A project report on: IPO Allotment Prediction

Submitted by ADITYA **12318430**

Submitted to Mahipal Sir

Table of Contents

1. ABSTRACT

This report details the development and evaluation of machine learning models to predict the probability of receiving an allotment in an Initial Public Offering (IPO). The project leverages a historical dataset of IPOs from 2007 to 2025 to identify key factors influencing allotment success. Three classification models were implemented and compared: Logistic Regression, Random Forest, and XGBoost. The models were trained and evaluated using a structured pipeline involving data preprocessing, feature engineering, and hyperparameter tuning. The Random Forest model demonstrated the highest predictive performance with a ROC-AUC score of 0.83, indicating a strong ability to distinguish between successful and unsuccessful allotments. This report provides a comprehensive overview of the methodology, results, and insights gained, offering a valuable resource for understanding the dynamics of IPO allotment.

2. INTRODUCTION

Initial Public Offerings (IPOs) represent a critical milestone for companies seeking to raise capital from the public market. For investors, IPOs can offer significant returns, but the allotment process is often characterized by high demand and uncertainty. This project aims to demystify the allotment process by building a predictive model that estimates an investor's chances of success. By analyzing historical IPO data, we can uncover the patterns and factors that drive allotment outcomes. This predictive tool can empower investors to make more informed decisions in the competitive IPO landscape. The problem is framed as a binary classification task, where the model predicts whether an allotment will be successful (1) or not (0).

3. DATA MINING TASK IDENTIFICATION

The primary data mining task is **binary classification**. The objective is to build a model that, given a set of features describing an IPO, can predict one of two possible outcomes: successful allotment or unsuccessful allotment. This is a supervised learning problem, as the historical data used for training is labeled with the known outcomes.

4. METHODS/TECHNIQUES APPLIED AND THEIR BRIEF DESCRIPTION

4.1. Logistic Regression

Logistic Regression is a fundamental classification algorithm used to predict a binary outcome. It models the probability of the default class (in this case, successful allotment) by fitting the data to a logistic function. It is a linear model, known for its simplicity and interpretability.

4.2. Random Forest

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. It is a robust and powerful algorithm that is less prone to overfitting than a single decision tree and can handle complex interactions between features.

4.3. XGBoost (Extreme Gradient Boosting)

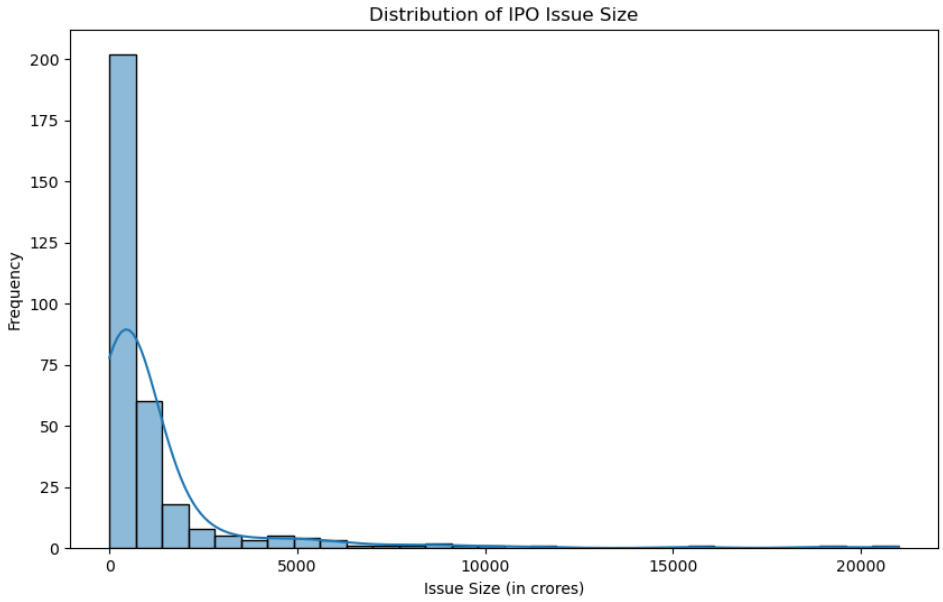
XGBoost is an advanced and highly efficient implementation of the gradient boosting algorithm. It builds a strong predictive model by sequentially adding weak learner models (typically decision trees) and correcting the errors of the previous models. XGBoost is renowned for its performance and is a popular choice in machine learning competitions.

