Submitted By:

Parvati Sheeba Unnithan
parvati742002@gmail.com

Internship Organization:

InLighnx Global, Corporate Office-
Office No: VO-301, WeWork
Prestige Central, Ground Floor, 36,
Infantry Rd, Tasker Town, Shivaji
Nagar, Bengaluru, Karnataka 560001

Internship Duration: 19th July 2025 – 20th August 2025

Date of Submission: 20-08-2020

CONTENTS

CHAPTER 1:	INTRODUCTION	3
	1.1 Overview	
CHAPTER 2:	METHODOLOGY	4-9
	2.1 Dataset Procurement	
	2.2 Library Imports and Dependencies	
	2.3 Exploratory Data Analysis	
	2.3.1 Handling Data Types and Missing Values	5
	2.3.2 Data Balancing, Feature Scaling, and Dataset Partitioning	7
	2.4 Training and Prediction	7
	2.4.1 Decision Tree Classifier	8
	2.4.2 Random Forest Classifier	9
	2.4.3 Gradient Boosting Classifier	
CHAPTER 3:	RESULT AND ANALYSIS	10-13
	3.1 Inference and Interpretation	
	3.1.1 Decision Tree Classifier Analysis	
	3.1.2 Random Forest Analysis	
	3.1.3 Gradient Boosting Classifier Analysis	11
	3.2 Optimization of Gradient Boosting Classifier	12
CHAPTER 4:	CONCLUSION	14
REFERENCES		15

LIST OF FIGURES

Figure No.	TITLE	Page No.
Figure 2.3.1:	Visualization of Object Datatypes and Missing Value Patterns in the Dataset	5
Figure 2.3.2:	Visualization of Unbalanced Dataset	
Figure 2.3.1.1:	Code for Dropping Columns, Converting Object Types to Numbers, and Cleaning Spaces	6
Figure 2.3.1.2:	Correlation between Target Class and Other Attributes	7
Figure 2.4.1.1:	Code for Decision Tree Training	8
Figure 2.4.2.1:	Code for Random Forest Classifier Training	
Figure 2.4.3.1:	Code for Gradient Boosting Classifier Training	9
Figure 3.1.1.1:	Evaluation Metrics of Decision Tree Classifier	10
Figure 3.1.2.1:	Evaluation Metrics of Random Forest	11
Figure 3.1.3.1:	Evaluation Metrics of Gradient Boosting Classifier	12
Figure 3.2.1:	Hyperparameter Tuning of Gradient Boosting Classifier using GridSearchCV	1
Figure 3.2.2:	Top 10 Most Important Features Influencing Customer Churn	13

CHAPTER 1

INTRODUCTION

1.1 Overview

Customer churn is when customers leave a service provider, is a major challenge for firms in the competitive telecommunications industry. It is usually cheaper to keep current customers than to find new ones. Therefore, being able to predict churn is important for business sustainability and growth. This project aims to create a model that identifies telecom customers who might stop using their services. This allows companies to act early and improve customer retention.

For this project, we used the Telco Customer Churn dataset from Kaggle, which IBM originally provided. This dataset includes a wide range of customer information, such as demographics, account details, service usage, and billing information. The target variable, Churn, shows whether a customer has left the company. With both categorical and numerical features present, we can use various preprocessing techniques and machine learning models effectively.

The project workflow involves cleaning the data, encoding categorical variables, and exploring the data to find key indicators of churn, including contract type, payment method, and monthly charges. After cleaning the data, we trained and evaluated models like Decision Tree, Random Forest, and Gradient Boosting Classifier. Among these models, Gradient Boosting achieved the highest accuracy and F1-scores, proving to be the best at identifying churned customers.

To improve model performance, fine-tuning is done in which the hyperparameters using GridSearchCV with 5-fold cross-validation are made used. This enhanced the Gradient Boosting model's ability to generalize. This project also analyzed feature importance to identify the most influential predictors, providing useful insights for targeted retention strategies.

By merging data-driven modeling with interpretability, this project not only predicts churn accurately but also uncovers the reasons behind customer loss. The results can help telecom companies make better decisions to reduce churn and increase customer satisfaction.

