DTM 2025 - MACHINE LEARNING

Predicting Customer Churn with Supervised Learning

Model Comparison and Feature Analysis

Mohammad Atabaki
Shahab Aminraoufpour

problem statment

- Handling customer churn is one of the major challenges faced by modern organizations, especially in competitive and subscription-based markets.
- Acquiring new customers is significantly more expensive than retaining existing ones, making churn a direct threat to profitability.
- Organizations require data-driven and predictive solutions to identify at-risk customers early and take proactive retention actions.

project objective

The primary objective of this project is to design, implement, and evaluate a supervised machine learning system for customer churn prediction using structured customer data.

project aims to

- Accurately predict whether a customer will churn based on demographic, behavioral, and subscription-related features.
- Formulate churn prediction as a binary classification problem
- Compare multiple ensemble learning models (Random Forest, XGBoost, and LightGBM)
- Evaluate model performance using robust metrics (Accuracy, Precision, Recall, F1-score)

Dataset

import data

The initial stage involves importing an archive called Customer Churn Dataset from Kaggle provided in two CSV files: **a training set** and a **test set**.

- The file contains 440,882 labeled customer records and is used to train the churn prediction models.
- Each record includes demographic, behavioral, and subscription features, such as **age**, **tenure**, **usage frequency**, **support calls**, **payment delay**, **contract type**, and **total spend**.

Holdout mthod :

To build reliable and generalizable machine learning models, the data is split into training (80%) and testing (20%) sets.

- **The training set is used to learn the model**
- **the testing set is used to evaluate its performance.**

Data understanding

Data profiling

Exploratory analysis was performed to examine **feature distributions** and **class balance**.

Numerical and categorical variables analyze to identify **key patterns**, **relationships with churn**, and **potential anomalies**.

Data distribution

Analysis of how values are spread or dispersed within a particular column or dataset.

Data preprocesing

Categorical **features were encoded**, **numerical features were normalized**, and **missing values and outliers were handled using** a consistent preprocessing pipeline to improve data quality and prevent data leakage.

Data profiling

This step allowed us to understand the dataset structure, feature semantics, data types, and the presence of missing values, forming a critical foundation for subsequent analysis.

```
<class 'pandas.core.frame.DataFrame'>
Index: 440832 entries, 0 to 440832
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   CustomerID         440832 non-null  float64
 1   Age                440832 non-null  float64
 2   Gender             440832 non-null  object 
 3   Tenure             440832 non-null  float64
 4   Usage Frequency    440832 non-null  float64
 5   Support Calls      440832 non-null  float64
 6   Payment Delay      440832 non-null  float64
 7   Subscription Type  440832 non-null  object 
 8   Contract Length    440832 non-null  object 
 9   Total Spend        440832 non-null  float64
 10  Last Interaction   440832 non-null  float64
 11  Churn              440832 non-null  float64
dtypes: float64(9), object(3)
memory usage: 43.7+ MB
```

Dataset inspection identified one fully empty row, which was removed to ensure data consistency before preprocessing and modeling.

Data distribution

By analyzing data distributions, we gained insights into key statistics (e.g., average age and maximum values), improving our understanding of the variables.

	CustomerID	Age	Tenure	Usage Frequency	Support Calls	Payment Delay	Total Spend	Last Interaction	Churn
count	440832.000000	440832.000000	440832.000000	440832.000000	440832.000000	440832.000000	440832.000000	440832.000000	440832.000000
mean	225398.667955	39.373153	31.256336	15.807494	3.604437	12.965722	631.616223	14.480868	0.567107
std	129531.918550	12.442369	17.255727	8.586242	3.070218	8.258063	240.803001	8.596208	0.495477
min	2.000000	18.000000	1.000000	1.000000	0.000000	0.000000	100.000000	1.000000	0.000000
25%	113621.750000	29.000000	16.000000	9.000000	1.000000	6.000000	480.000000	7.000000	0.000000
50%	226125.500000	39.000000	32.000000	16.000000	3.000000	12.000000	661.000000	14.000000	1.000000
75%	337739.250000	48.000000	46.000000	23.000000	6.000000	19.000000	830.000000	22.000000	1.000000
max	449999.000000	65.000000	60.000000	30.000000	10.000000	30.000000	1000.000000	30.000000	1.000000

```
df.sample(5)
```