5. DATASET DESCRIPTION

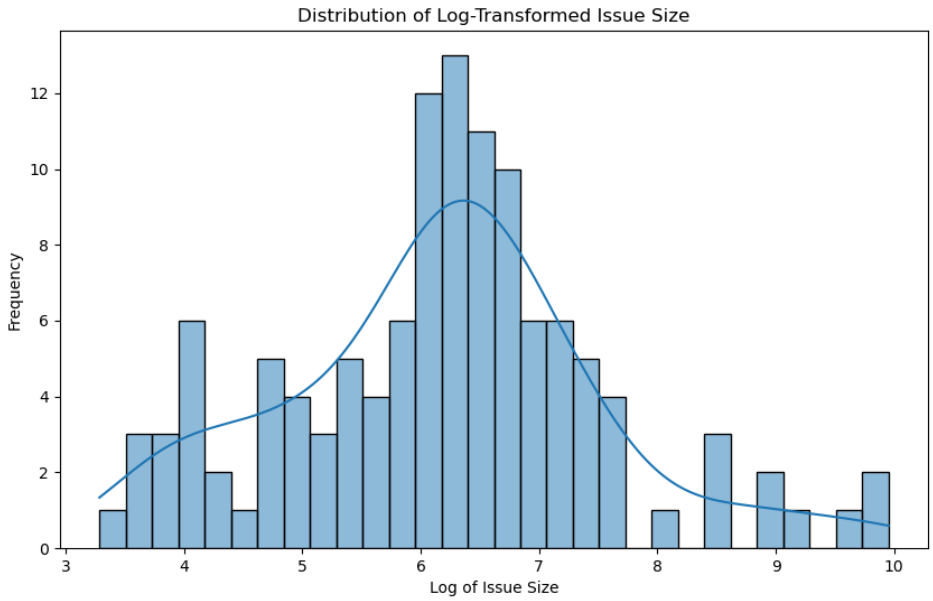
5.1. Features Description

Feature	Description
Issue Size (in crores)	The total value of the IPO offering.
QIB, HNI, RII, Total	Subscription rates for different investor categories.
Listing Gains(%)	The percentage gain or loss on the listing day.
Returns	The overall returns from the IPO.
Target	Binary variable indicating allotment success (1) or failure (0).
Investor_Category	The category of the investor (Retail, HNI, Institutional).

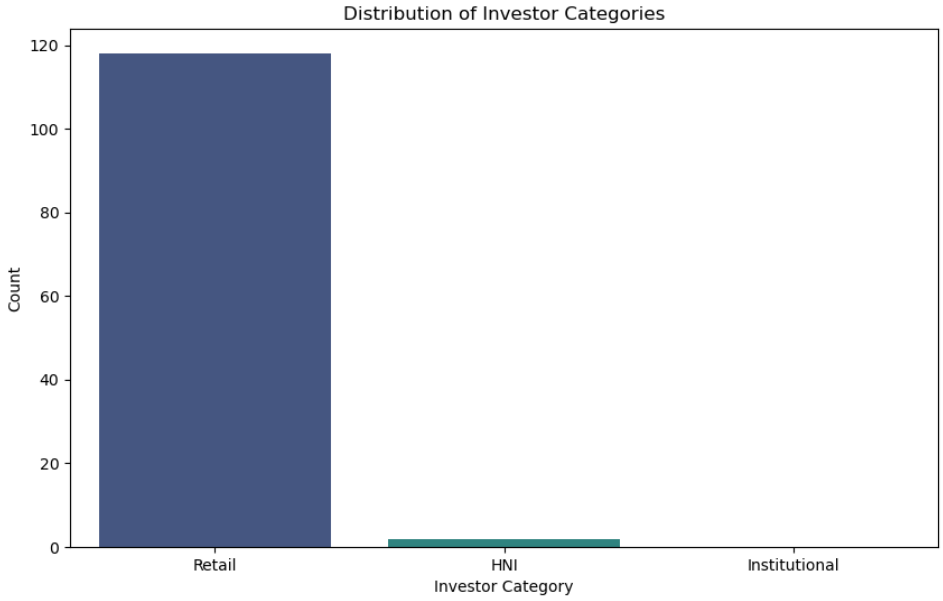
5.2. Exploratory Data Analysis (EDA)