CHAPTER 2

METHODOLOGY

2.1 Dataset Procurement

Data used in this project is sourced from IBM Sample Data Sets, is designed to help predict customer churn in the telecom industry. Each row represents a unique customer, with attributes covering service subscriptions, account details, and demographics. Key features include contract type, monthly and total charges, service usage for instance internet, tech support and whether the customer has churned. The dataset enables the development of machine learning models aimed at identifying customers likely to leave, supporting focused retention strategies.

2.2 Library Imports and Dependencies

The dataset sourced is stored in the form of a zip file and is extracted using the ZipFile class from Python's built-in zipfile module. The necessary Python libraries were imported for data preprocessing, visualization, model training, evaluation, and handling class imbalance to build an effective customer churn prediction model.

2.3 Exploratory Data Analysis

In the Telco Customer Churn dataset, the initial EDA shows that several columns have a data type of object as shown in the **Figure 2.3.1** even though some of them like TotalCharges should ideally be numeric for analysis and modelling. This happens because the column might contain non-numeric values such as spaces or text, causing pandas to treat it as a string instead of a number. Non-numeric data in columns that represent numerical information can lead to errors during model training and must be converted to appropriate data types after proper cleaning or encoding.

The missing values check shows no missing data in the dataset as shown in **Figure 2.3.1**, which is a good sign and means we can proceed without imputation for now. However, the Churn column is highly imbalanced with 5,174 'No' entries compared to 1,869 'Yes' entries, meaning most customers have not churned **Figure 2.3.2**. This class imbalance can cause bias in model training toward predicting "No Churn" so balancing techniques such as oversampling (SMOTE).

```

datatypes of attributes :
customerID      object
gender          object
SeniorCitizen   int64
Partner         object
Dependents      object
tenure          int64
PhoneService    object
MultipleLines   object
InternetService object
OnlineSecurity  object
OnlineBackup    object
DeviceProtection object
TechSupport     object
StreamingTV     object
StreamingMovies object
Contract        object
PaperlessBilling object
PaymentMethod   object
MonthlyCharges  float64
TotalCharges    object
Churn           object
dtype: object
find missing values information :
customerID      0
gender          0
SeniorCitizen   0
Partner         0
Dependents      0
tenure          0
PhoneService    0
MultipleLines   0
InternetService 0
OnlineSecurity  0
OnlineBackup    0
DeviceProtection 0
TechSupport     0
StreamingTV     0
StreamingMovies 0
Contract        0
PaperlessBilling 0
PaymentMethod   0
MonthlyCharges  0
TotalCharges    0
Churn           0
dtype: int64

```

Figure 2.3.1: Visualization of Object Datatypes and Missing Value Patterns in the Dataset

```

find the data is balanced or not :
Churn
No      5174
Yes     1869
Name: count, dtype: int64

```

Figure 2.3.2: Visualization of unbalanced dataset

2.3.1 Handling Data Types and Missing Values

During data exploration **Figure 2.3.1.1**, the TotalCharges column was identified as having an object data type, despite containing primarily numeric-looking values. Conversion attempts using `astype(float)` resulted in errors due to the presence of empty strings (' ') that are not valid numerical

entries. To address this, these empty strings were first replaced with NaN using the `replace()` method from Pandas. Subsequently, rows containing these NaN values were removed using `dropna()` to maintain dataset integrity. Once cleaned, the `TotalCharges` column was successfully cast to float, enabling its use in numerical analysis and model training.

This process ensures that the `TotalCharges` attribute is correctly formatted for subsequent analyses, such as correlation computations and predictive modeling, thereby improving the integrity of the churn prediction project. The column `customerID` column was dropped from the dataset, as it serves only as a unique identifier and holds no predictive value for churn modeling **Figure 2.3.1.1**.

```
[19] df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSu
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows x 21 columns

```
[21] df.drop('customerID', axis=1, inplace=True)
```

```
[22] df['TotalCharges'].replace(' ', pd.NA, inplace=True)
df.dropna(subset=['TotalCharges'], inplace=True)
df['TotalCharges'] = df['TotalCharges'].astype(float)
```

/tmp/ipython-input-729438867.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy. For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead.