	CustomerID	Age	Gender	Tenure	Usage Frequency	Support Calls	Payment Delay	Subscription Type	Contract Length	Total Spend	Last Interaction	Churn
302275	309389.0	21.0	Male	58.0	1.0	1.0	1.0	Standard	Quarterly	676.03	23.0	0.0
25565	25574.0	39.0	Male	8.0	20.0	6.0	20.0	Standard	Quarterly	769.00	10.0	1.0
330533	337648.0	31.0	Male	5.0	8.0	1.0	20.0	Premium	Annual	718.17	11.0	0.0
46041	46656.0	29.0	Female	36.0	4.0	1.0	5.0	Premium	Quarterly	992.00	29.0	1.0
211050	216337.0	23.0	Male	47.0	18.0	2.0	30.0	Premium	Annual	641.14	6.0	1.0

Data preprocessing

1 Drop the unnecessary columns

There are some features that we can drop, as they are not relevant for the goal:

- CustomerID

```
df = df.drop(['CustomerID'], axis=1)
```

2 Checking correlations

Correlation analysis shows that frequent support calls and payment delays are associated with higher churn, while higher total spending is linked to lower churn. No strong correlations among features were found, so all variables were retained for modeling.

3 Encoding

Encode Gender into boolean values
Categorical features were converted into binary variables using one-hot encoding, with one category used as a reference to avoid multicollinearity. This ensured all features were suitable for model training.

4 Normalization

Normalize the range of independent variables or features of data.

Machine learing Modeling

Random Forest

XGboost

LightGBM

Machine learning modeling

Random forest

The Random Forest Classifier builds multiple Decision Trees and combines their outputs to make more accurate and stable predictions. Each tree is trained on a random subset of the data and considers a random subset of features when splitting nodes, which helps reduce overfitting and improves generalization. The final prediction is made by majority voting among the trees.

Best accuracy obtained is **0.99976**

Random Forest

```
# Define the base model
rf = RandomForestClassifier(random_state=42,n_jobs=-1)

# Define the hyperparameter search space
param_dist = {
    'n_estimators': [100, 200, 300, 500],
    'max_depth': [None, 10, 20, 30, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
    'bootstrap': [True, False]
}

# Randomized Search CV
rf_random = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_dist,
    n_iter=20,
    cv=5,
    scoring='accuracy',
    random_state=42,
    verbose=2,
    n_jobs=-1
)
```

Machine learning modeling

Random forest

1 Hyperparameter Tuning

- Model performance strongly depends on parameters such as:
1. number of trees
2. tree depth
3. minimum samples per split
- Default settings may lead to suboptimal performance or overfitting.

2 Randomized Search & Validation

- Instead of exhaustively testing all parameter combinations, Randomized Search was used.
- This approach:
1. randomly samples combinations from a predefined parameter space
2. is computationally efficient
3. finds strong configurations with significantly lower cost than Grid Search

- 5-fold cross-validation was applied during tuning.
- Each configuration was evaluated across multiple data splits to ensure:
  - stable performance
  - reduced sensitivity to data noise

3 Evaluation Metrics & Overfitting

- Model performance was evaluated using:
1. Accuracy
2. ROC-AUC
3. Precision, Recall, and F1-score
- ROC-AUC was emphasized as it reflects the model's ability to distinguish churners from non-churners.
- Model robustness was evaluated by comparing training and test performance.
- Similar ROC-AUC scores indicate strong generalization and no overfitting.

Machine learning modeling

XGBoost

XGBoost (Extreme Gradient Boosting) is a boosting-based ensemble model that builds decision trees sequentially.
Each new tree focuses on correcting the errors made by previous trees.
This iterative learning process allows XGBoost to capture complex patterns and subtle relationships in the data.
Regularization techniques are built into the model to control complexity and prevent overfitting and effectively capture non-linear interactions between customer behavior variables.
Best accuracy obtained is 0.99985

