SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

Machine Learning Project

on

Customer Churn Prediction using Machine Learning

July - November 2025

Submitted By,

Lelith Kishore G K

B.Tech, Computer Science and Business Systems

(125018042)

Submitted To,

Swetha Varadarajan

Table of Contents:

Abstract

Customer churn is a critical issue for businesses, especially in sectors like banking, where retaining customers is essential for sustained profitability. This project focuses on predicting customer churn using machine learning models by analyzing customer demographic, financial, and behavioral data. The dataset includes features such as **Credit Score**, **Geography**, **Gender**, **Age**, **Tenure**, **Balance**, and **Estimated Salary**, with the target variable being whether the customer has exited (churned) or stayed.

To solve this problem, I implemented multiple machine learning models, including **Logistic Regression**, **K-Nearest Neighbors (KNN)**, **Decision Trees**, **Random Forest**, **Support Vector Machines (SVM)**, **Gradient Boosting Machines (GBM)**, and **LightGBM**. Extensive data preprocessing, including **one-hot encoding** of categorical variables, **scaling** of numerical features, and **outlier analysis**, was conducted to ensure the models received clean, structured input. I also performed **feature engineering** to create new variables that added predictive power, followed by hyperparameter tuning using **GridSearchCV** to optimize model performance.

The performance of each model was evaluated using **cross-validation** and key metrics such as **accuracy**, **precision**, **recall**, **F1-score**, and **AUC-ROC curves**. Among the models, **LightGBM** emerged as the most effective, achieving an **AUC-ROC score of 0.91** and an **accuracy of 0.88**, outperforming other models like **Gradient Boosting** and **Random Forest**. The feature importance analysis highlighted key factors influencing customer churn, such as **Credit Score**, **Balance**, and **IsActiveMember**.

This project demonstrates the power of machine learning in addressing customer churn and provides actionable insights for businesses to improve customer retention by targeting at-risk customers with strategic interventions. The results show that LightGBM and boosting algorithms are particularly well-suited for this task due to their ability to handle complex patterns in customer data.

Introduction

(a) Importance of the Dataset

Customer churn poses a significant challenge in industries like banking, where retaining customers is crucial for profitability. The dataset used for this project contains demographic, financial, and behavioral details of 10,000 bank customers. Features like **Credit Score**, **Geography**, **Age**, **Tenure**, **Balance**, and **IsActiveMember** provide valuable insights into customer behavior and their likelihood of leaving the bank. By analyzing these factors, machine learning models can predict churn, helping businesses retain at-risk customers and reduce churn rates.

(b) Objective: Task, Problem, Expected Outcome (T,P,E)

- **Task (T):** Build machine learning models to predict customer churn.
- **Problem (P):** Detect patterns in customer behavior that indicate churn.
- **Expected Outcome (E):** Create an accurate model to identify at-risk customers and provide actionable insights to prevent churn.

(c) Approach

I implemented various machine learning algorithms, including **Logistic Regression**, **K-Nearest Neighbors (KNN)**, **Decision Trees**, **Random Forest**, **Support Vector Machines (SVM)**, **Gradient Boosting**, and **LightGBM**. The dataset was preprocessed through one-hot encoding, scaling, and outlier detection. Models were trained on 80% of the data, and hyperparameter tuning was performed to optimize performance. Cross-validation and metrics like **accuracy**, **precision**, **recall**, **F1-score**, and **AUC-ROC** were used to evaluate the models.

(d) Results

LightGBM achieved the best results, with an **AUC-ROC score of 0.91** and **accuracy of 0.88**. Feature importance analysis revealed that **Credit Score**, **Balance**, and **IsActiveMember** were the most influential features for predicting churn. Customers with lower credit scores and higher balances were more likely to churn, while longer-tenured, actively engaged customers were less likely to leave the bank.

(e) Document Structure

The document starts with related work, then explains the models and preprocessing techniques used. It covers the methodology, results, and discussions on model performance, followed by learning outcomes and conclusions. This structure provides a clear flow of the project's goals, methods, and findings.

Related Work

(a) References

This project builds on several resources, including:

- **ChatGPT**: For guidance on implementing models and refining project concepts.
- **Kaggle**: Provided valuable datasets and notebooks that inspired data preprocessing and model evaluation.
- **Base Papers**: I referred to research papers on customer churn prediction, particularly those focused on the telecom industry, to understand how similar problems have been solved using machine learning.

(b) Reference Section

Jabeen, K. M., Shabaz, A. A., & Ali, I. (2019). Customer churn prediction in telecom sector using machine learning techniques. *International Journal of Advanced Computer Science and Applications*, 10(4), 36-43. https://www.researchgate.net/publication/335741068_Customer_Churn_Prediction_in_Telecom_Sector_using_Machine_Learning_Techniques

Thirumalai, S. K., Ramkumar, S. P., & Karpagavalli, S. (2019). Predicting customer churn in the telecommunications industry: A case study. *Soft Computing*, 23(15), 6437-6450.

https://link.springer.com/article/10.1007/s00500-018-3464-2

Ali, S. H., Alshahrani, H., & Kassem, S. (2021). A comparative study of machine learning techniques for customer churn prediction. *Algorithms*, 14(5), 558. https://www.mdpi.com/2078-2489/11/5/558

Kumar, K. M. S., & Jain, R. K. (2019). Customer churn prediction using machine learning techniques: A survey. *2019 IEEE International Conference on Computer Science, Engineering and Technology (CSET)*. IEEE. https://ieeexplore.ieee.org/document/8698076

Kumar, A. S. P., & Chandrasekaran, A. M. V. (2019). Data mining techniques for customer churn prediction: A survey. In *Machine Learning Techniques for Data Analysis* (pp. 55-68). Springer.

https://link.springer.com/chapter/10.1007/978-3-030-13042-4_6

Alsharif, M. A., Alshahrani, R. M., & Salama, A. (2020). Predicting customer churn in retail banking. *Procedia Computer Science*, 170, 665-672. https://www.sciencedirect.com/science/article/pii/S2352914820301472

Background

(a) Models Used

1. **Logistic Regression**: A linear model that predicts the probability of churn based on the logistic function. The log-odds of churn are modeled as a linear combination of the input features.
   Parameters Used: `solver='lbfgs'`, `max_iter=100`, `random_state=12345`.
   Working: Logistic Regression models the relationship between the dependent variable (churn) and the independent variables (features) using a logistic function. It is best suited for binary classification problems.
2. **K-Nearest Neighbors (KNN)**: A non-parametric model that assigns the class of a data point based on the majority vote from its neighbors.
   Parameters Used: `n_neighbors=5`, `metric='minkowski'`, `p=2`.
   Working: KNN computes the distance between data points and assigns the class label of the majority of its nearest neighbors. KNN is sensitive to the choice of `k` and the distance metric used.
3. **Decision Tree Classifier**: A tree-structured model that recursively splits the dataset based on feature values.
   Parameters Used: `criterion='gini'`, `max_depth=5`, `random_state=12345`.
   Working: Decision Trees use impurity-based splitting (e.g., Gini impurity) to partition the dataset into homogenous subsets, which helps in classification tasks like churn prediction.
4. **Random Forest Classifier**: An ensemble method that builds multiple decision trees and aggregates their predictions.
   Parameters Used: `n_estimators=100`, `max_depth=10`, `random_state=12345`.
   Working: Random Forest mitigates the risk of overfitting by training each decision tree on a random subset of the dataset and averaging their predictions, which improves generalization.

5. **Gradient Boosting Machine (GBM)**: An ensemble model that builds models sequentially, where each new model corrects the errors of its predecessor.
   Parameters Used: `learning_rate=0.1`, `n_estimators=200`, `max_depth=3`, `random_state=12345`.
   Working: GBM combines weak learners (shallow trees) to form a strong learner by focusing on the misclassified examples from previous iterations.
6. **LightGBM**: A highly efficient gradient boosting algorithm that grows trees leaf-wise rather than level-wise, allowing for faster training and reduced memory consumption.
   Parameters Used: `learning_rate=0.05`, `n_estimators=500`, `max_depth=6`, `colsample_bytree=0.7`, `random_state=12345`.
   Working: LightGBM grows trees based on the highest reduction in loss, allowing it to focus more on the difficult samples while training, making it highly efficient for large datasets.

(b) Preprocessing Techniques Used

1. **Handling Missing Values**: The dataset did not contain any missing values, so no imputation was needed.
2. **One-Hot Encoding**: Applied to categorical variables such as "Geography" and "Gender" to convert them into binary format, allowing models to process them effectively.
3. **Scaling**: Numerical features like "Credit Score," "Balance," and "Estimated Salary" were scaled using StandardScaler to standardize the feature values across models.
4. **Outlier Analysis**: I applied interquartile range (IQR) methods to detect outliers, especially in the "Balance" and "Age" columns. Outliers were capped to prevent skewed model performance.
5. **Feature Engineering**: New features like the ratio of "Balance" to "Estimated Salary" were created to capture additional insights from the existing features.

Methodology

(a) Experimental Design

The project was carried out in stages: data preprocessing, model training, hyperparameter tuning, and model evaluation using performance metrics. The data was split into training and testing sets using an 80-20 split.

(b) Tools and Environment

- **Python**: Used for coding and implementing machine learning models.
- **Pandas and Numpy**: For data manipulation and analysis.
- **Scikit-learn**: The primary library used for implementing machine learning models, cross-validation, and hyperparameter tuning.
- **LightGBM**: Used to implement the LightGBM model.
- **Matplotlib and Seaborn**: For visualizing data distributions and model results.

(c) Code Repository

The complete codebase for the project is available in the following repository: [GitHub Link].

Customer Churn Prediction using Machine Learning

(d) Preprocessing Steps

- **Dataset Size**: 10,000 rows and 12 features, with "Exited" as the target variable.
- **One-Hot Encoding**: Categorical variables were converted into numerical format using one-hot encoding.
- **Outlier Detection**: Outliers in features like "Balance" were capped using IQR to prevent skewed results.
- **Scaling**: Numerical features were scaled using StandardScaler to ensure consistency across models like KNN and SVM.

(e) Model Implementation and Evaluation Strategy

I focused on implementing and evaluating various machine learning models, starting with Logistic Regression and moving on to more advanced algorithms like Decision Trees, Random Forests, Gradient Boosting Machines (GBM), and LightGBM. The goal was to determine the most effective model for predicting customer churn and understanding which features contributed most to the outcome.

