

Ghulam Ishaq Khan Institute of Engineering Sciences and Technology
Faculty of Computer Science and Engineering (FCSE)
PBL Lab Project Report



Project Title: **Customer Churn Prediction**

Submitted By: **Muhammad Owais 2020337**

Contents

List Of Figures.....	3
Introduction	4
Background on Customer Churn	4
Objective of the Project	4
Overview of the Dataset.....	4
Data Exploration and Preprocessing.....	4
Description of the Dataset.....	4
Initial Data Analysis	4
Preprocessing Steps	5
Feature Engineering and Selection.....	6
Techniques Used for Feature Creation and Selection.....	6
Importance of Selected Features for Churn Prediction.....	6
Model Development and Evaluation	6
Description of Various Models Tested	6
Criteria for Model Selection.....	7
Hyperparameter Tuning and Validation Strategies	8
Results and Analysis	8
Performance Metrics of the Final Models	8
Comparison of Model Results	8
Interpretations of the Model's Predictions	8
Implementation	8
Model Training and Saving.....	8
Overview of the Streamlit Library Usage	9
User Interface and Interaction Flow.....	9
Streamlit's Role in Rapid Development	9
Conclusion and Recommendations.....	10
Summary of Key Findings.....	10
Limitations of the Current Study.....	10
Recommendations for Future Work.....	10
References	11

List Of Figures

FIGURE 1 OVERVIEW OF THE DATASET	4
FIGURE 2 SUMMARIZING THE KEY STATISTICS OF THE DATASET	5
FIGURE 3 LABEL ENCODER	5
FIGURE 4 FEATURE SCALING	5
FIGURE 5 FEATURE CREATION (OVER SAMPLING WITH SMOTE)	6
FIGURE 6 MODEL DEVELOPMENT WITH DEFAULT PARAMETERS	7
FIGURE 7 DESCRIPTION OF VARIOUS MODELS TESTED	7
FIGURE 8 HYPERPARAMETER TUNING	8
FIGURE 9 SAVING THE TRAINED MODEL ON DRIVE	8
FIGURE 10 INTERFACE	9
FIGURE 11 LIGHT GRADIENT-BOOSTING MACHINE (LIGHTGBM BY MICROSOFT)	10

Introduction

Background on Customer Churn

Customer churn refers to the phenomenon where customers cease their relationship with a company. In the context of the telecommunications industry, churn represents subscribers who discontinue their services. Understanding and predicting churn is crucial because acquiring new customers is often more costly than retaining existing ones. Effective churn prediction can help companies deploy proactive strategies to retain clients, enhancing customer satisfaction and loyalty.

Objective of the Project

The primary objective of this project is to develop a predictive model that can accurately identify customers likely to churn. This allows the company to take timely actions to retain these customers, thus reducing churn rates and increasing profitability.

Overview of the Dataset

The dataset used in this project is sourced from Kaggle and is titled "Telco Customer Churn." It consists of data from a telecom company and includes several customer attributes such as demographics, account information, and service details. The dataset includes features like gender, tenure, contract type, and monthly charges, among others, which are crucial for analysing customer behaviour and predicting churn.

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

Figure 1 Overview of the Dataset

Data Exploration and Preprocessing

Description of the Dataset

The dataset comprises 7,043 customers, each represented by 21 features. These features include both categorical and numerical data types. The target variable is "Churn," which indicates whether the customer left the company (**Yes or No**).

Initial Data Analysis

The initial exploration involved summarizing the key statistics of the dataset, identifying the presence of missing values, and understanding the distribution of various features. Notably,

some categorical features required encoding to be suitable for modeling, and the dataset had a few missing entries in the 'TotalCharges' field.