```
df['TotalCharges'].replace(' ', pd.NA, inplace=True)
```

Figure 2.3.1.1: Code for dropping columns, converting object types to numbers and cleaning spaces

A list named `check_lst` was created to store the names of categorical columns (with object data type) that were identified for conversion to numerical format. Simultaneously, `before_lst` stored the full list of columns in the dataset. By comparing these two lists, a new list called `balance_lst` was generated to capture any remaining object-type columns from `before_lst` that were not included in `check_lst`. This approach ensures that no categorical columns are unintentionally excluded before proceeding with encoding, thus maintaining data consistency for modeling.

Binary categorical columns were label-encoded (0/1) for simplicity, while multi-class categorical columns were one-hot encoded to avoid introducing false ordinal relationships. Boolean results from one-hot encoding were converted to integers for compatibility. Correlation analysis was then performed, and low-correlation features were removed to reduce noise and improve model performance **Figure 2.3.1.2**.

	TotalCharges	Churn	...	\
gender	0.000048	-0.008545	...	
SeniorCitizen	0.102411	0.150541	...	
Partner	0.319072	-0.149982	...	
Dependents	0.064653	-0.163128	...	
tenure	0.825880	-0.354049	...	
PhoneService	0.113008	0.011691	...	
PaperlessBilling	0.157830	0.191454	...	
MonthlyCharges	0.651065	0.192858	...	
TotalCharges	1.000000	-0.199484	...	
Churn	-0.199484	1.000000	...	
MultipleLines_No phone service	-0.113008	-0.011691	...	
MultipleLines_Yes	0.469042	0.040033	...	
InternetService_Fiber optic	0.360769	0.307463	...	
InternetService_No	-0.374878	-0.227578	...	
OnlineSecurity_No internet service	-0.374878	-0.227578	...	
OnlineSecurity_Yes	0.412619	-0.171270	...	
OnlineBackup_No internet service	-0.374878	-0.227578	...	
OnlineBackup_Yes	0.510100	-0.082307	...	
DeviceProtection_No internet service	-0.374878	-0.227578	...	
DeviceProtection_Yes	0.522881	-0.066193	...	
TechSupport_No internet service	-0.374878	-0.227578	...	
TechSupport_Yes	0.432868	-0.164716	...	
StreamingTV_No internet service	-0.374878	-0.227578	...	
StreamingTV_Yes	0.515709	0.063254	...	
StreamingMovies_No internet service	-0.374878	-0.227578	...	
StreamingMovies_Yes	0.519867	0.060860	...	
Contract_One year	0.170569	-0.178225	...	
Contract_Two year	0.358036	-0.301552	...	
PaymentMethod_Credit card (automatic)	0.182663	-0.134687	...	
PaymentMethod_Electronic check	-0.060436	0.301455	...	
PaymentMethod_Mailed check	-0.294708	-0.090773	...	

Figure 2.3.1.2: Correlation between target class and other attributes

2.3.2 Data Balancing, Feature Scaling, and Dataset Partitioning

To ensure the robustness and fairness of the predictive modeling process, key preprocessing steps were applied following categorical encoding and feature selection. First, class imbalance in the target variable Churn was addressed using Synthetic Minority Oversampling Technique known as SMOTE. This method generates synthetic examples for the minority class, resulting in a balanced distribution between classes. Post-oversampling value counts confirmed successful balancing. Next, the feature matrix (X_os) was standardized using StandardScaler to normalize the feature values, ensuring that all variables contribute equally to the model's learning process. The fully preprocessed dataset was split into training and testing subsets in a 70:30 ratio using a fixed random_state to guarantee reproducibility. These steps completed the data preparation phase and established a clean foundation for model development and evaluation.

2.4 Training and Prediction

2.4.1 Decision Tree Classifier

In **Figure 2.4.1** the code segment involves the process of training a Decision Tree classifier on the training dataset and subsequently using the trained model to make predictions on the unseen test dataset.

First, the Decision Tree Classifier is instantiated with a fixed random state to ensure consistent results upon repeated runs. The model is then fitted on the training feature set and corresponding target labels, allowing it to learn from the data. After the training phase, the model generates predictions for the test features. The predicted values are prepared for evaluation through various metrics, including accuracy, classification reports, and confusion matrices **Figure 2.4.1.1**. However, this section solely focuses on the setup and execution of the training and prediction processes, without analysing or interpreting the resulting performance metrics.