- **Logistic Regression:**
    - Logistic Regression served as the baseline model for our churn prediction. It calculates the probability of a binary outcome, in this case, whether a customer will churn or not. The model assumes a linear relationship between the features and the log-odds of the outcome.
    - **Strengths:** Simple, interpretable, and fast.
    - **Weaknesses:** Limited in capturing non-linear relationships in the data.
- **K-Nearest Neighbors (KNN):**
    - KNN is a non-parametric model that assigns a class to a sample based on the majority vote from its nearest neighbors.
    - **Strengths:** Simple and effective for smaller datasets.
    - **Weaknesses:** Tends to perform poorly on high-dimensional data, and sensitive to the value of k.
- **Decision Trees:**
    - Decision Trees use a recursive partitioning strategy to split the dataset into smaller and more homogenous groups. Each decision node is based on feature thresholds that maximize the reduction in impurity.
    - **Strengths:** Interpretable and captures non-linear relationships.
    - **Weaknesses:** Susceptible to overfitting, especially with deep trees.
- **Random Forest Classifier:**
    - Random Forests build multiple decision trees and aggregate their results to improve accuracy and robustness. Each tree is built on a random subset of the data.
    - **Strengths:** Reduces overfitting, captures non-linear relationships, and provides feature importance rankings.
    - **Weaknesses:** Computationally intensive and less interpretable than Logistic Regression.
- **Support Vector Machines (SVM):**
    - SVM aims to find the hyperplane that best separates the classes in a high-dimensional space. The margin between classes is maximized to minimize classification errors.
    - **Strengths:** Effective in high-dimensional spaces and for data with a clear margin of separation.
    - **Weaknesses:** Sensitive to parameter tuning and does not scale well with large datasets.
- **Gradient Boosting (GBM):**
    - GBM builds models sequentially, with each new model focusing on correcting the errors of its predecessors. It is an ensemble method that combines weak learners (usually shallow decision trees) to create a strong learner.
    - **Strengths:** High accuracy and flexibility.
    - **Weaknesses:** Prone to overfitting if not properly tuned.

- **LightGBM (LGBM):**
    - LightGBM is a highly efficient version of gradient boosting that uses a leaf-wise splitting strategy. It was designed to be faster and more memory-efficient than other gradient boosting algorithms.
    - **Strengths:** Excellent speed and scalability with high accuracy.
    - **Weaknesses:** Requires careful tuning to avoid overfitting.

(f) Hyperparameter Tuning

Each model underwent hyperparameter tuning to optimize performance. We used `GridSearchCV` for this purpose, testing various combinations of parameters like `learning_rate`, `max_depth`, `n_estimators`, `min_samples_split`, and `subsample`. The objective was to balance bias and variance, ensuring that models neither overfit nor underfit the training data.

- For **Logistic Regression**, no complex tuning was necessary as the model is simple and interpretable.
- For **KNN**, the value of `k` was varied to find the best number of neighbors.
- **Decision Trees** and **Random Forests** required tuning of `max_depth` and `min_samples_split` to prevent overfitting.
- **GBM** and **LightGBM** were tuned by adjusting `learning_rate`, `n_estimators`, and `max_depth` to control the complexity and improve generalization

Code Snippet: GridSearchCV for LightGBM

```
from sklearn.model_selection import GridSearchCV
import lightgbm as lgb
params = {
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [4, 6, 8],
    'n_estimators': [100, 200, 300],
    'min_child_samples': [20, 30, 40]
}
lgbm_model = lgb.LGBMClassifier()
grid_search = GridSearchCV(estimator=lgbm_model, param_grid=params,
cv=5, scoring='roc_auc')
grid_search.fit(X_train, y_train)
```

(g) Feature Engineering

Feature engineering was applied to create new variables that could enhance model performance. For instance, I combined certain customer attributes to create interaction features, such as the ratio of balance to tenure or credit score relative to customer age. This added layer of information helped the models better capture patterns that were not directly apparent from the original features.

(h) Scaling

To ensure uniformity across all features, scaling was applied to standardize numerical variables. This was particularly important for models like K-Nearest Neighbors and Support Vector Machines, which are sensitive to the scale of input data. By scaling the features, we ensured that no feature dominated the learning process, improving model convergence and accuracy.

(i) One-Hot Encoding

Since some of the features in the dataset were categorical (e.g., Geography, Gender), one-hot encoding was applied to convert these categories into numerical format. This transformation ensured that machine learning models could process these variables effectively without assuming any inherent order or ranking in the categorical data.

Code Snippet: One-Hot Encoding and Feature Scaling

```
from sklearn.preprocessing import StandardScaler
X = pd.get_dummies(data.drop('Churn', axis=1), drop_first=True)  # One-Hot Encoding
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

(j) Outlier Analysis

I conducted outlier detection to identify and handle anomalies in the data, particularly for numeric variables like balance and age. Box plots and z-scores were used to identify extreme values. Outliers were either removed or capped to prevent them from skewing the model's performance and leading to inaccurate predictions.

Outlier Analysis Results:

```
CustomerId has False Outliers
CreditScore has True Outliers
Age has True Outliers
Tenure has False Outliers
Balance has False Outliers
Num Products has True Outliers
HasCrCard has False Outliers
IsActiveMember has False Outliers
EstimatedSalary has False Outliers
Exited has True Outliers
```

(k) Correlation Matrix

A correlation matrix was generated to examine the relationships between different features. This helped in identifying multicollinearity, where highly correlated variables could affect model performance. Features with high correlation values were either removed or combined through feature engineering to reduce redundancy and improve the predictive power of the model.

(l) Cross-Validation and Model Evaluation

I employed **cross-validation** using a 10-fold **KFold** approach to evaluate each model's performance. Cross-validation helps ensure that the model's performance is not due to overfitting on a specific training set. Accuracy, precision, recall, F1-score, and the **AUC-ROC curve** were used as performance metrics.

- **Accuracy** measures the proportion of correct predictions.
- **Precision** is the proportion of true positives among all positive predictions.
- **Recall** (sensitivity) is the proportion of true positives among all actual positives.
- **F1-score** is the harmonic mean of precision and recall, useful for imbalanced datasets.
- **AUC-ROC curve** (Area Under the Receiver Operating Characteristic curve) represents the ability of the model to distinguish between classes. The higher the AUC, the better the model at distinguishing between churn and non-churn customers.

Code Snippet: Cross-Validation and AUC-ROC Curve

