EMPLOYEE RETENTION REPORT

System Study / Domain Analysis:

Business Processes Description and Analytics Application:

In the current landscape of human resources management, retaining skilled employees is as important as acquiring them. The dataset used is from an organization having a 15% attrition rate with over 4,000 employees. The high attrition rate has created the need to assess the strategies of workforce management. It has variables associated with job satisfaction, environment satisfaction, work-life balance, salary, department, and whether or not an employee leaves an organization (attrition). The extra data includes the timestamp of employee arrival and departure.

Business Process:

The dataset is used within the organization by the HR department for identifying the reasons behind employee attrition and mitigating the factors to reduce the rate of employee turnover. Besides, employee attrition hampers the workflow, increasing the cost related to recruitment and training of new employees. One has to understand the dynamics at play in each department, in different roles, and across demographic segments in order to come up with effective retention strategies.

Type of Analytics

- **Descriptive Analytics:** Understanding the basic features of the dataset and the distribution of key variables with respect to employee satisfaction and turnover.
- **Diagnostic Analytics:** To establish the causes of certain patterns, such as high attrition rates in departments or particular age groups.
- **Predictive Analytics:** The facility of logistic regression and decision tree models to predict likely attrition based on a wide variety of attributes and behaviors of employees allows predictive analytics to identify future trends and prepare preventive measures.
- **Prescriptive Analytics:** To prescribe actions based on the insights and forecasts of predictive analytics.

Utility of Analysis:

- **Policy Formulation:** The analysis provides insight into policy formulation aimed at bettering job satisfaction, satisfaction with the environment, and work-life balance, directly affecting the factors of attrition.
- **Strategic Interventions:** Analysis clearly identifies the 'at high risk' groups in the organization, and therefore, HR can make strategic interventions in these critical groups to reduce turnover.
- **Optimized Training and Development:** The organization can learn the correlation of given training and retention, optimizing the investments into training that touch only high-impact areas to improve employees' satisfaction and loyalty.
- **Benefits and Compensation:** Analytical insights can support developing competitive yet sustainable compensation packages, which are much closer to matching industry standards and employee expectations.
- **Strategic Planning:** This amalgamation of analytical processes would further support the strategic decision-making process with the data foundation of HR policies and practices and can help in reducing turnover.

Exploratory Data Analysis:

Once we loaded the dataset, we perform the Exploratory Data Analysis (EDA) in order to help us understand the details regarding the data. By examining the column names, data types, and the data information, we are able to gauge a better understanding regarding the characteristics of the variables, and how to move towards the next steps in out preprocessing.

This was used in creating two new variables in trying to predict better about attrition using the time in and time out sheets. The first one was the number of absences, where it counted missed time stamps for a certain day as an absent, whose total is summed up for the year of an employee, as increasing numbers of absences potentially signifying disinterest of the employee and possible attrition. The second variable was the average duration of time spent in the office by each employee, the difference between time out and time in summed up for every day, with the duration averaged for the year so as to find out if less average time spent in the office could point to the fact that the employee is likely to leave.

In [116]:
```
data.head(10)
```

Out[116]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeID	Gender	JobLevel	...	YearsAtCompany	Year
0	51.0	No	Travel_Rarely	Sales	6.0	2.0	Life Sciences	1.0	Female	1.0	...	1.0	
1	31.0	Yes	Travel_Frequently	Research & Development	10.0	1.0	Life Sciences	2.0	Female	1.0	...	5.0	
2	32.0	No	Travel_Frequently	Research & Development	17.0	4.0	Other	3.0	Male	4.0	...	5.0	
3	38.0	No	Non-Travel	Research & Development	2.0	5.0	Life Sciences	4.0	Male	3.0	...	8.0	
4	32.0	No	Travel_Rarely	Research & Development	10.0	1.0	Medical	5.0	Male	1.0	...	6.0	
5	46.0	No	Travel_Rarely	Research & Development	8.0	3.0	Life Sciences	6.0	Female	4.0	...	7.0	
6	28.0	Yes	Travel_Rarely	Research & Development	11.0	2.0	Medical	7.0	Male	2.0	...	0.0	
7	29.0	No	Travel_Rarely	Research & Development	18.0	3.0	Life Sciences	8.0	Male	2.0	...	0.0	
8	31.0	No	Travel_Rarely	Research & Development	1.0	3.0	Life Sciences	9.0	Male	3.0	...	9.0	
9	25.0	No	Non-Travel	Research & Development	7.0	4.0	Medical	10.0	Female	4.0	...	6.0	