```
DECISION TREE TRAINING

[41] dtc=DecisionTreeClassifier(random_state=42)
dtc.fit(X_train, Y_train)

DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)

DECISION TREE Testing and Result

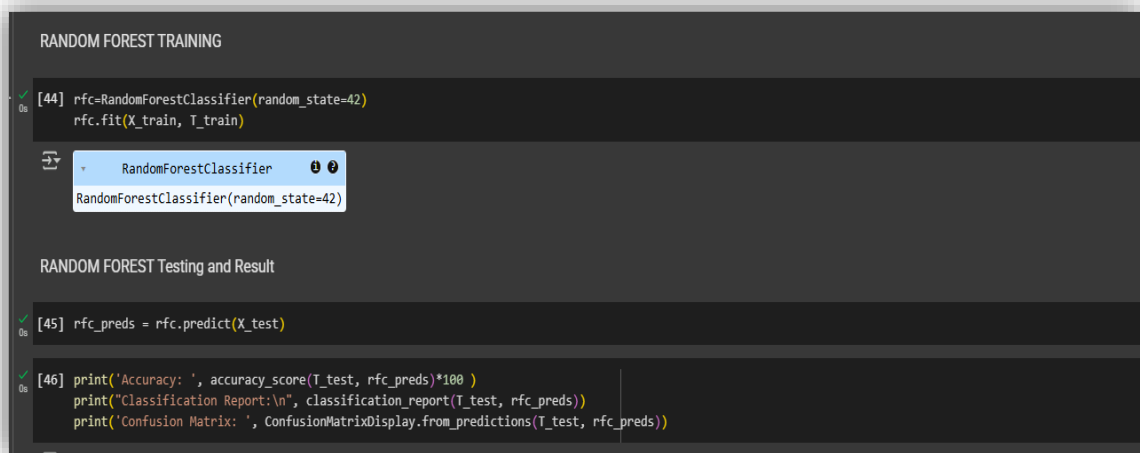
[42] dtc_preds = dtc.predict(X_test)

print('Accuracy: ', accuracy_score(Y_test, dtc_preds)*100 )
print("Classification Report:\n", classification_report(Y_test, dtc_preds))
print('Confusion Matrix: ', ConfusionMatrixDisplay.from_predictions(Y_test, dtc_preds))
```

Figure 2.4.1.1: Code for Decision Tree training

2.4.2 Random Forest Classifier

After training the model using a Decision Tree, a Random Forest classifier was also employed to enhance performance. Random Forest is an ensemble method that builds multiple decision trees and combines their predictions, which reduces the risk of overfitting, improves accuracy, and provides better generalization to unseen data. Moreover, it can handle noisy data more effectively and offers feature importance scores to help identify the most influential predictors **Figure 2.4.2.1**.



```
RANDOM FOREST TRAINING

[44] rfc=RandomForestClassifier(random_state=42)
rfc.fit(X_train, Y_train)

RandomForestClassifier
RandomForestClassifier(random_state=42)

RANDOM FOREST Testing and Result

[45] rfc_preds = rfc.predict(X_test)