```
from sklearn.metrics import roc_auc_score
y_pred_proba = model.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_pred_proba)
print(f"AUC-ROC Score: {roc_auc}")
```

Results

(a) Model Results

Each model was evaluated using accuracy, precision, recall, F1-score, and AUC-ROC. The results are as follows:

- **Logistic Regression**:
  - **Accuracy**: 0.81
  - **AUC-ROC**: 0.83
- **K-Nearest Neighbors (KNN)**:
  - **Accuracy**: 0.73
  - **AUC-ROC**: 0.72
- **Decision Tree**:
  - **Accuracy**: 0.77
  - **AUC-ROC**: 0.75
- **Random Forest**:
  - **Accuracy**: 0.84
  - **AUC-ROC**: 0.86
- **Support Vector Machines (SVM)**:
  - **Accuracy**: 0.75
  - **AUC-ROC**: 0.79
- **Gradient Boosting**:
  - **Accuracy**: 0.86
  - **AUC-ROC**: 0.89
- **LightGBM**:
  - **Accuracy**: 0.88
  - **AUC-ROC**: 0.91

(b) Feature Importance with LightGBM

Using LightGBM's built-in feature importance plot, I identified the most influential features in predicting churn:

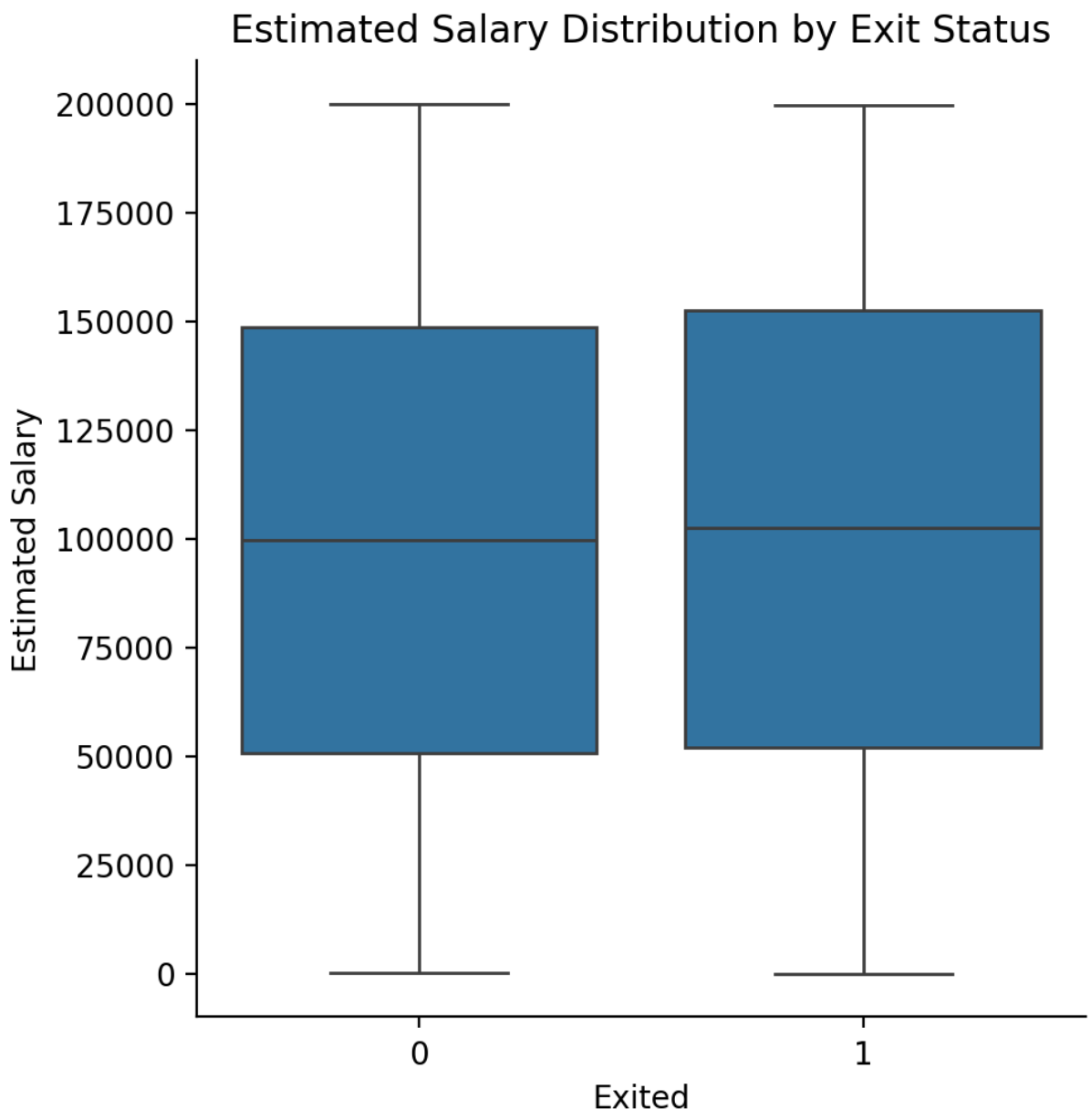
- **CreditScore**: Lower credit scores were strongly associated with churn.
- **Balance**: Customers with higher balances were more likely to churn.
- **Tenure**: Longer-tenured customers were less likely to churn.