In [118]:
```
data.columns
```

Out[118]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',
       'Education', 'EducationField', 'EmployeeID', 'Gender', 'JobLevel',
       'JobRole', 'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',
       'PercentSalaryHike', 'StockOptionLevel', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion',
       'YearsWithCurrManager', 'EnvironmentSatisfaction', 'JobSatisfaction',
       'WorkLifeBalance', 'JobInvolvement', 'PerformanceRating',
       'Avg Time Spent(Mins)', 'Absences'],
      dtype='object')
```

```
In [124]: data.dtypes

Out[124]: Age                        float64
          Attrition                   object
          BusinessTravel              object
          Department                  object
          DistanceFromHome           float64
          Education                  float64
          EducationField              object
          EmployeeID                 float64
          Gender                      object
          JobLevel                   float64
          JobRole                     object
          MaritalStatus               object
          MonthlyIncome              float64
          NumCompaniesWorked         float64
          PercentSalaryHike          float64
          StockOptionLevel           float64
          TotalWorkingYears          float64
          TrainingTimesLastYear      float64
          YearsAtCompany             float64
          YearsSinceLastPromotion    float64
          YearsWithCurrManager       float64
          EnvironmentSatisfaction    float64
          JobSatisfaction            float64
          WorkLifeBalance            float64
          JobInvolvement             float64
          PerformanceRating          float64
          Avg Time Spent(Mins)       float64
          Absences                   float64
          dtype: object
```

```
In [117]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4411 entries, 0 to 4410
Data columns (total 28 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      4410 non-null   float64
 1   Attrition                4410 non-null   object
 2   BusinessTravel           4410 non-null   object
 3   Department               4410 non-null   object
 4   DistanceFromHome         4410 non-null   float64
 5   Education                4410 non-null   float64
 6   EducationField           4410 non-null   object
 7   EmployeeID               4410 non-null   float64
 8   Gender                   4410 non-null   object
 9   JobLevel                 4410 non-null   float64
 10  JobRole                  4410 non-null   object
 11  MaritalStatus            4410 non-null   object
 12  MonthlyIncome            4410 non-null   float64
 13  NumCompaniesWorked       4392 non-null   float64
 14  PercentSalaryHike        4410 non-null   float64
 15  StockOptionLevel         4410 non-null   float64
 16  TotalWorkingYears        4401 non-null   float64
 17  TrainingTimesLastYear    4410 non-null   float64
 18  YearsAtCompany           4410 non-null   float64
 19  YearsSinceLastPromotion  4410 non-null   float64
 20  YearsWithCurrManager     4410 non-null   float64
 21  EnvironmentSatisfaction  4385 non-null   float64
 22  JobSatisfaction          4390 non-null   float64
 23  WorkLifeBalance          4372 non-null   float64
 24  JobInvolvement           4410 non-null   float64
 25  PerformanceRating        4410 non-null   float64
 26  Avg Time Spent(Mins)     4410 non-null   float64
 27  Absences                 4410 non-null   float64
dtypes: float64(21), object(7)
memory usage: 965.0+ KB
```

Data Cleaning and Preprocessing:

After exploring the dataset, we were able to notice that there were multiple missing values under different columns that needed to be dealt with before we could run the given data through our models. There were 39 missing values in WorkLifeBalance, 26 in EnvironmentSatisfaction, 21 in JobSatisfaction, and 19 in NumCompaniesWorked, along with at least 1 under all other columns. We dropped all the rows that had missing values, and reset the index before running the data through different normalization, discretization, and encoding techniques to avoid any issues.