[46] print('Accuracy: ', accuracy_score(Y_test, rfc_preds)*100 )
print("Classification Report:\n", classification_report(Y_test, rfc_preds))
print('Confusion Matrix: ', ConfusionMatrixDisplay.from_predictions(Y_test, rfc_preds))
```

Figure 3.4.2.1: Code for Random Forest Classifier training

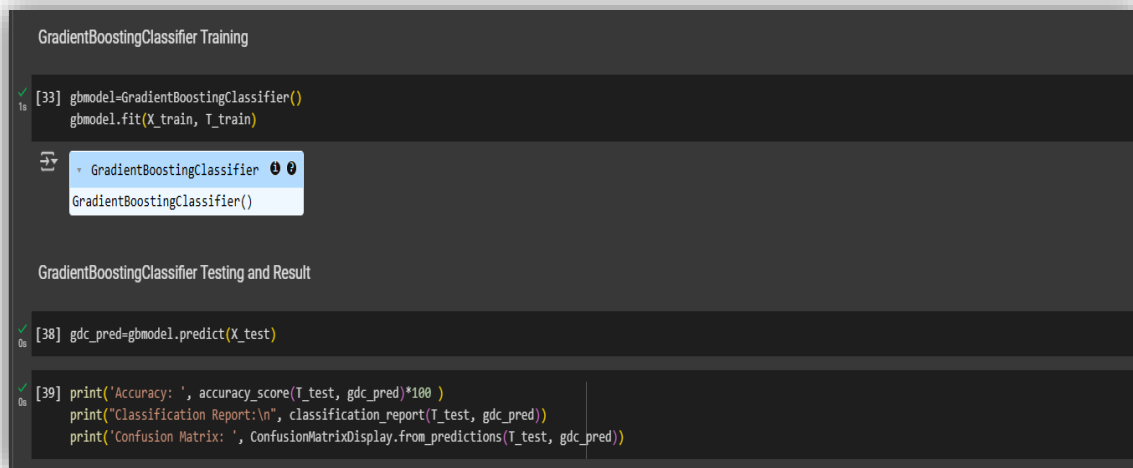
2.4.3 Gradient Boosting Classifier

In the final stage of model development, we introduced the Gradient Boosting Classifier to improve prediction performance. Gradient Boosting is an ensemble learning technique that builds models one after another. Each new model tries to fix the mistakes made by the previous ones. This process allows it to capture complex, non-linear patterns in the data better than standalone models can.

In this step, it involves creating the classifier using scikit-learn's GradientBoostingClassifier with a fixed random state to ensure reproducibility. The model is trained on the standardized training dataset, learning how customer features relate to churn behavior. Once training was finished, the model is used to make predictions on the test set. Then evaluation of these predictions with accuracy, precision, recall, F1-score, and confusion matrix metrics are performed. This evaluation gave us insights into the model's ability to correctly identify both churn and non-churn customers as shown in **Figure 3.4.3.1**.

To enhance its predictive abilities, the method of fine-tuning the Gradient Boosting model using GridSearchCV and 5-fold cross-validation is performed. This optimization step tested different combinations of parameters like learning rate, maximum depth of trees, and the number of estimators. The best parameter set we found was `learning_rate = 0.05`, `max_depth = 5`, and `n_estimators = 100`. These values are used to retrain the model, which led to better generalization and performance.

Ultimately, the Gradient Boosting Classifier proved to be the best-performing model in this project. It combined high accuracy with low error rates and was selected as the final model for deployment and analysis.



```
GradientBoostingClassifier Training

[33] gbmodel=GradientBoostingClassifier()
      gbmodel.fit(X_train, Y_train)

GradientBoostingClassifier
GradientBoostingClassifier()

GradientBoostingClassifier Testing and Result

[38] gdc_pred=gbmodel.predict(X_test)