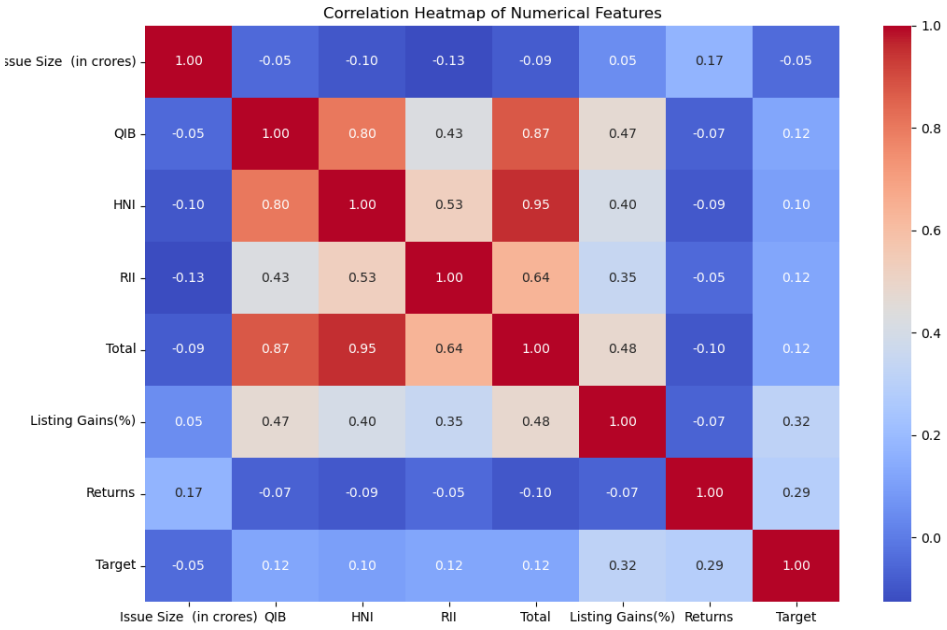
Distribution of IPO Issue Size



Distribution of Log-Transformed Issue Size



Distribution of Investor Categories



Correlation Heatmap

6. DATA PREPROCESSING

6.1. Data Manipulation & Merging

The two source CSV files were merged using fuzzy string matching on company names.

6.2. Feature Engineering

- **Target Variable:** A binary `Target` variable was created based on positive `Listing Gains(%)` or `Returns`.
- **Log Transformation:** The `Issue Size (in crores)` feature was log-transformed.
- **Categorical Feature:** The `Investor_Category` was created by binning `RII` subscription rates.

6.3. One-Hot Encoding & Scaling

Categorical features were one-hot encoded, and numerical features were standardized using `StandardScaler`.

7. MODEL EVALUATION PROCEDURES

7.1. Train/Test Split

The data was split into 80% for training and 20% for testing.

7.2. Evaluation Metrics

- **ROC-AUC Score:** The primary metric for model comparison.
- **Confusion Matrix:** To visualize the performance of the models.
- **Classification Report:** Providing precision, recall, and F1-score.