```
In [119]: data.isnull().sum()

Out[119]: Age                        1
          Attrition                  1
          BusinessTravel             1
          Department                 1
          DistanceFromHome           1
          Education                  1
          EducationField             1
          EmployeeID                 1
          Gender                     1
          JobLevel                   1
          JobRole                    1
          MaritalStatus              1
          MonthlyIncome              1
          NumCompaniesWorked        19
          PercentSalaryHike          1
          StockOptionLevel           1
          TotalWorkingYears         10
          TrainingTimesLastYear      1
          YearsAtCompany             1
          YearsSinceLastPromotion    1
          YearsWithCurrManager       1
          EnvironmentSatisfaction   26
          JobSatisfaction           21
          WorkLifeBalance           39
          JobInvolvement             1
          PerformanceRating          1
          Avg Time Spent(Mins)       1
          Absences                   1
          dtype: int64
```

```
In [120]: data.dropna(inplace=True)

In [122]: data.reset_index(drop=True, inplace=True)

In [123]: data.head(10)
```

Out[123]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeID	Gender	JobLevel	...	YearsAtCompany	Year
0	51.0	No	Travel_Rarely	Sales	6.0	2.0	Life Sciences	1.0	Female	1.0	...	1.0	
1	31.0	Yes	Travel_Frequently	Research & Development	10.0	1.0	Life Sciences	2.0	Female	1.0	...	5.0	
2	32.0	No	Travel_Frequently	Research & Development	17.0	4.0	Other	3.0	Male	4.0	...	5.0	
3	38.0	No	Non-Travel	Research & Development	2.0	5.0	Life Sciences	4.0	Male	3.0	...	8.0	
4	32.0	No	Travel_Rarely	Research & Development	10.0	1.0	Medical	5.0	Male	1.0	...	6.0	
5	46.0	No	Travel_Rarely	Research & Development	8.0	3.0	Life Sciences	6.0	Female	4.0	...	7.0	
6	28.0	Yes	Travel_Rarely	Research & Development	11.0	2.0	Medical	7.0	Male	2.0	...	0.0	
7	29.0	No	Travel_Rarely	Research & Development	18.0	3.0	Life Sciences	8.0	Male	2.0	...	0.0	
8	31.0	No	Travel_Rarely	Research & Development	1.0	3.0	Life Sciences	9.0	Male	3.0	...	9.0	
9	25.0	No	Non-Travel	Research & Development	7.0	4.0	Medical	10.0	Female	4.0	...	6.0	

Firstly, we applied Discretization on the Years at Company columns, where people who had worked between 1-5 years were labelled as Apprentice, while the next level included Journeyman up until 10 years. People who had worked for between 10-15 years while Experts, and any employee above that duration was considered a Master at their job.

```
In [132]: #Discretization (Years at Company)
          data['YearsAtCompany'] = np.where((data['YearsAtCompany'] > 1) & (data['YearsAtCompany'] <= 5), 'Apprentice',
                                   np.where((data['YearsAtCompany'] > 5) & (data['YearsAtCompany'] <= 10), 'Journeyman',
                                            np.where(data['YearsAtCompany'] > 15, 'Expert', 'Master')))
```

The next step entailed Manual Encoding of the Gender tab, where the Male data was replaced with ‘1’, and the Females were labelled as ‘0’. We also used discretization on the Average Time Spent (Mins) column using data distribution, and the data was distributed into different ranges, labelled as Lower outlier, Low, Low Median, High Median, High, and upper outlier.