[39] print('Accuracy: ', accuracy_score(Y_test, gdc_pred)*100 )
      print('Classification Report:\n', classification_report(Y_test, gdc_pred))
      print('Confusion Matrix: ', ConfusionMatrixDisplay.from_predictions(Y_test, gdc_pred))
```

Figure 2.4.3.1: Code for Gradient Boosting Classifier training

CHAPTER 3

RESULT AND ANALYSIS

3.1 Inference and Interpretation

After training and testing the models, as the next step evaluating each classifier using metrics like accuracy, precision, recall, F1-score, and confusion matrix is done. The Random Forest and Gradient Boosting models performed better than the Decision Tree when it came to accuracy and generalization.

3.1.1 Experimental analysis using Decision Tree classifier

The Decision Tree classifier served as a strong baseline model for predicting customer churn, achieving an accuracy of approximately 88.6% with balanced performance across both classes f1-score = 0.89 for non-churn and 0.88 for churn, as evidenced by the confusion matrix which reported 1471 true negatives, 1415 true positives, 162 false positives, and 209 false negatives; however, despite effectively capturing underlying data patterns, the model remains susceptible to overfitting particularly in the absence of pruning or depth constraints which may result in reduced generalization on unseen data.

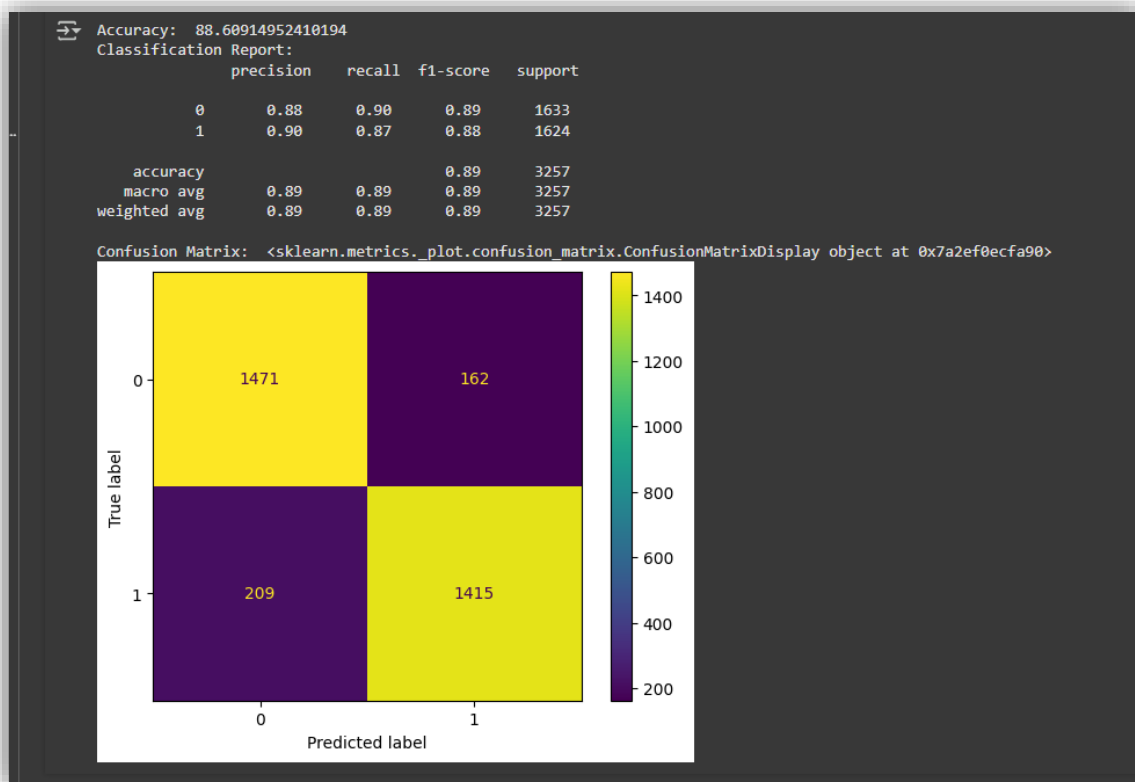


Figure 3.1.1.1: Evaluation metrics of Decision Tree classifier

3.1.2 Experimental analysis using Random Forest algorithm

In contrast, Random Forest, which is an ensemble method, showed greater stability and accuracy. It achieved around 91.18% accuracy by combining predictions from multiple trees. The model had balanced precision, recall, and F1-scores of 0.91 for both churn and non-churn classes. The confusion matrix revealed 1,488 true negatives and 1,482 true positives, with only 145 false positives and 142

false negatives. This indicates the model managed both classes nearly equally well. This balance means the model is well-calibrated and not biased toward the majority class, which is essential in churn prediction. When compared to the Decision Tree with an accuracy of 88.6%, the Random Forest clearly surpassed it in overall accuracy and consistency across class predictions. The ensemble approach helped reduce variance and improved generalization. It also showed better resistance to noise and overfitting, which are common issues with Decision Tree models. Thus, Random Forest provided significant performance gains and greater reliability, making it a more suitable choice for predicting customer churn in this case.