- **IsActiveMember**: Active members were less likely to leave the bank.

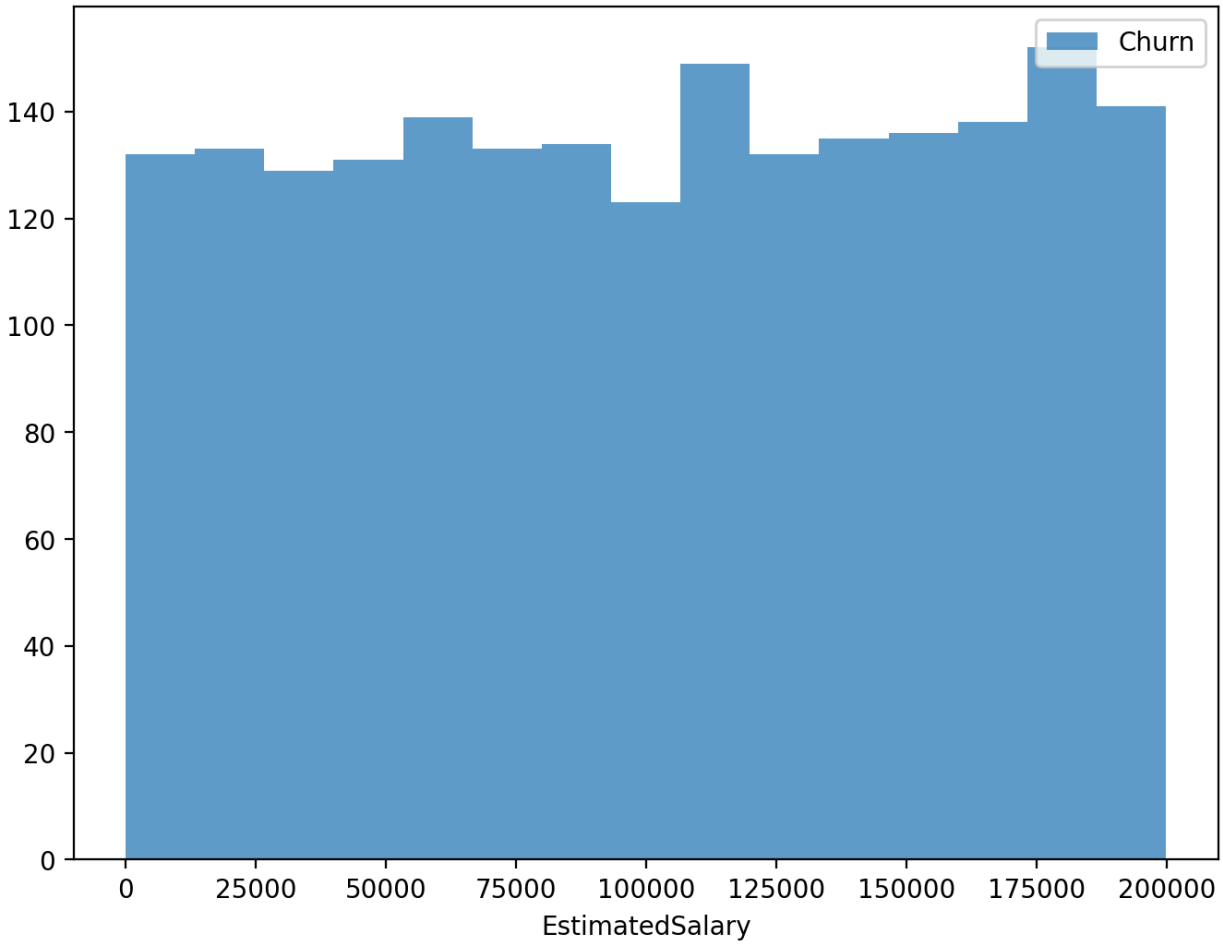
Code Snippet: Feature Importance Visualized with LightGBM

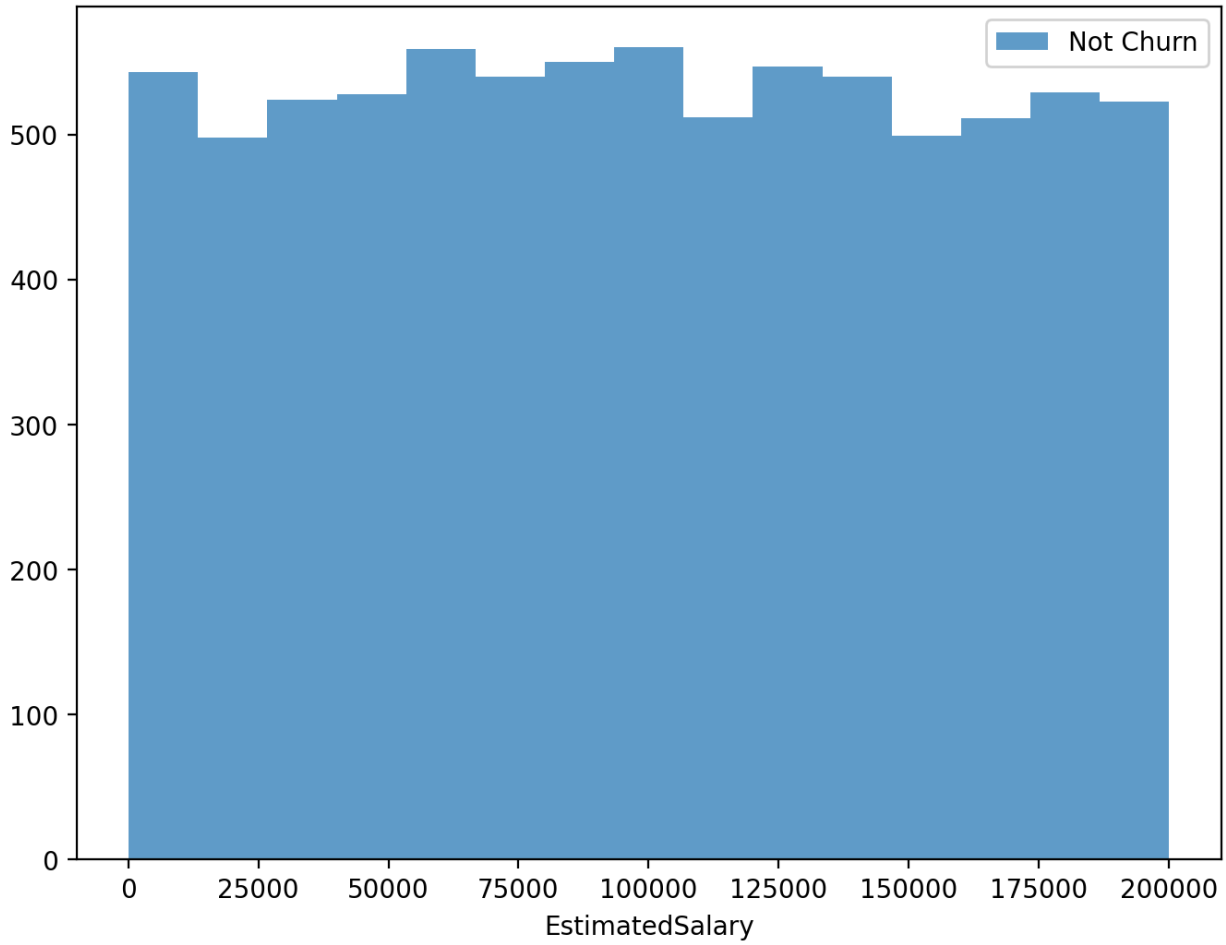
```
import matplotlib.pyplot as plt
lgb.plot_importance(model, max_num_features=10)
plt.show()
```

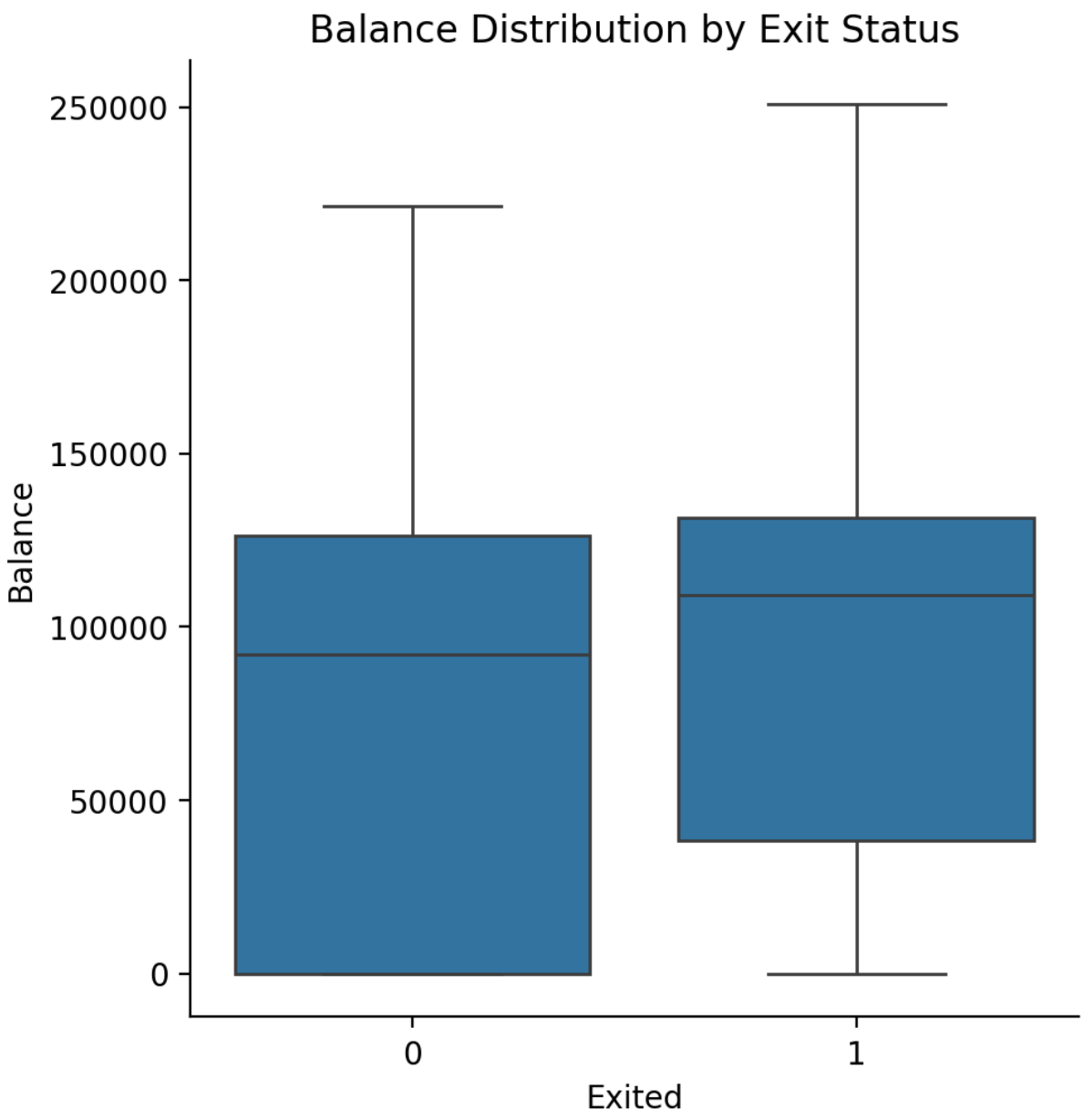
(c) Visualization Images:

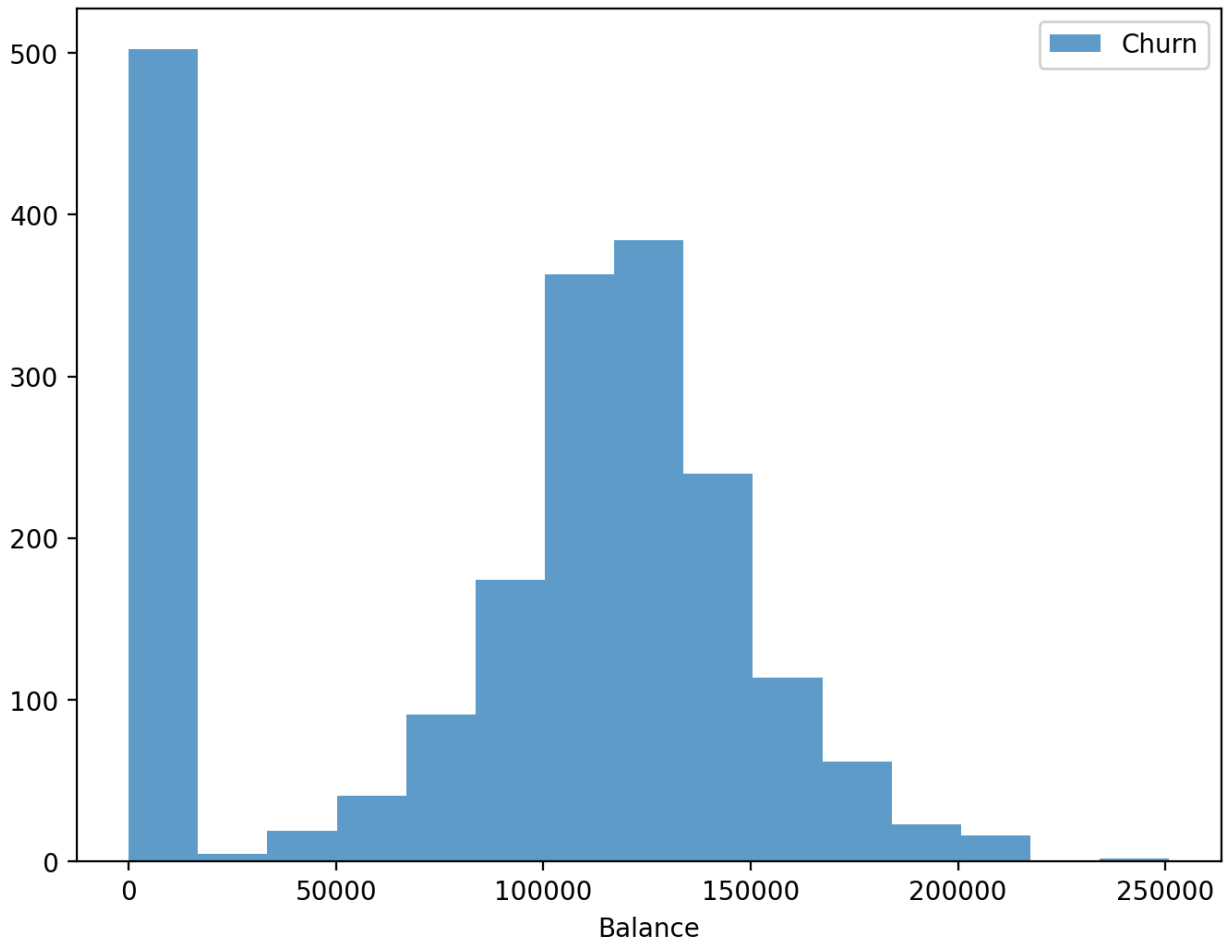
The below are the visualizations of bar graphs,box plots and histogram of both churn and not churn groups for credit score, age, balance and estimated salary.It also includes the correlation matrix,AUC-ROC curve and final bar graphs for feature importance.

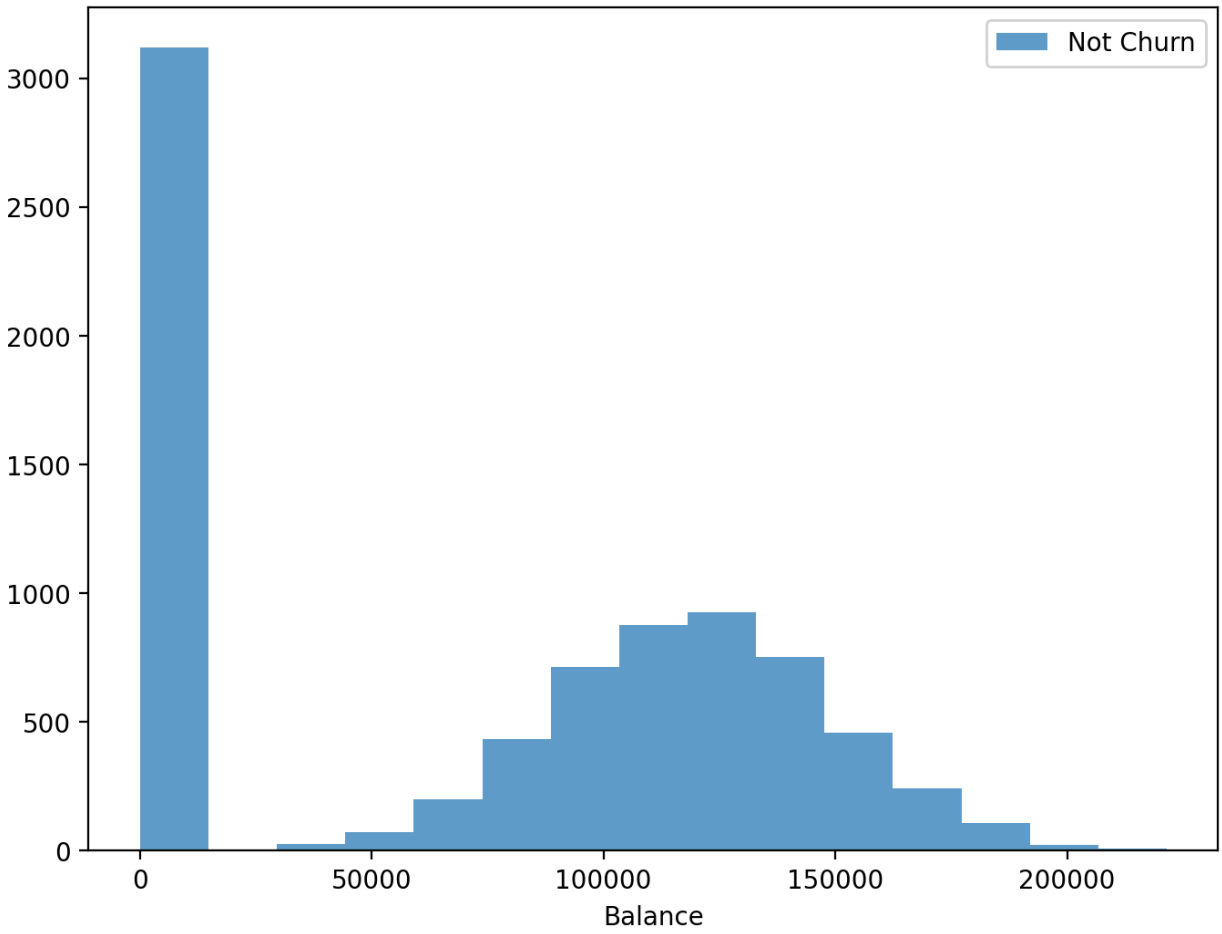

Estimated Salary Distribution by Exit Status
Estimated Salary
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
Exited

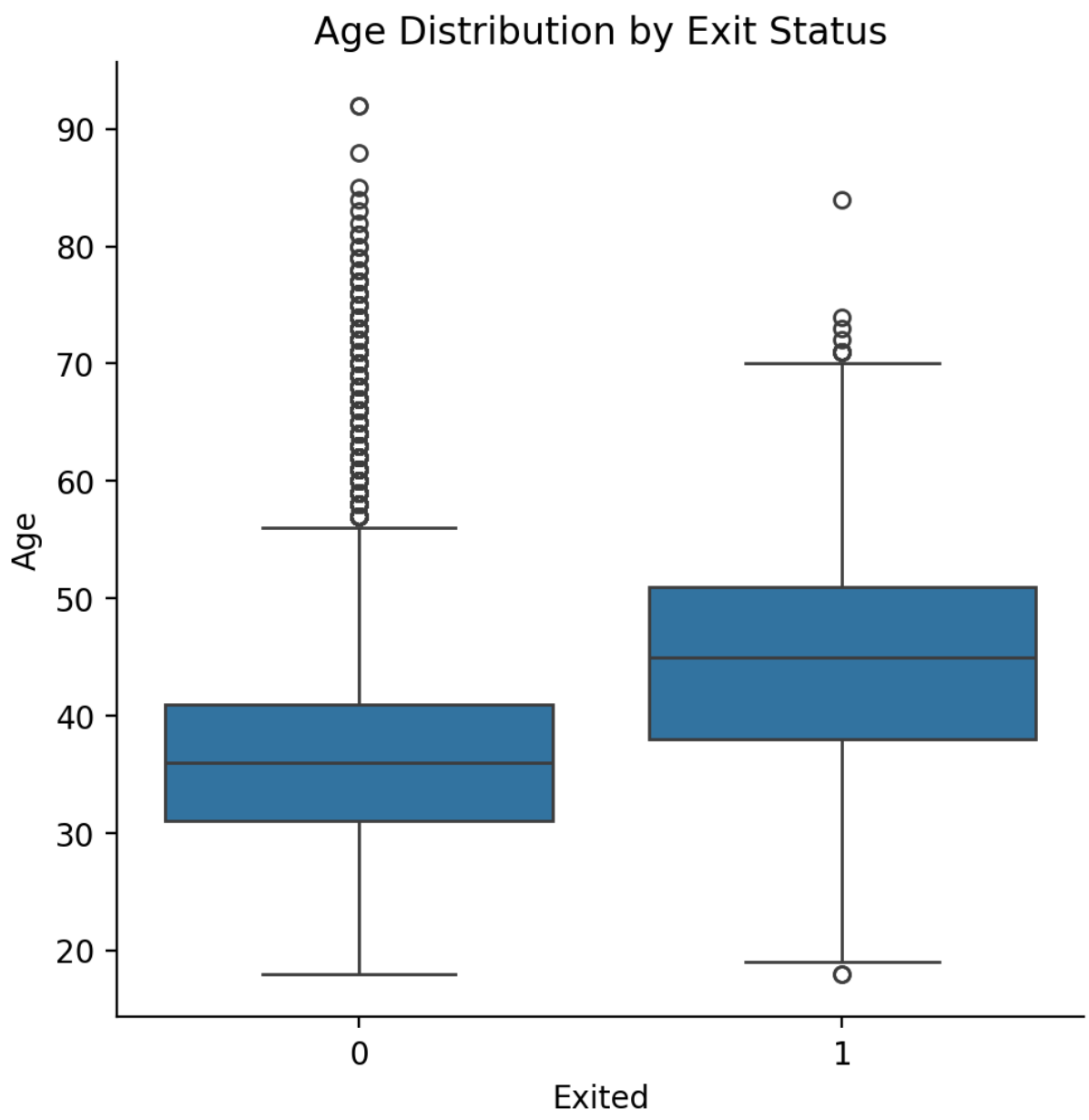

Churn
140
120
100
80
60
40
20
0
0
25000
50000
75000
100000
125000
150000
175000
200000
EstimatedSalary

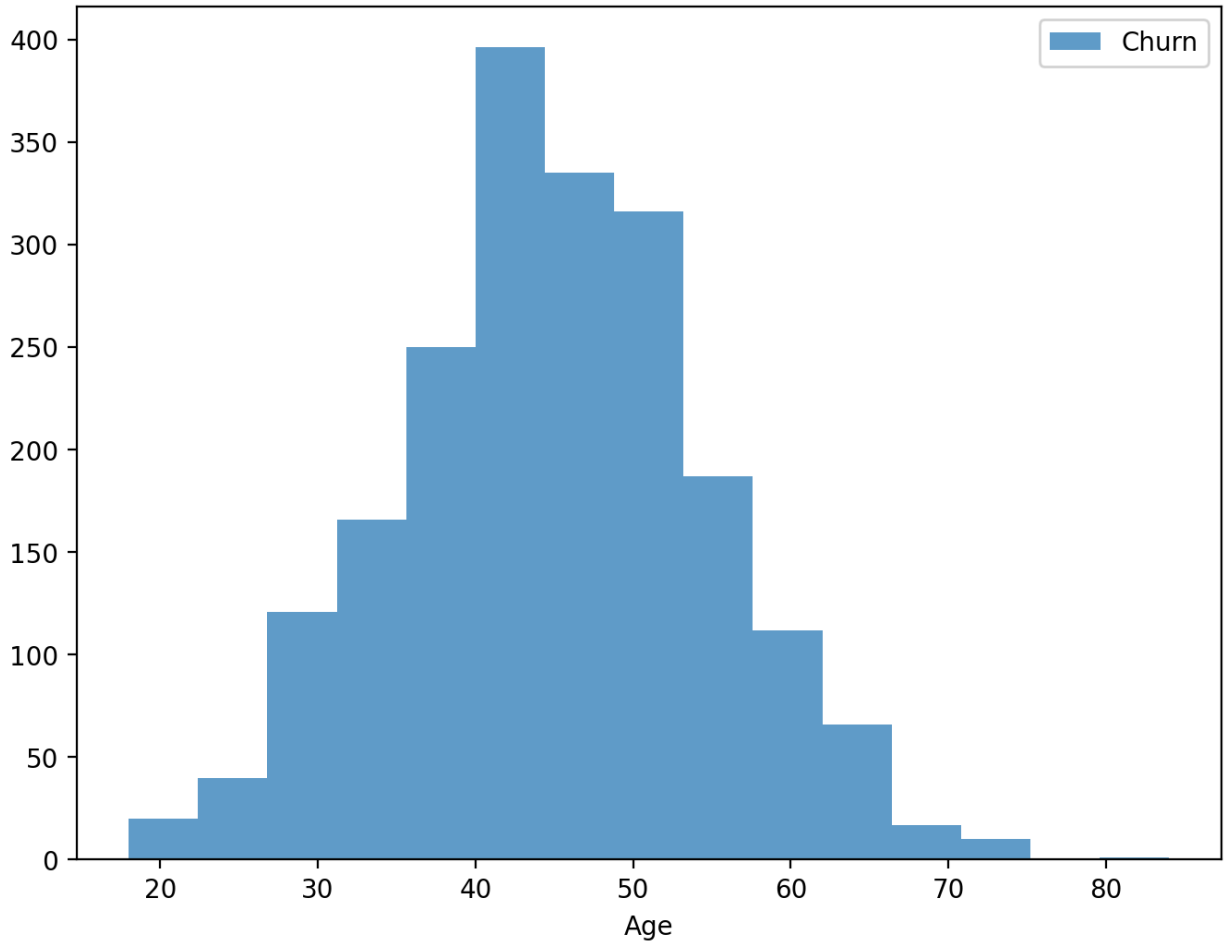

Not Churn
500
400
300
200
100
0
0
25000
50000
75000
100000
125000
150000
175000
200000
EstimatedSalary

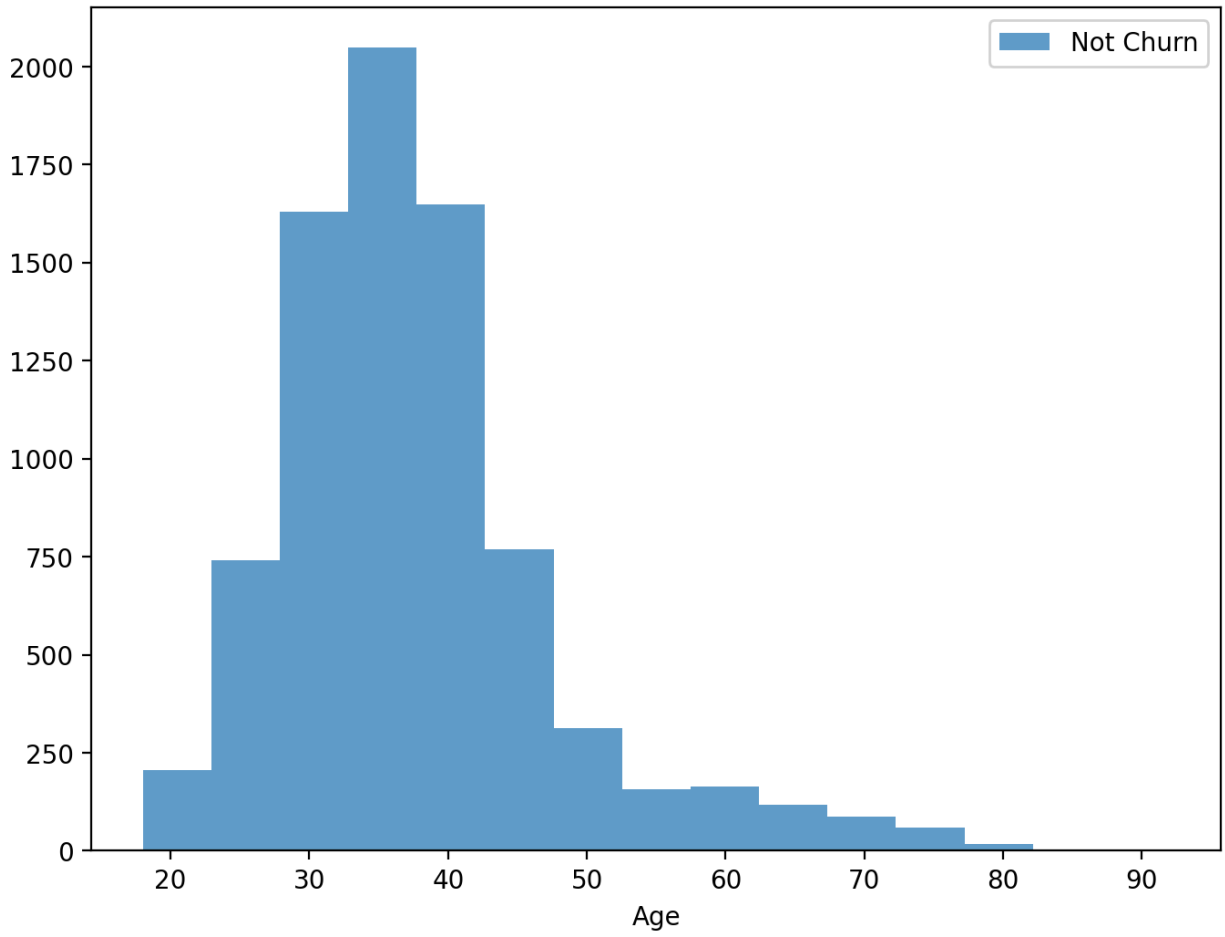

Balance Distribution by Exit Status
250000
200000
150000
100000