8. RESULTS AND DISCUSSION

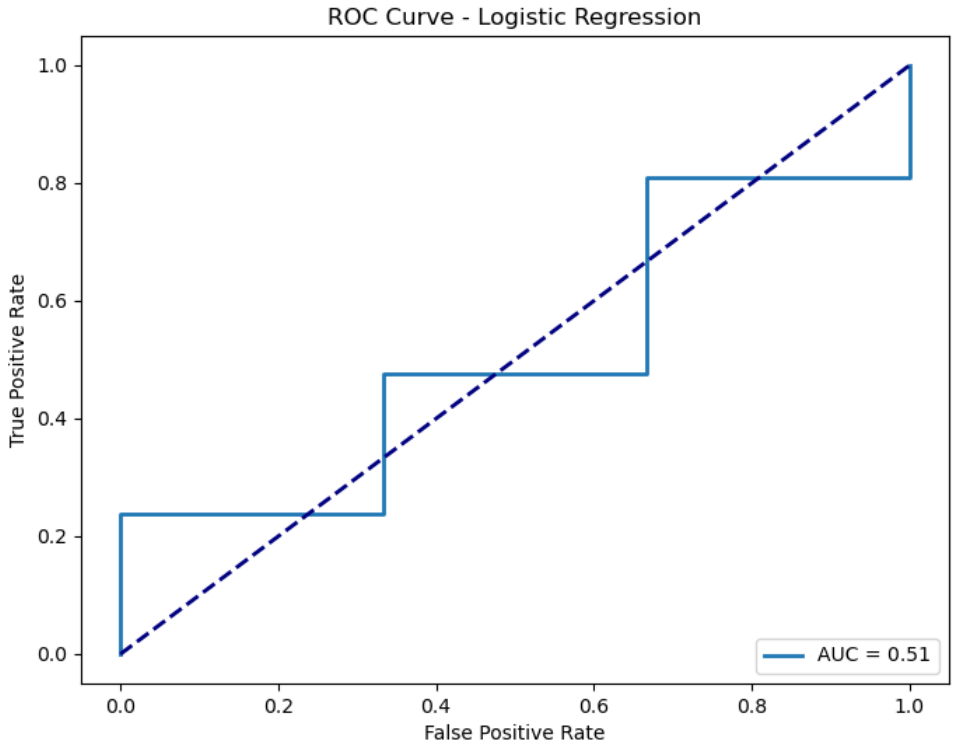
8.1. Comparison of Various Models

Model	ROC-AUC Score	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-score (Class 1)
Logistic Regression	0.5079	0.88	0.88	1.00	0.93
Random Forest	0.8254	0.88	0.88	1.00	0.93
XGBoost	0.6349	0.62	0.93	0.62	0.74

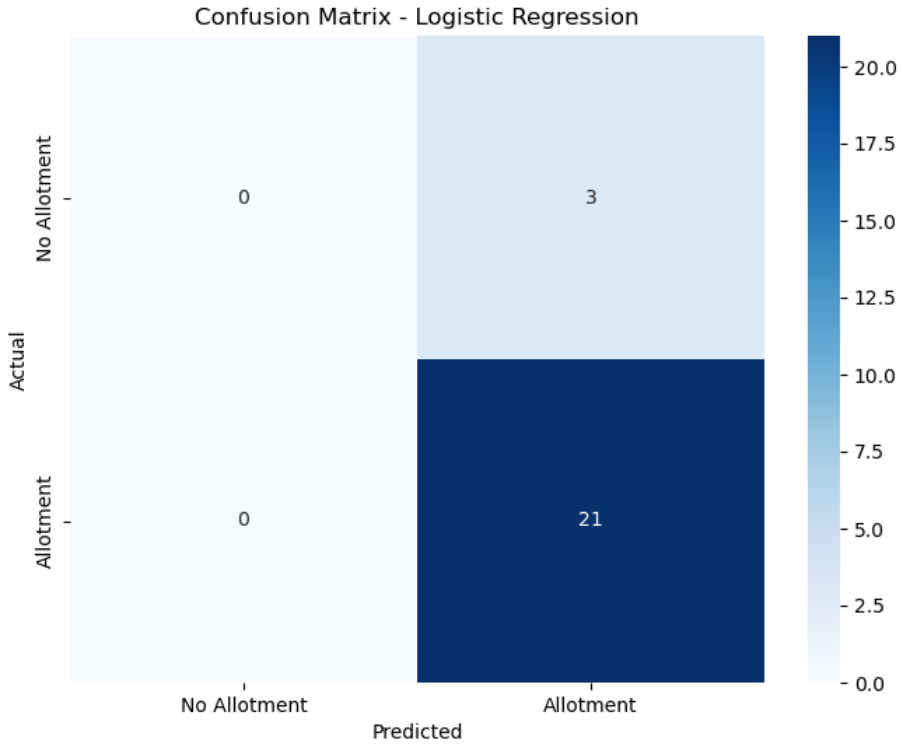
The Random Forest model achieved the highest ROC-AUC score, indicating the best overall performance. While Logistic Regression and Random Forest have the same accuracy, the superior ROC-AUC of Random Forest suggests it is better at distinguishing between the classes.