Xgboost

```
# XGBoost classifier (binary churn)
xgb_clf = xgb.XGBClassifier(
    objective="binary:logistic",
    eval_metric="logloss",
    tree_method="hist",
    n_jobs=-1,
    random_state=42
)

# Hyperparameter search space
param_dist = {
    "n_estimators": [200, 400, 600, 800],
    "learning_rate": [0.01, 0.05, 0.1, 0.2],
    "max_depth": [3, 4, 5, 6, 8],
    "min_child_weight": [1, 3, 5, 7],
    "subsample": [0.7, 0.8, 0.9, 1.0],
    "colsample_bytree": [0.7, 0.8, 0.9, 1.0],
    "gamma": [0, 0.1, 0.2, 0.5],
    "reg_alpha": [0, 0.01, 0.1, 1.0],
    "reg_lambda": [1.0, 1.5, 2.0]
}

# Randomized Search CV
xgb_random = RandomizedSearchCV(
    estimator=xgb_clf,
    param_distributions=param_dist,
    n_iter=25,
    scoring="roc_auc",
    cv=5,
    verbose=2,
    random_state=42,
    n_jobs=-1
)
```

Machine learning modeling

XGBoost

Hyperparameter Optimization

- Model performance depends on multiple interacting parameters (e.g. tree depth, learning rate).
- Randomized Search was used to efficiently explore the hyperparameter space.
- Compared to exhaustive search, this approach:

1. reduces computational cost
2. identifies strong configurations more quickly

- 5-fold cross-validation ensures stable and reliable parameter selection.

Model Evaluation & Generalization

- Model performance was evaluated using:

1. ROC-AUC
2. Accuracy and classification metrics

- ROC-AUC was emphasized as it measures the model's ability to separate churners from non-churners.
- Similar training and test ROC-AUC scores indicate that strong generalization and no significant overfitting

Unlike Random Forest, XGBoost learns sequentially from its own mistakes, which allows it to achieve very high performance while still controlling overfitting.

Machine learning modeling

LightGBM

- LightGBM is a gradient boosting framework optimized for efficiency and scalability.
- It builds decision trees in a leaf-wise manner, allowing the model to focus on the most informative splits.
- This strategy enables faster training and strong performance, especially on large datasets.
- LightGBM includes built-in regularization to control model complexity.
- Capable of capturing complex, non-linear patterns in customer behavior.

Best accuracy obtained is 0.99982

Lightgbm

```
# LightGBM classifier (binary churn)
lgbm_clf = lgb.LGBMClassifier(
    objective="binary",
    n_jobs=-1,
    random_state=42
)

# Hyperparameter search space
param_dist = {
    "n_estimators": [300, 500, 800, 1200],
    "learning_rate": [0.01, 0.03, 0.05, 0.1],
    "num_leaves": [15, 31, 63, 127],
    "max_depth": [-1, 4, 6, 8, 12],
    "min_child_samples": [10, 20, 30, 50, 100],
    "subsample": [0.7, 0.8, 0.9, 1.0],
    "colsample_bytree": [0.7, 0.8, 0.9, 1.0],
    "reg_alpha": [0.0, 0.01, 0.1, 1.0],
    "reg_lambda": [0.0, 0.1, 1.0, 2.0]
}

# Randomized Search CV
lgbm_random = RandomizedSearchCV(
    estimator=lgbm_clf,
    param_distributions=param_dist,
    n_iter=25,
    scoring="roc_auc",
    cv=5,
    verbose=2,
    random_state=42,
    n_jobs=-1
)
```

Machine learning modeling

LightGBM

LightGBM improves boosting efficiency by growing trees where they matter most, achieving high performance with fast training and strong generalization.

Hyperparameter Optimization

- LightGBM performance depends on parameters controlling:

1. tree structure
2. learning speed
3. model regularization

- Randomized Search was used to explore the hyperparameter space efficiently.
- This approach reduces training time and avoids exhaustive parameter combinations
- 5-fold cross-validation ensures robust and stable parameter selection.

Model Evaluation & Generalization

- Model performance was evaluated using:

1. ROC-AUC
2. Accuracy and classification metrics

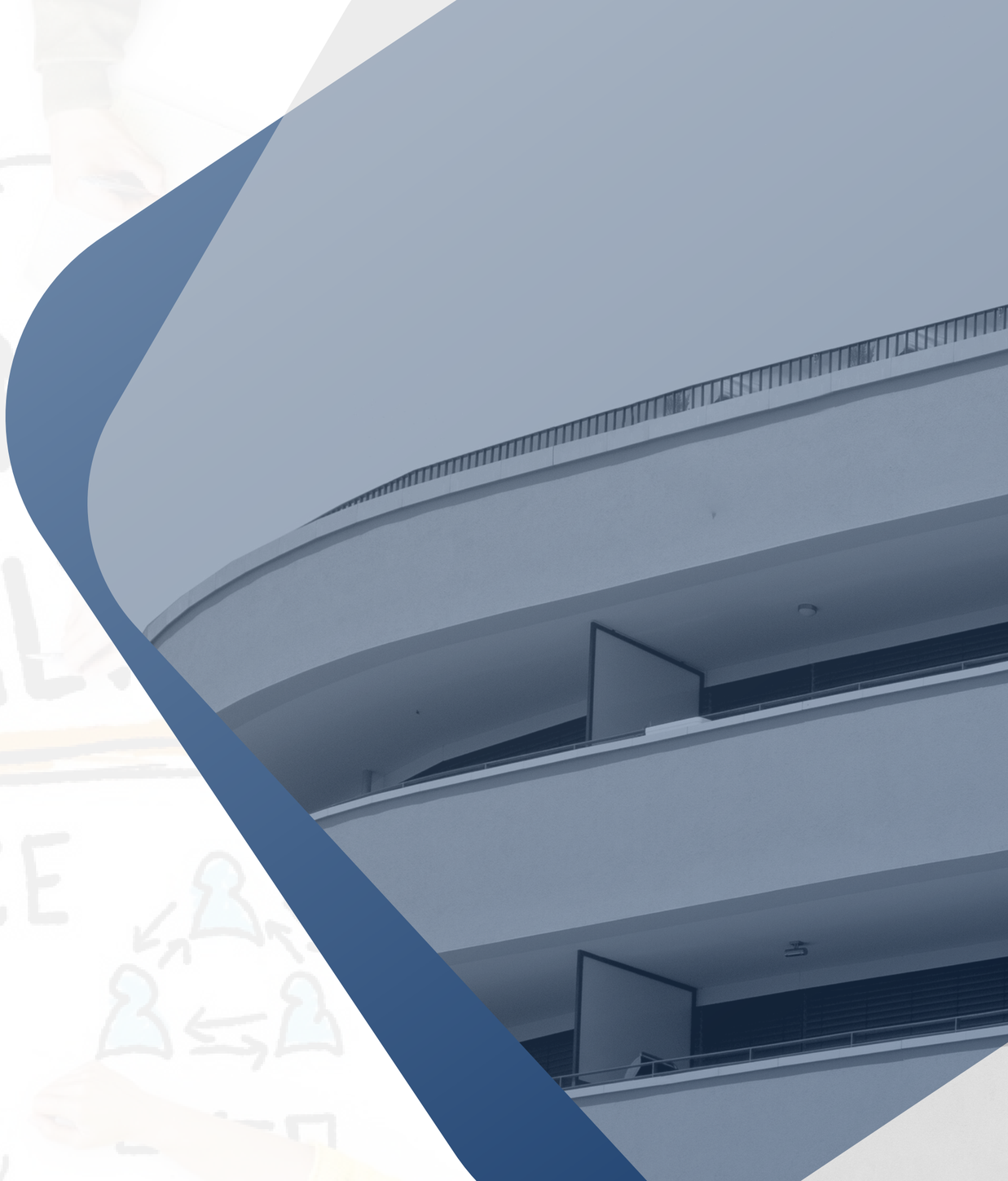
- ROC-AUC was prioritized to assess the model's ability to rank churn risk.
- Comparable training and test ROC-AUC values indicate:

1. strong generalization
2. absence of overfitting

Conclusion

- This study evaluated **ensemble machine learning** models for customer churn prediction using a unified preprocessing and evaluation framework. **Random Forest**, **XGBoost**, and **LightGBM** all achieved exceptionally high performance, with **accuracy exceeding 99.97%**, confirming the effectiveness of ensemble methods for churn prediction.
- Among the models, **XGBoost** delivered the best overall performance
- Beyond accuracy, the analysis showed that churn is mainly driven by **behavioral** and **service-related** f**actors,** including **customer engagement**, **tenure** and **payment behavior**.
- These insights demonstrate that machine learning models can both accurately predict churn and support proactive, **data-driven retention strategies**, making **churn prediction a strategic tool for long-term business value**.

Thank you for listening



Mohammad Atabaki
Shahab Aminraoufpour