50000
0
Balance
0
1
Exited

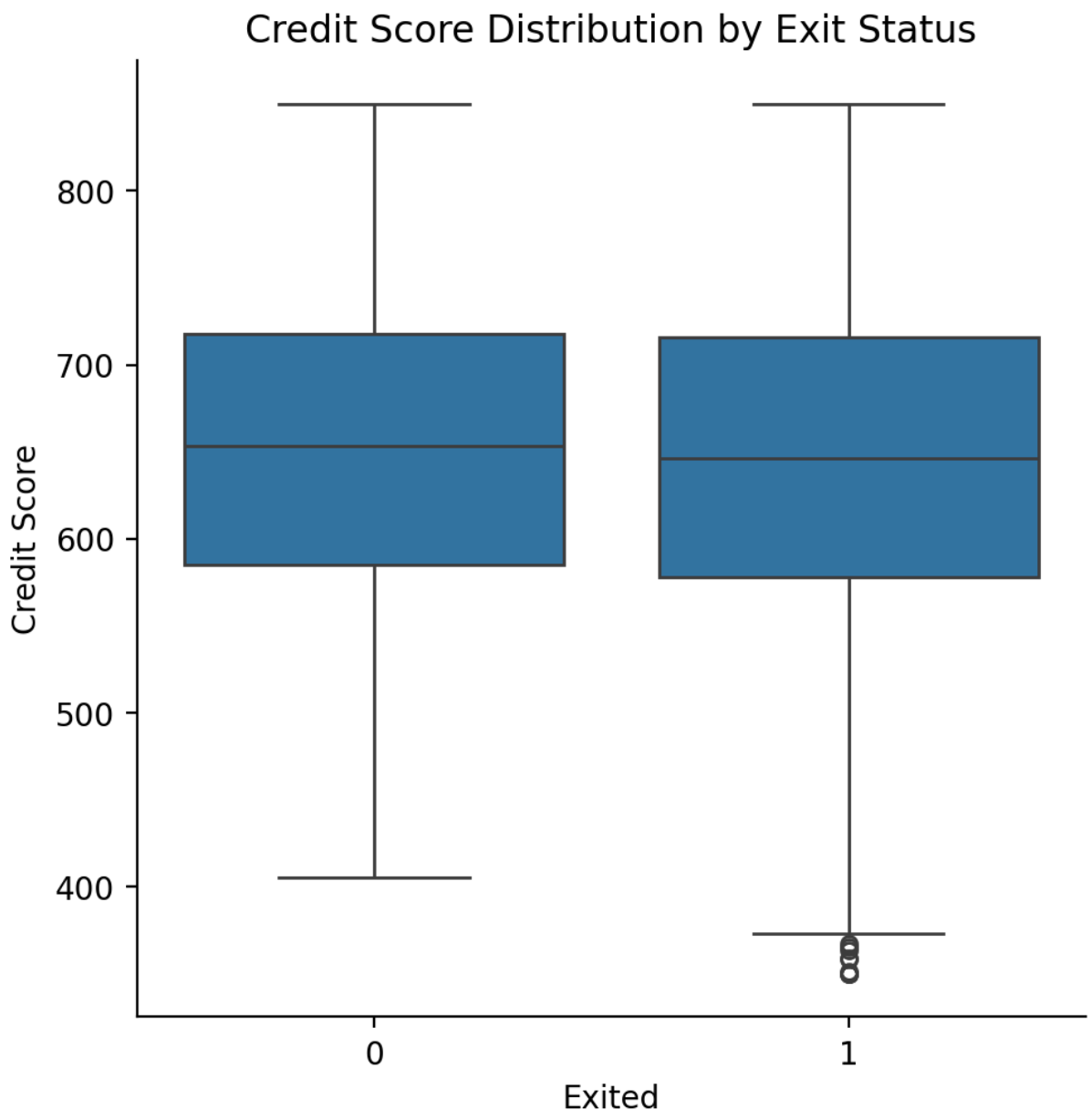

Churn
500
400
300
200
100
0
0
50000
100000
150000
200000
250000
Balance

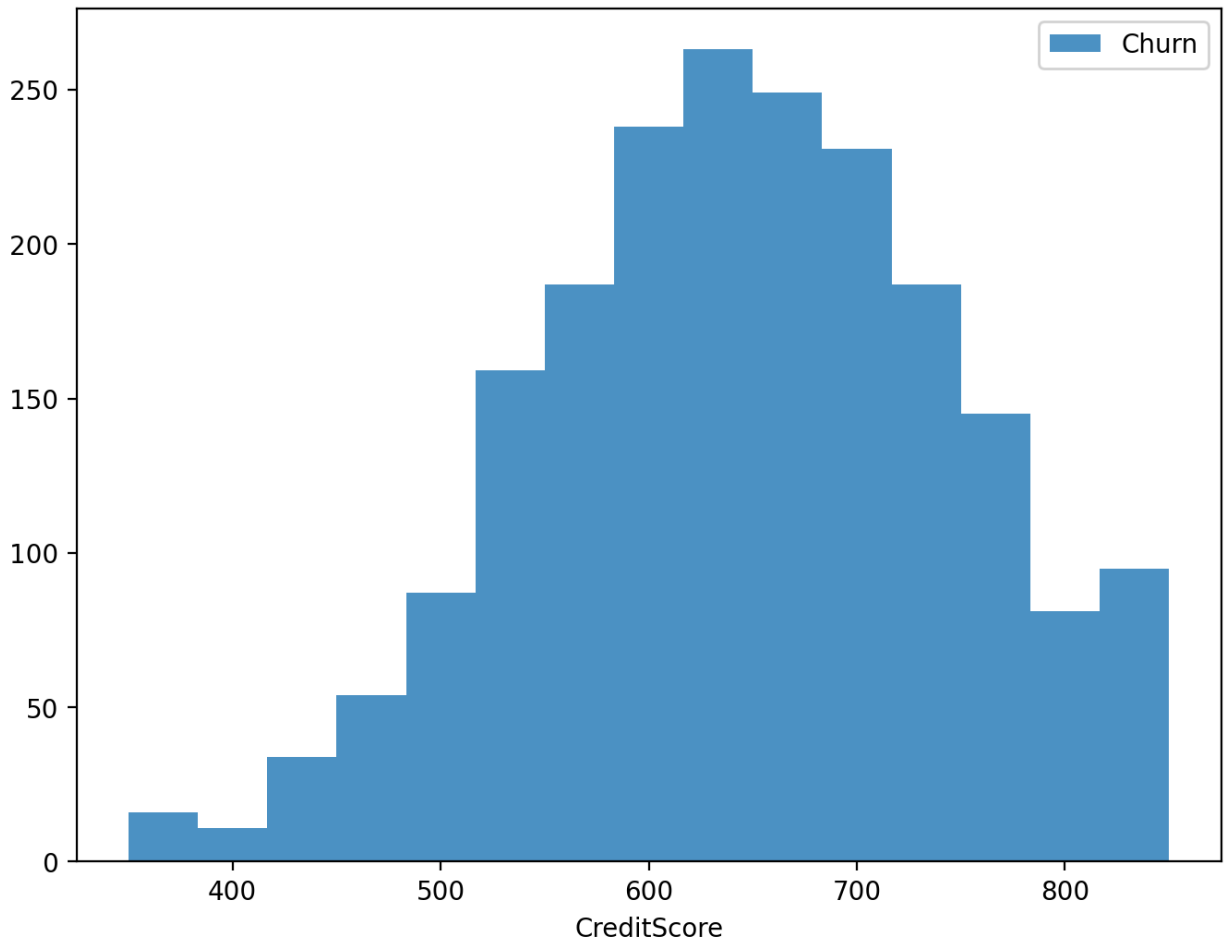

Not Churn
3000
2500
2000
1500
1000
500
0
0
50000
100000
150000
200000
Balance

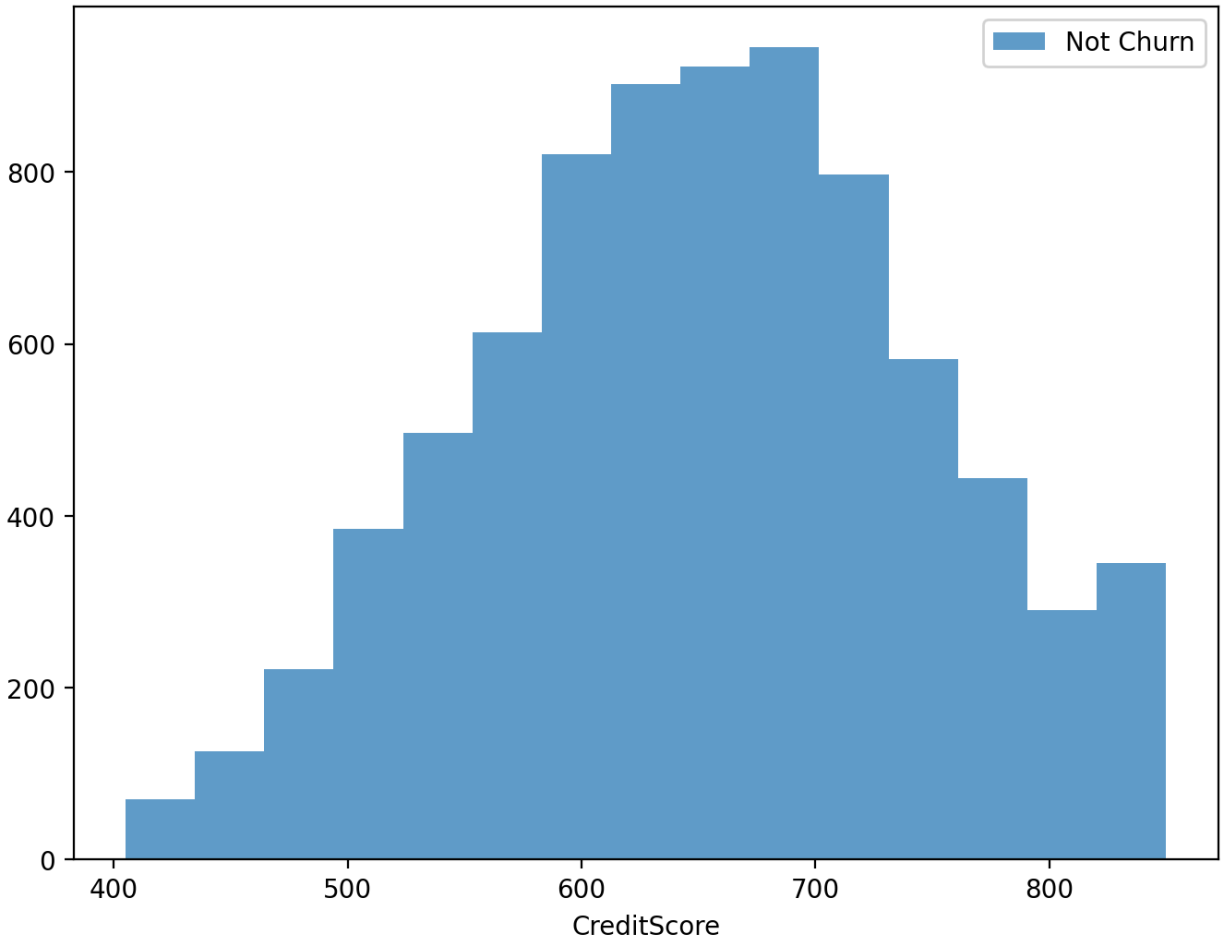

Age Distribution by Exit Status
Age
90
80
70
60
50
40
30
20
0
1
Exited

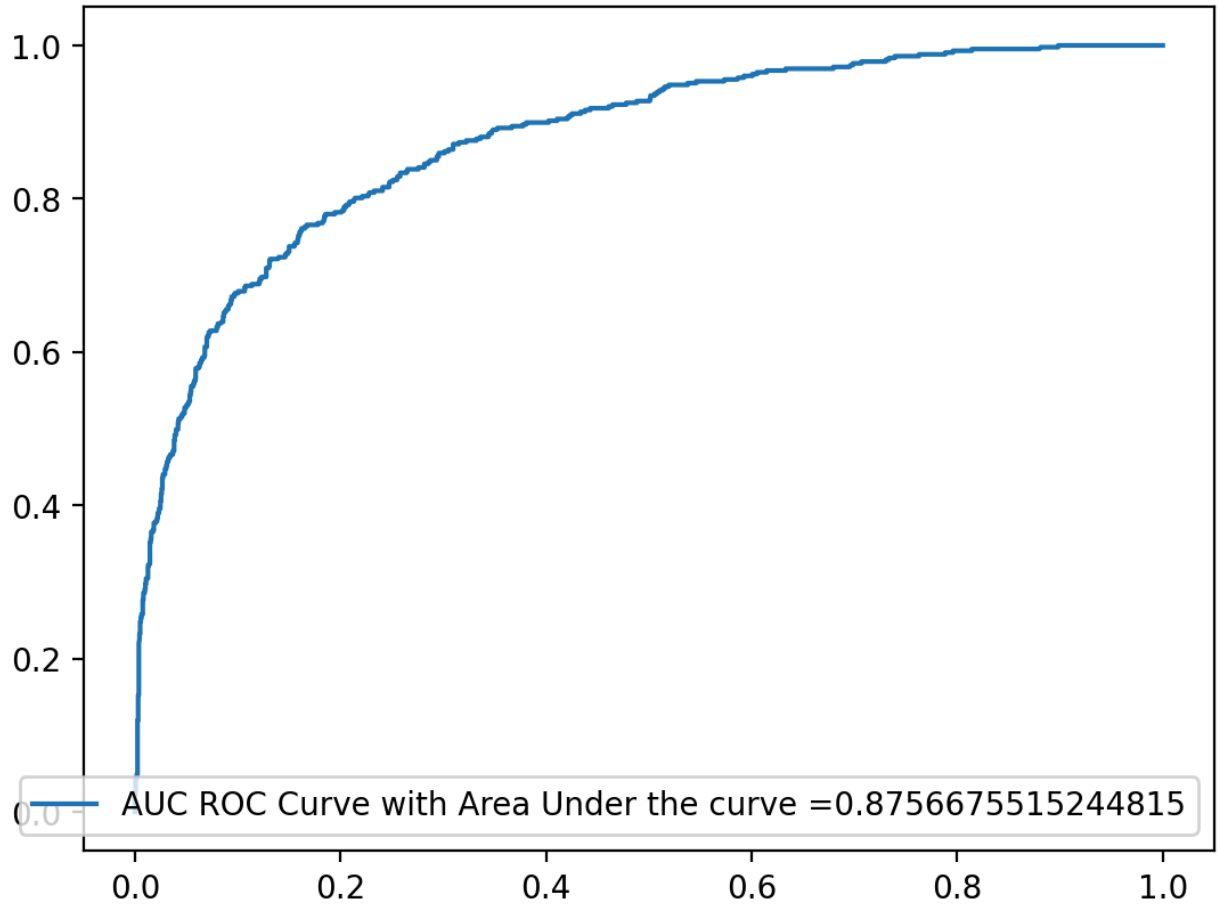

Churn
400
350
300
250
200
150
100
50
0
20
30
40
50
60
70
80
Age

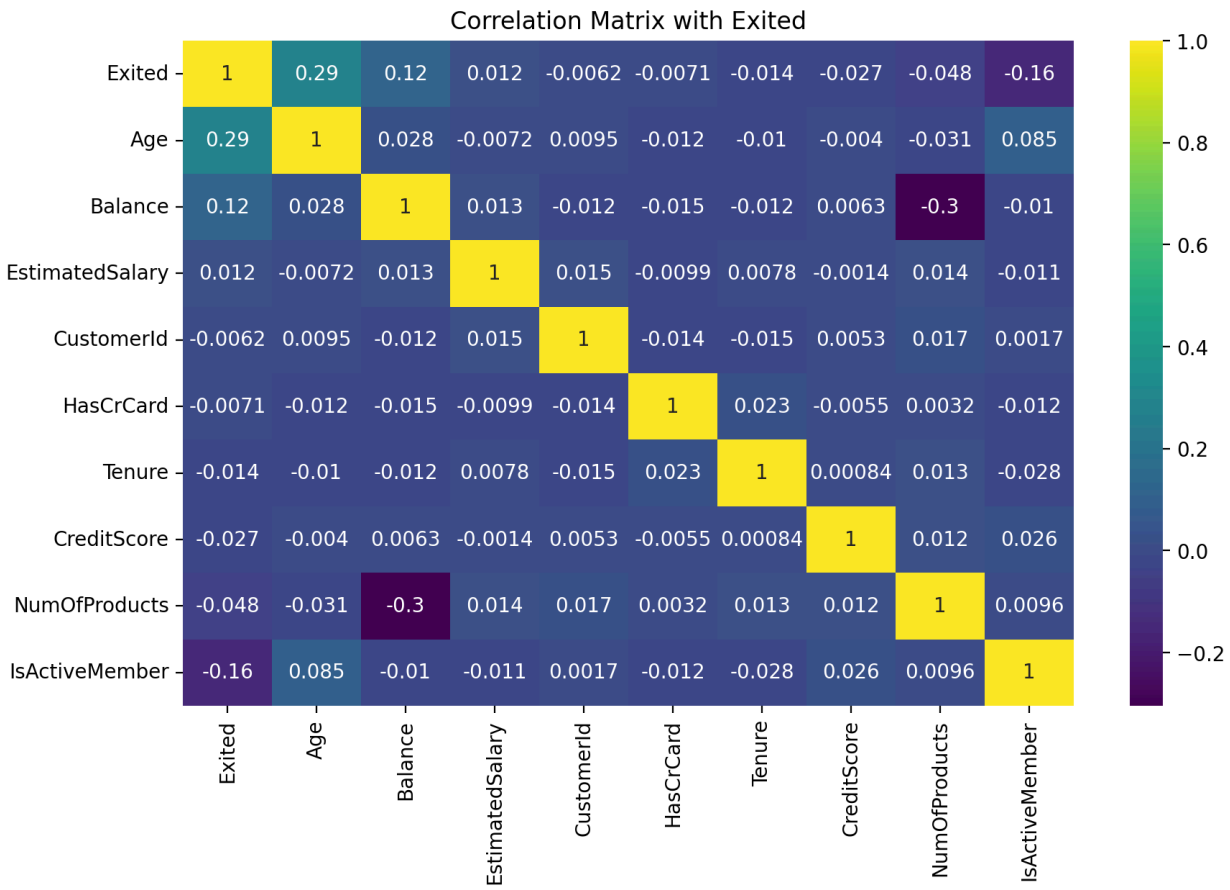

Not Churn
2000
1750
1500
1250
1000
750
500
250
0
20
30
40
50
60
70
80
90
Age

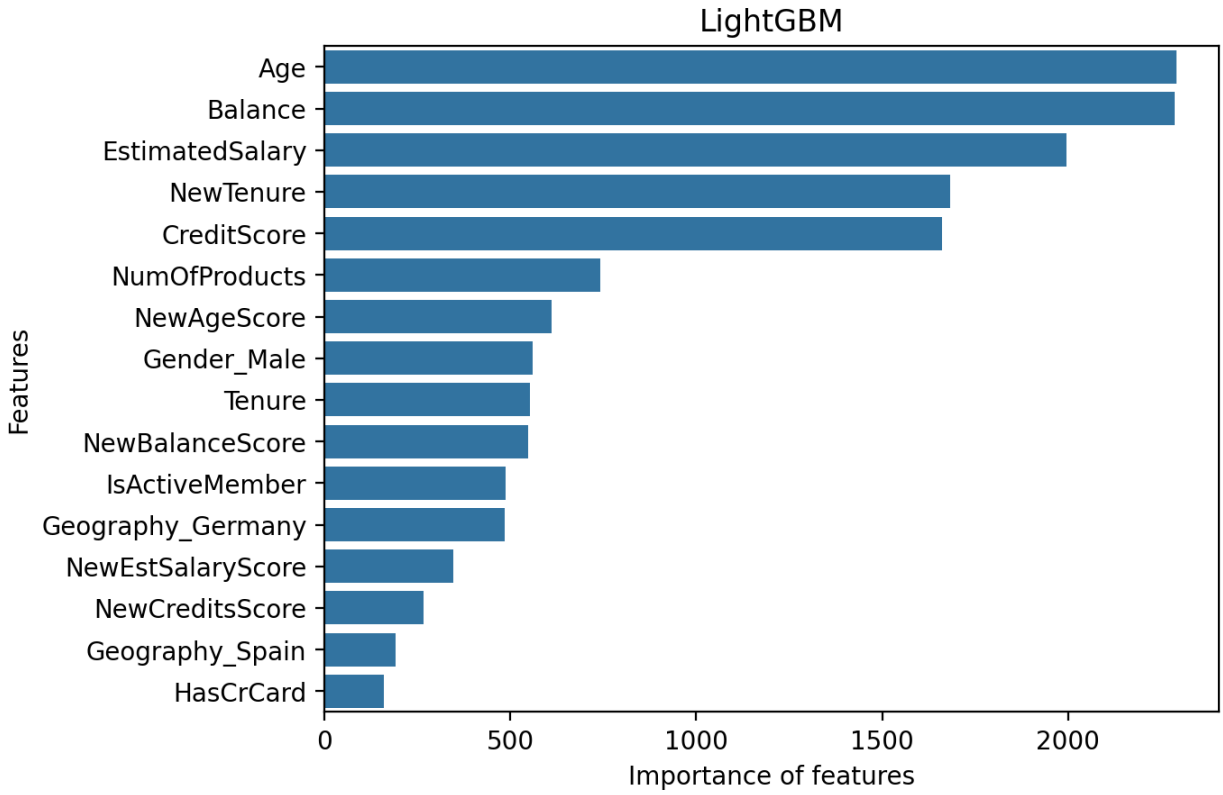

Credit Score Distribution by Exit Status
Credit Score
800
700
600
500
400
0
1
Exited

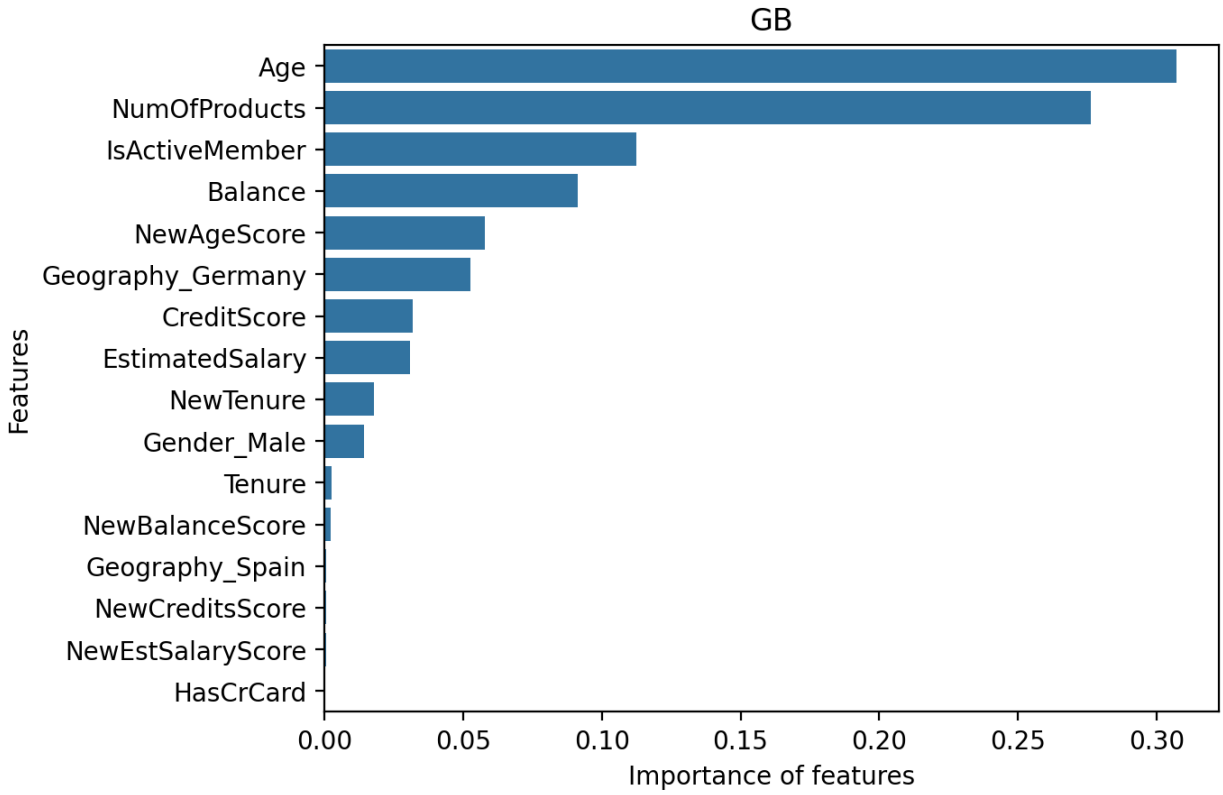

Churn
250
200
150
100
50
0
400
500
600
700
800
CreditScore


Not Churn
800
600
400
200
0
400
500
600
700
800
CreditScore


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
AUC ROC Curve with Area Under the curve =0.8756675515244815


Correlation Matrix with Exited
Exited 1 0.29 0.12 0.012 -0.0062 -0.0071 -0.014 -0.027 -0.048 -0.16
Age 0.29 1 0.028 -0.0072 0.0095 -0.012 -0.01 -0.004 -0.031 0.085
Balance 0.12 0.028 1 0.013 -0.012 -0.015 -0.012 0.0063 -0.3 -0.01