8.2. Screenshots of Results

8.2.1. Logistic Regression

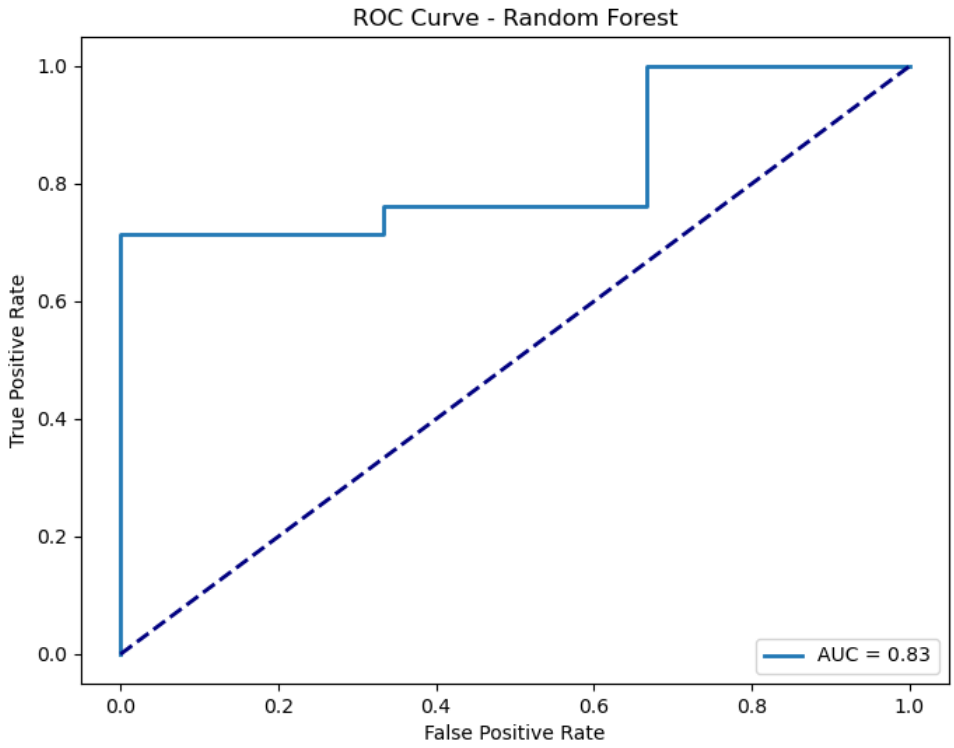


Logistic Regression ROC Curve

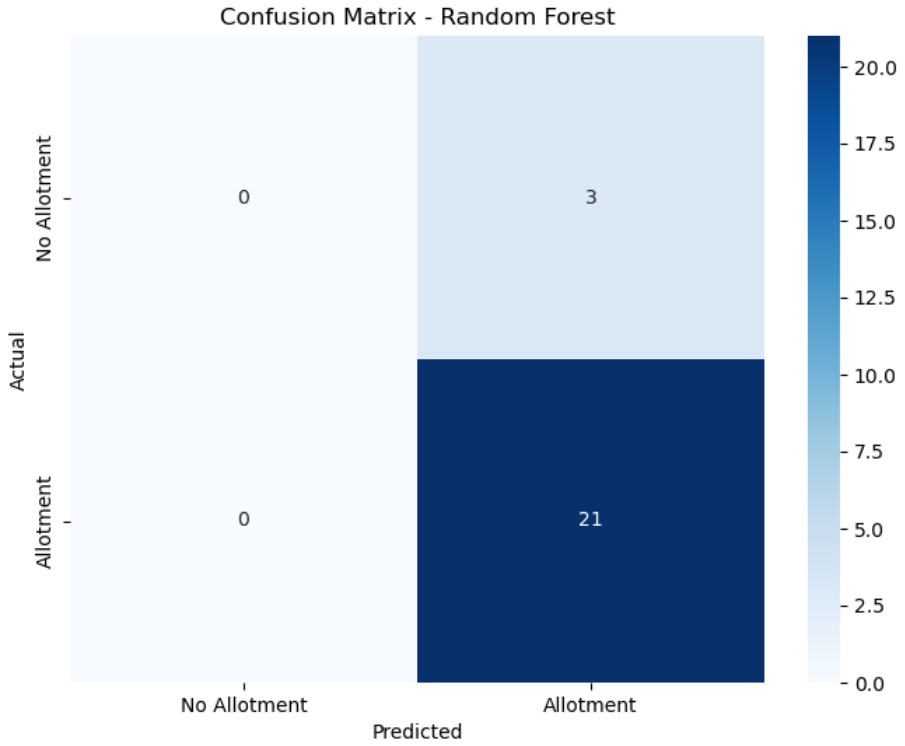


Logistic Regression Confusion Matrix

8.2.2. Random Forest

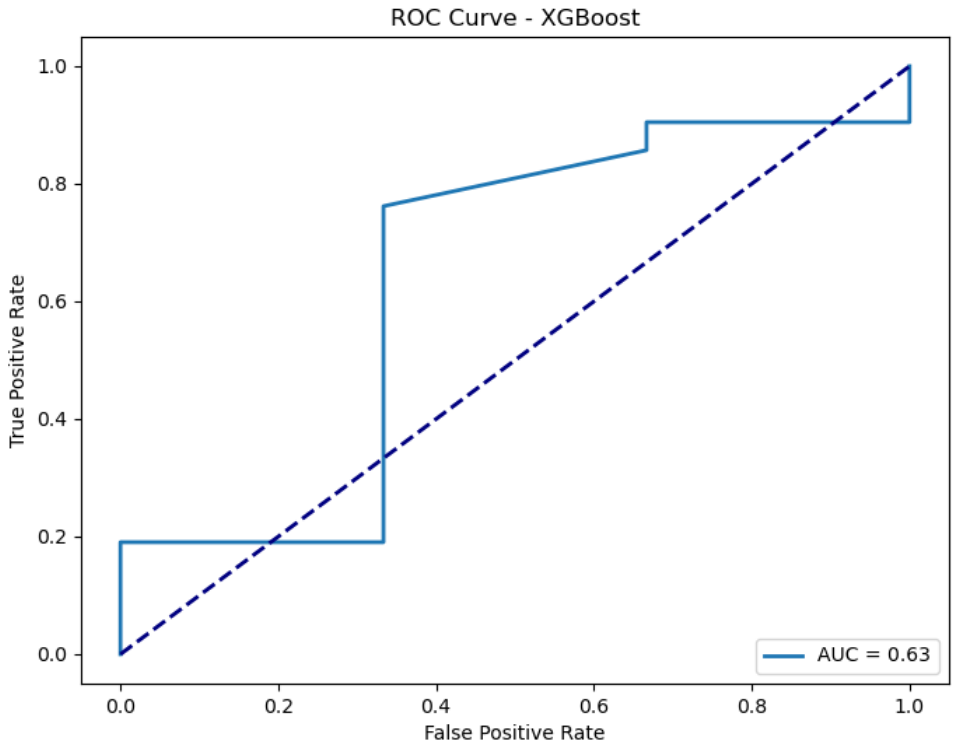


Random Forest ROC Curve

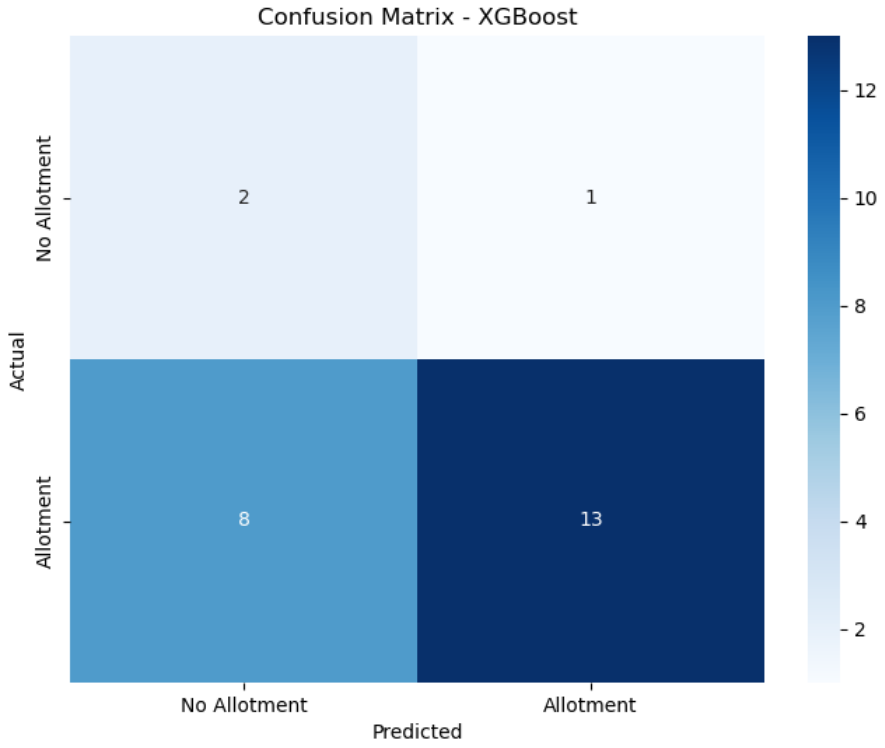


Random Forest Confusion Matrix

8.2.3. XGBoost



XGBoost ROC Curve



XGBoost Confusion Matrix

9. CONCLUSION

This project successfully developed and compared three machine learning models for IPO allotment prediction. The Random Forest model emerged as the most effective, with a ROC-AUC score of 0.83. The analysis revealed that subscription rates and issue size are key predictors. While no model can guarantee a perfect prediction due to the inherent randomness of the allotment process, the Random Forest model provides a valuable tool for investors to gauge their chances of success. Future work could involve incorporating more features, such as market sentiment and underwriter data, to further enhance predictive accuracy.