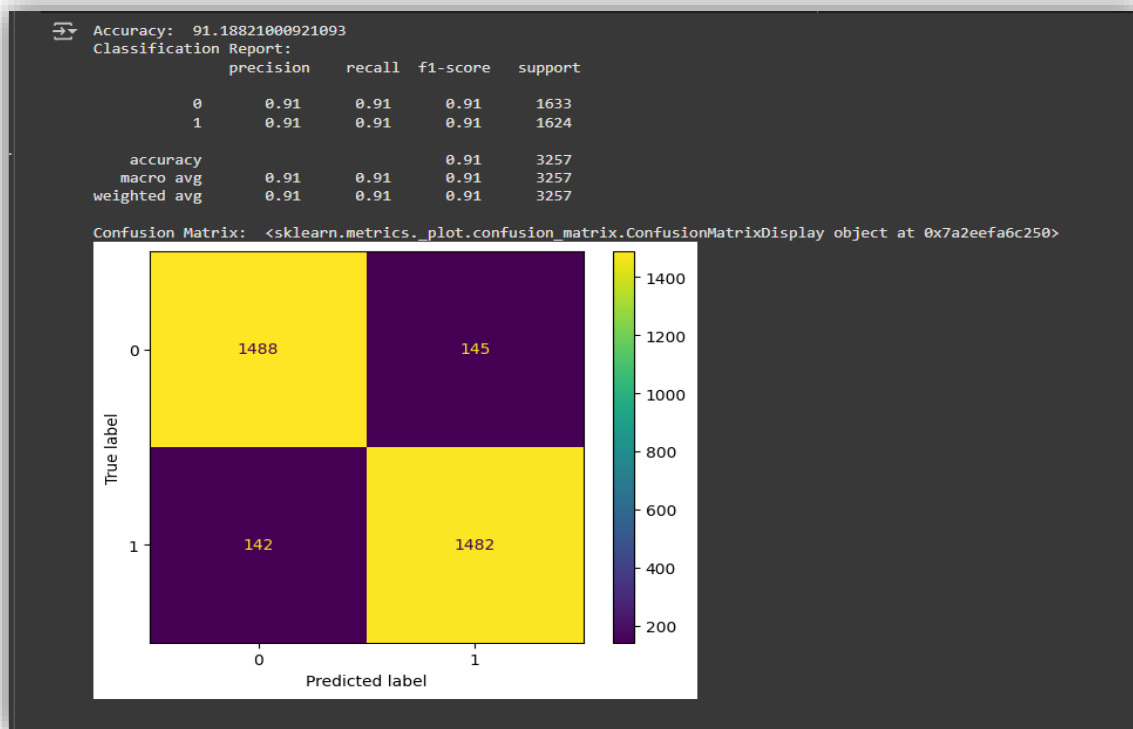


Figure 3.1.2.1: Evaluation metrics of Random Forest

3.1.3 Experimental analysis using Gradient Boosting classifier

The Gradient Boosting classifier achieved the highest accuracy among the tested models, reaching about 91.71%. It also had macro-averaged precision, recall, and F1-scores of 0.92. The confusion matrix reflects this strong performance, with 1,482 true negatives and 1,505 true positives, 151 false positives, and just 119 false negatives. This is the fewest missed churn cases of all the models evaluated. Gradient Boosting's iterative approach, which aims to fix the errors of previous models by training new ones on their residuals, helped it capture complex patterns in the data effectively. When comparing it to the Decision Tree, which had an accuracy of 88.6%, and the Random Forest, which reached 91.18%, the Gradient Boosting classifier outperformed both across all measures and provided the most consistent results. While Random Forest performed better concerning overfitting than the Decision Tree, Gradient Boosting surpassed both by achieving greater precision in identifying churned customers. This makes it particularly valuable for minimizing customer loss in real-world scenarios. In fact, Gradient Boosting emerged as the top-performing model for predictive accuracy and error reduction.



Figure 3.1.3.1: Evaluation metrics of Gradient Boosting classifier

3.2 Optimization of the Gradient Boosting Classifier

During the experimentation with three algorithms for classification, it was seen that Gradient Boosting slightly outperformed Random Forest in terms of precision and generalization so to further enhance the performance of the Gradient Boosting classifier, hyperparameter tuning was conducted using GridSearchCV along with 5-fold cross-validation. This systematic approach allowed the model to be optimized by testing multiple combinations of key parameters such as `learning_rate`, `max_depth`, and `n_estimators` as shown in **Figure 3.2.1**. The best parameters identified were `learning_rate=0.05`, `max_depth=5`, and `n_estimators=100`, which collectively improved the model's generalization capabilities and reduced the risk of overfitting. Once the model was retrained with these optimized settings, it demonstrated slightly improved accuracy and robustness compared to the initial implementation.



Figure 3.2.1: Hyperparameter tuning of the Gradient Boosting Classifier using GridSearchCV with 5-fold cross-validation, followed by evaluation metrics of the optimized model