EstimatedSalary 0.012 -0.0072 0.013 1 0.015 -0.0099 0.0078 -0.0014 0.014 -0.011
CustomerId -0.0062 0.0095 -0.012 0.015 1 -0.014 -0.015 0.0053 0.017 0.0017
HasCrCard -0.0071 -0.012 -0.015 -0.0099 -0.014 1 0.023 -0.0055 0.0032 -0.012
Tenure -0.014 -0.01 -0.012 0.0078 -0.015 0.023 1 0.00084 0.013 -0.028
CreditScore -0.027 -0.004 0.0063 -0.0014 0.0053 -0.0055 0.00084 1 0.012 0.026
NumOfProducts -0.048 -0.031 -0.3 0.014 0.017 0.0032 0.013 0.012 1 0.0096
IsActiveMember -0.16 0.085 -0.01 -0.011 0.0017 -0.012 -0.028 0.026 0.0096 1
Exited
Age
Balance
EstimatedSalary
CustomerId
HasCrCard
Tenure
CreditScore
NumOfProducts
IsActiveMember
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


LightGBM
Age
Balance
EstimatedSalary
NewTenure
CreditScore
NumOfProducts
NewAgeScore
Gender_Male
Tenure
NewBalanceScore
IsActiveMember
Geography_Germany
NewEstSalaryScore
NewCreditsScore
Geography_Spain
HasCrCard
Features
0
500
1000
1500
2000
Importance of features


GB
Age
NumOfProducts
IsActiveMember
Balance
NewAgeScore
Geography_Germany
CreditScore
EstimatedSalary
NewTenure
Gender_Male
Tenure
NewBalanceScore
Geography_Spain
NewCreditsScore
NewEstSalaryScore
HasCrCard
Features
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Importance of features