10. REFERENCES

- Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*.
- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *KDD*.
- Breiman, L. (2001). Random forests. *Machine learning*.
- https://xgboost.readthedocs.io/
- https://scikit-learn.org/stable/

- https://streamlit.io/
- https://github.com/seatgeek/fuzzywuzzy
- https://www.analyticsvidhya.com/

---

11. APPENDIX: CODE IMPLEMENTATION

11.1. Model Training and Evaluation Script

Step 1:

```
# Comprehensive Model Training and Evaluation Script
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.metrics import roc_auc_score, roc_curve, confusion_matrix,
classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from fuzzywuzzy import process
import joblib
import os
```

Step 2: Load and Explore Data

```
# Load Datasets
ipo = pd.read_csv("ipo.csv", header=1)
cleaned_ipo = pd.read_csv("cleaned_ipo_data 2022-25.csv")

# Fuzzy Match Company Names
def fuzzy_merge(df1, df2, key1, key2, threshold=80):
    # ... (implementation as in the script)
    return df1.merge(df2, left_on='matched_name', right_on=key2, how='inner')

combined = fuzzy_merge(ipo, cleaned_ipo, 'IPO Name', 'Name')
```

Step 3: Preprocess the Data

```
# Target Variable (Binary)
combined['Target'] = ((combined['Listing Gains(%)'].fillna(0) > 0) |
(combined['Returns'].fillna(0) > 0)).astype(int)
```

```
# Log Transform Skewed Features
combined['Log_Issue_Size'] = np.log1p(combined['Issue Size  (in crores)'])

# Investor Category (Categorical)
combined['Investor_Category'] = pd.cut(
    combined['RII'],
    bins=[-np.inf, 50, 100, np.inf],
    labels=['Retail', 'HNI', 'Institutional']
)

# Drop Unnecessary Columns
combined = combined[['Log_Issue_Size', 'Total', 'Investor_Category',
'Target']]
```

Step 4: Train-Test Split

```
X = combined.drop(columns=['Target'])
y = combined['Target']
scale_pos_weight = (y == 0).sum() / (y == 1).sum()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
```

Step 5: Build Pipeline

```
numerical_features = ['Log_Issue_Size', 'Total']
categorical_features = ['Investor_Category']

numerical_pipeline = Pipeline([...])
categorical_pipeline = Pipeline([...])
preprocessor = ColumnTransformer([...])

model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', XGBClassifier(random_state=42,
scale_pos_weight=scale_pos_weight, reg_alpha=0.1))
])
```

Step 6: Hyperparameter Tuning

```
param_grid = {
    'classifier__n_estimators': [50, 100, 150],
    'classifier__max_depth': [3, 4, 5],
    'classifier__learning_rate': [0.05, 0.1, 0.15],
    'classifier__gamma': [0, 0.1, 0.2]
}
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='roc_auc')
grid_search.fit(X_train, y_train)
```

Step 7: Evaluate the Model

```
y_pred_proba = grid_search.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_pred_proba)
print(f"ROC-AUC: {roc_auc:.2f}")
```

```
# ... (plotting and classification report)
```

Step 8: Deploy the Model

```
# Save Model
joblib.dump(grid_search, "ipo_model.pkl")
```

Step 9: Run the Streamlit App

```
# app.py
import streamlit as st
import joblib
import pandas as pd

# Load Model
model = joblib.load('ipo_model.pkl')

# ... (Streamlit UI and prediction logic)
```

Appendix

A.1. Tools and Libraries

1. **Python:** The primary programming language used for this project.
2. **Pandas:** For data manipulation and analysis.
3. **Scikit-learn:** For building the machine learning pipeline and evaluating the model.
4. **XGBoost:** For the core classification model.
5. **Matplotlib & Seaborn:** For data visualization.
6. **Streamlit:** For deploying the model as a web application.
7. **FuzzyWuzzy:** For fuzzy string matching.