```
In [136]: data['Gender']=data['Gender'].replace(['Male','Female'],[1,0])
```

```
In [138]: #Discretization using Data Distribution (Avg Time Spent(Mins)
          Q1=data ["Avg Time Spent(Mins)"].quantile(0.25)
          Q2=data['Avg Time Spent(Mins)'].quantile(0.50)
          Q3=data[ 'Avg Time Spent(Mins)'].quantile(0.75)
          IQR=Q3-Q1
          uw=Q3+1.5*IQR
          lw=Q1-1.5*IQR
```

```
In [139]: column = 'Avg Time Spent(Mins)'

          data[column] = np.where(data[column] < lw, 'lower_outlier',
                                  np.where((data[column] >= lw) & (data[column] <= Q1), 'low',
                                           np.where((data[column] > Q1) & (data[column] <= Q2), 'low_median',
                                                    np.where((data[column] > Q2) & (data[column] <= Q3), 'high_median',
                                                             np.where((data[column] > Q3) & (data[column] <= Q3 + 1.5 * IQR), 'high',
                                                                      'upper_outlier')))))
```

We also applied Z Score Normalization on the Distance From Home column, as well as the absences column as a standardization feature, so that all the features are being evaluated on the same scale. The Year with Current Manager tab was run through the MinMax Normalization to ensure that there was reduced impact of any outliers, and the data was prescribed within a given range for easier interpretability.

```
In [146]: #Z Score Normalization (Distance From Home)
          from sklearn.preprocessing import StandardScaler
          scaler = StandardScaler()
          data['DistanceFromHome'] = scaler.fit_transform(data[['DistanceFromHome']]).round(2)
```

```
In [152]: #Z Score Normalization (Absences)

          scaler = StandardScaler()
          data['Absences'] = scaler.fit_transform(data[['Absences']]).round(2)
```

```
In [150]: #MinMax Normalization (Years with current manager)
          data['YearsWithCurrManager']=scaler.fit_transform(data[['YearsWithCurrManager']]).round(2)
```

The final processing step in order to transform the categorical variables as binary vectors, we applied the One-hot encoding, so that the data was much more compatible with the machine learning algorithms, and it became easier for us to run the data through these models, such as the logistic regression and other processing, increasing interpretability and making it easier to find relationship and improving their performance.

```
In [191]:
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
```

```
In [192]:
encoded_df = pd.get_dummies(data, columns=['Age', 'YearsAtCompany'])
```

```
In [193]:
categorical_cols = ['BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus']
label_encoder = LabelEncoder()
```

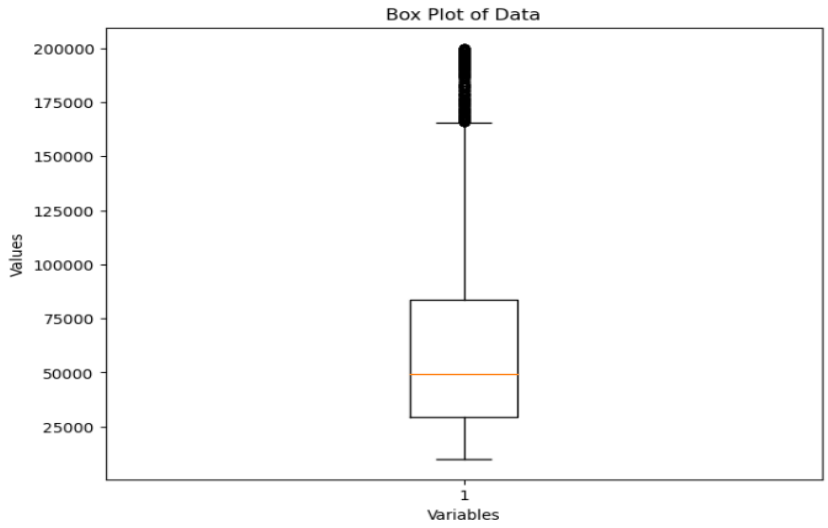
```
In [194]:
encoded_time_spent = label_encoder.fit_transform(encoded_df['Avg Time Spent(Mins)'])