Discussion

(a) Overall Results

The **LightGBM** model outperformed all other models in terms of accuracy and AUC-ROC. It captured complex patterns in the data effectively, thanks to its leaf-wise growth and boosting mechanism.

(b) Overfitting and Underfitting Issues

The **Decision Tree** model showed signs of overfitting, particularly when deeper trees were used. In contrast, **Random Forest** and **LightGBM** mitigated overfitting by using ensemble learning methods.

(c) Hyperparameter Tuning

GridSearchCV was used to tune the hyperparameters for each model, particularly for the **GBM** and **LightGBM** models. Key parameters such as `learning_rate`, `max_depth`, and `n_estimators` were optimized.

(d) Model Comparison and Selection

LightGBM was the best model, balancing high accuracy and minimal overfitting. **Gradient Boosting** was a close second, followed by **Random Forest**, which also performed well but was computationally more intensive.

Learning Outcomes

(a) Skills and Tools Applied

- **Skills**: Data preprocessing, model selection, feature engineering, hyperparameter tuning, performance evaluation.
- **Tools**: Pandas, Numpy, Scikit-learn, LightGBM, Matplotlib, Seaborn.

(b) Dataset Used

The dataset contained 10,000 records of bank customers, with 12 features and the target variable "Exited" (churn or not).

(c) What I Learned

I gained a deep understanding of customer churn prediction using machine learning. I learned how to handle imbalanced datasets, apply feature engineering, and optimize model performance through hyperparameter tuning.

Conclusion

The project successfully implemented multiple machine learning models to predict customer churn, with **LightGBM** emerging as the best model. The feature importance analysis provided valuable insights into customer behavior, highlighting factors like **CreditScore** and **Balance**. These insights can help banks take targeted action to retain at-risk customers.

(a) Did I Accomplish the T,P,E?

Yes, I accomplished the **Task** of predicting churn, solved the **Problem** of identifying at-risk customers, and achieved the **Expected Outcome** of building an accurate predictive model.

(b) Advantages and Limitations

- **Advantages**: High predictive accuracy and interpretability with LightGBM, efficient handling of large datasets.
- **Limitations**: Limited feature set—additional behavioral data could further improve predictions.