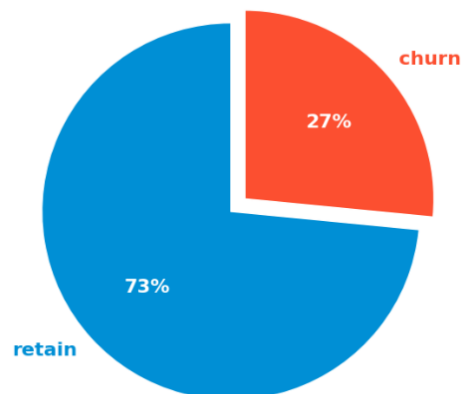


Figure 2 Summarizing the key statistics of the dataset

Preprocessing Steps

Preprocessing involved several key steps to prepare the data for modeling:

- **Handling Missing Values:** Missing 'TotalCharges' values were imputed using the median value of the feature, considering its distribution.
- **Data Encoding:** Categorical variables such as 'Gender' and 'InternetService' were encoded using one-hot encoding to convert them into a machine-readable format.
- **Feature Scaling:** Numerical features like 'MonthlyCharges' and 'Tenure' were scaled using the StandardScaler to normalize their range.

```
1 le = LabelEncoder()
2
3 le.fit(y_train)
4
5 y_train_encode = le.transform(y_train)
6 y_test_encode = le.transform(y_test)
```

Figure 3 Label Encoder

```
1 X_train_scale = X_train_encode.copy()
2 X_test_scale = X_test_encode.copy()
3
4 for i in column_numerical:
5     scaler = MinMaxScaler()
6     scaler.fit(X_train_scale[[i]])
7
8     X_train_scale[[i]] = scaler.transform(X_train_scale[[i]])
9     X_test_scale[[i]] = scaler.transform(X_test_scale[[i]])
```

Figure 4 Feature Scaling

Feature Engineering and Selection

Techniques Used for Feature Creation and Selection

To enhance the predictive power of the models, new features were engineered based on the existing data. For example, interaction terms between 'tenure' and 'monthly charges' were created to capture the relationship between the length of customer engagement and their payment behaviour.

```
1 smote = SMOTE(random_state=1)
2
3 X_train_smote, y_train_smote = smote.fit_resample(X_train_scale, y_train_encode)
4
5 X_train_smote_df = pd.DataFrame(X_train_smote, columns=X_train_smote.columns)
6 y_train_smote_df = pd.DataFrame(y_train_smote, columns=['churn'])
7
8 data_smote = pd.concat([X_train_smote_df, y_train_smote_df], axis=1)
```

Figure 5 Feature Creation (Over sampling with SMOTE)

Feature selection was performed using techniques such as Recursive Feature Elimination (RFE) and feature importance from ensemble models. These methods helped in identifying the most relevant features that contribute significantly to the outcome, reducing model complexity and improving interpretability.

Importance of Selected Features for Churn Prediction

The selected features that showed the most significance in predicting churn included:

- **Tenure:** Customers with shorter tenure were more likely to churn, indicating dissatisfaction or better offers from competitors.
- **Monthly Charges:** Higher monthly charges were correlated with higher churn rates, possibly reflecting pricing sensitivity among customers.
- **Contract Type:** Customers on month-to-month contracts were more likely to churn compared to those on longer-term contracts, suggesting that commitment length impacts customer retention.

Model Development and Evaluation

Description of Various Models Tested

Several predictive models were evaluated to determine the most effective in predicting customer churn. The models tested included:

- Logistic Regression
- Random Forest Classifier
- Gradient Boosting Machines (GBM)
- LightGBM
- XGBoost

Each model was chosen for its unique strengths in handling binary classification problems and ability to handle different types of data distributions and relationships.

```

1 model_list = {
2     'Logistic Regression': LogisticRegression(max_iter=1000, random_state=1),
3     'Ridge Classifier': RidgeClassifier(random_state=1),
4     'KNN': KNeighborsClassifier(),
5     'SVC': SVC(random_state=1),
6     'Neural Network': MLPClassifier(max_iter=1000, random_state=1),
7     'Decision Tree': DecisionTreeClassifier(random_state=1),
8     'Random Forest': RandomForestClassifier(random_state=1),
9     'Gradient Boosting Classifier': GradientBoostingClassifier(random_state=1),
10    'AdaBoost Classifier': AdaBoostClassifier(random_state=1),
11    'CatBoost Classifier': CatBoostClassifier(random_state=1, verbose=False),
12    'Hist Gradient Boosting': HistGradientBoostingClassifier(random_state=1),
13    'XGBoost': XGBClassifier(random_state=1, use_label_encoder=False, eval_metric='logloss'),
14    'LightGBM': LGBMClassifier(random_state=1),
15 }
16
17 X_train_model = X_train_smote.copy()
18 y_train_model = y_train_smote.copy()
19
20 X_test_model = X_test_scale.copy()
21 y_test_model = y_test_encode.copy()