To complement this evaluation, the top Ten important features were extracted and shown in **Figure 3.2.2**. The plot reveals that the most influential predictors of customer churn relate to payment method, with PaymentMethod_Mailed check and PaymentMethod_Electronic check having the highest importance scores. Other key features include TotalCharges, InternetService_No, and PaperlessBilling. Demographic and service-related variables such as tenure, Partner, gender, and Dependents also play a role. These insights confirm that account activity and billing preferences are crucial in predicting churn behavior. This makes them significant factors for customer retention strategies.

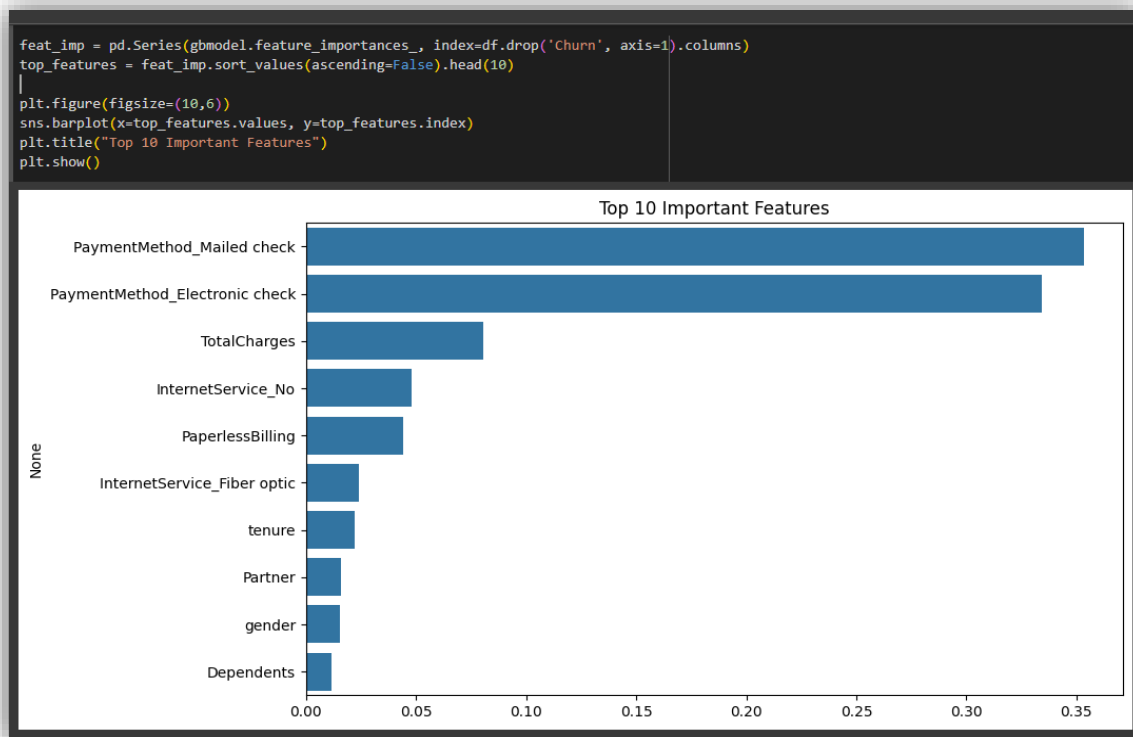


Figure 3.2.2: Top 10 most important features influencing customer churn as identified by the optimized Gradient Boosting model

CHAPTER 4

CONCLUSION

The development of a robust customer churn prediction model is a crucial step toward improving customer retention in the highly competitive telecommunications industry. In this project, systematically processed and analysed the Telco Customer Churn dataset, applying essential preprocessing steps such as handling data type inconsistencies, removing irrelevant features, balancing classes using SMOTE, and scaling features for uniformity. Multiple machine learning algorithms, including Decision Tree, Random Forest, and Gradient Boosting Classifier, were trained and evaluated using metrics like accuracy, precision, recall, F1-score, and confusion matrix. Among these, the Gradient Boosting Classifier emerged as the best-performing model, achieving an impressive accuracy of 91.71% with balanced performance across churn and non-churn classes. Hyperparameter tuning further enhanced its generalization capabilities, reducing errors and improving predictive power. Feature importance analysis revealed key drivers of churn, particularly payment method, billing type, total charges, and tenure, offering actionable insights for targeted retention strategies. By integrating predictive modeling with interpretability, this project not only provides a reliable tool for churn prediction but also delivers meaningful business intelligence to guide decision-making. The results demonstrate the potential of machine learning in enabling proactive customer engagement, ultimately leading to reduced churn rates, increased customer satisfaction, and sustainable business growth.

REFERENCES

- [1] <https://www.kaggle.com/datasets/blastchar/telco-customer-churn>