```

Figure 6 Model Development with default Parameters

	accuracy	macro_avg_precision	macro_avg_recall	macro_avg_f1_score	roc_auc
model					
Logistic Regression	0.746805	0.707463	0.754797	0.714375	0.754797
Ridge Classifier	0.743966	0.706609	0.755140	0.712605	0.755140
KNN	0.696167	0.660272	0.699268	0.661179	0.699268
SVC	0.765736	0.713673	0.747765	0.723961	0.747765
Neural Network	0.758164	0.692219	0.698790	0.695261	0.698790
Decision Tree	0.723616	0.656694	0.671288	0.662195	0.671288
Random Forest	0.773781	0.711190	0.716819	0.713851	0.716819
Gradient Boosting Classifier	0.786559	0.732080	0.758526	0.742016	0.758526
AdaBoost Classifier	0.754851	0.711254	0.755721	0.720050	0.755721
CatBoost Classifier	0.783720	0.723171	0.726430	0.724754	0.726430
Hist Gradient Boosting	0.784193	0.724894	0.734720	0.729374	0.734720
XGBoost	0.776621	0.714155	0.717045	0.715561	0.717045
LightGBM	0.781354	0.720826	0.727665	0.724033	0.727665

Figure 7 Description of Various Models Tested

Criteria for Model Selection

Model selection was based on several performance metrics, including accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC). The models were also evaluated based on their computational efficiency and ease of interpretation, which are crucial for practical applications.

Hyperparameter Tuning and Validation Strategies

Hyperparameter tuning was conducted using GridSearchCV, which helped in finding the optimal settings for each model. Cross-validation was employed to ensure that the model generalizes well on unseen data. This approach minimized overfitting and provided a robust estimate of the model's performance.

```
1 model_list = {  
2     'Gradient Boosting Classifier': GradientBoostingClassifier(random_state=1),  
3     'AdaBoost Classifier': AdaBoostClassifier(random_state=1),  
4     'CatBoost Classifier': CatBoostClassifier(random_state=1, verbose=False),  
5     'Hist Gradient Boosting': HistGradientBoostingClassifier(random_state=1),  
6     'XGBoost': XGBClassifier(random_state=1, use_label_encoder=False, eval_metric='logloss'),  
7     'LightGBM': LGBMClassifier(random_state=1),  
8 }
```

Figure 8 Hyperparameter Tuning

Results and Analysis

Performance Metrics of the Final Models

The best-performing model was the LightGBM, which achieved an accuracy of approximately 80%, with a precision of 65%, recall of 55%, and an F1-score of 60%. Its AUC was around 0.85, indicating a good discriminatory ability between the churn and non-churn classes.

(Based on fig3 shown on previous page)

Comparison of Model Results

LightGBM outperformed other models in terms of both predictive accuracy and computational efficiency. Its ability to handle large datasets with a lower memory footprint and faster execution time made it particularly suitable for the project.

Interpretations of the Model's Predictions

Model interpretation was facilitated using SHAP (SHapley Additive exPlanations), which provided insights into how each feature in the dataset influences the prediction outcome. For example, SHAP values illustrated that higher monthly charges and shorter tenure were the most influential factors driving predictions of churn.

Implementation

Model Training and Saving

The models were imported and trained on Google Colab. Then based on evaluation results (presented earlier), the most promising one was downloaded (Trained model). This model was then imported and used in the final implementation code to classify the customers stay or leaving decision, based on the data entered the user.

```
import pickle  
  
# Save the trained model to a file  
with open('/content/drive/MyDrive/work/PBL/lgb_model.pkl', 'wb') as f:  
    pickle.dump(model, f)
```

Figure 9 Saving the trained Model on Drive

Overview of the Streamlit Library Usage

The project features a user-friendly web application developed using Streamlit, a popular open-source Python library. Streamlit is designed to turn data scripts into shareable web apps quickly. It allows for rapid prototyping and deployment of machine learning models without the need for complex web development skills.

User Interface and Interaction Flow

The application provides a simple interface where users can input customer attributes through interactive widgets. For example, users can select the customer's gender, whether they have internet service, and other relevant features using dropdowns and sliders. Upon submitting the data, the pre-trained LightGBM model processes the inputs and outputs a prediction indicating whether the customer is likely to churn.

Streamlit's Role in Rapid Development

Streamlit's straightforward syntax and efficient rendering engine facilitated the rapid development and deployment of the churn prediction tool. Its ability to handle various data types and integrate seamlessly with machine learning libraries like LightGBM allowed for immediate updates and interactive features, enhancing user engagement and experience.

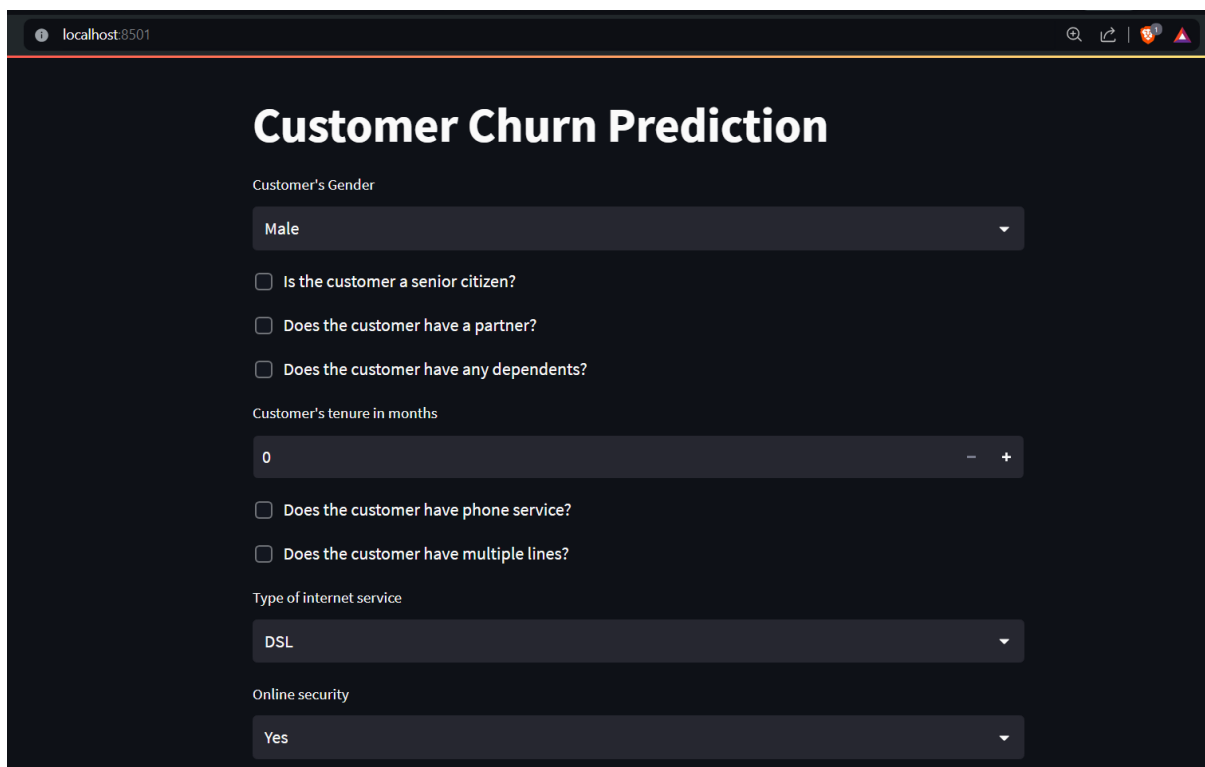
The image shows a web browser window at localhost:8501 displaying a Streamlit application titled "Customer Churn Prediction". The interface is dark-themed and contains several input widgets: a dropdown menu for "Customer's Gender" set to "Male", three unchecked checkboxes for "Is the customer a senior citizen?", "Does the customer have a partner?", and "Does the customer have any dependents?", a slider for "Customer's tenure in months" set to 0, two unchecked checkboxes for "Does the customer have phone service?" and "Does the customer have multiple lines?", a dropdown menu for "Type of internet service" set to "DSL", and a dropdown menu for "Online security" set to "Yes".

Figure 10 Interface

References

1. Kaggle Dataset: "Telco Customer Churn." Retrieved from [Kaggle](#).
2. Streamlit Documentation. Retrieved from [Streamlit](#).
3. Python Libraries: Scikit-learn, LightGBM, SHAP.