# Replace the original column with the encoded values
encoded_df['Avg Time Spent(Mins)'] = encoded_time_spent
```

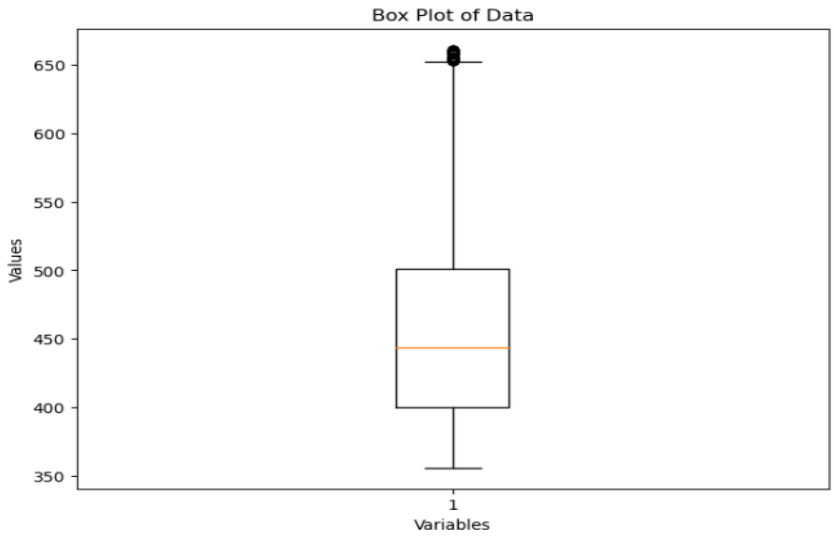
```
In [195]:
# Perform one-hot encoding for the categorical variable
encoded_df = pd.get_dummies(encoded_df, columns=['Avg Time Spent(Mins)'])
```

The two variables that were considered vulnerable to outliers were: Monthly Income, and Average Time Spent (mins), as the remaining columns were verified to be within the ranges, or were classifiers, such as Gender, however, by utilizing the boxplot feature in Python, we were able to detect no such extreme values that needed to be dealt with. The data seemed to be pretty well distributed amongst the ranges.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
plt.boxplot(data.MonthlyIncome)
plt.title('Box Plot of Data')
plt.xlabel('Variables')
plt.ylabel('Values')
plt.show()
```



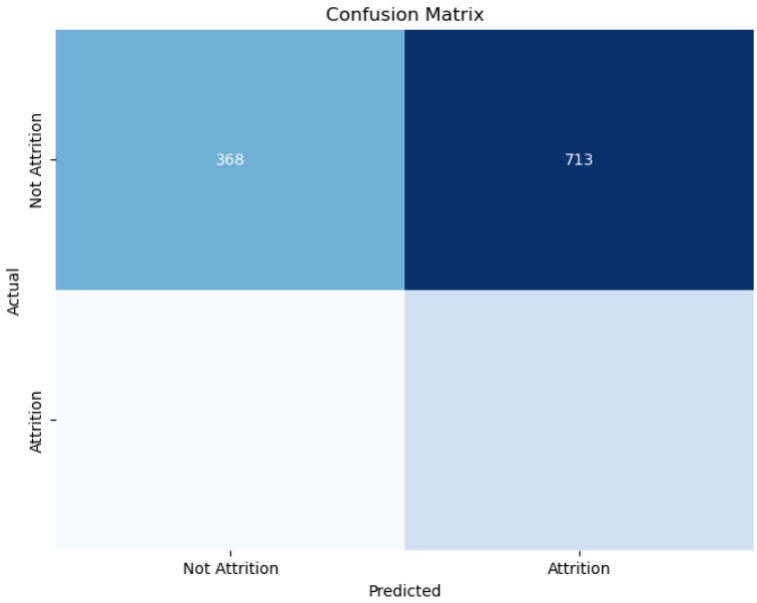
```
plt.figure(figsize=(8, 6))
plt.boxplot(datatime)
plt.title('Box Plot of Data')
plt.xlabel('Variables')
plt.ylabel('Values')
plt.show()
```



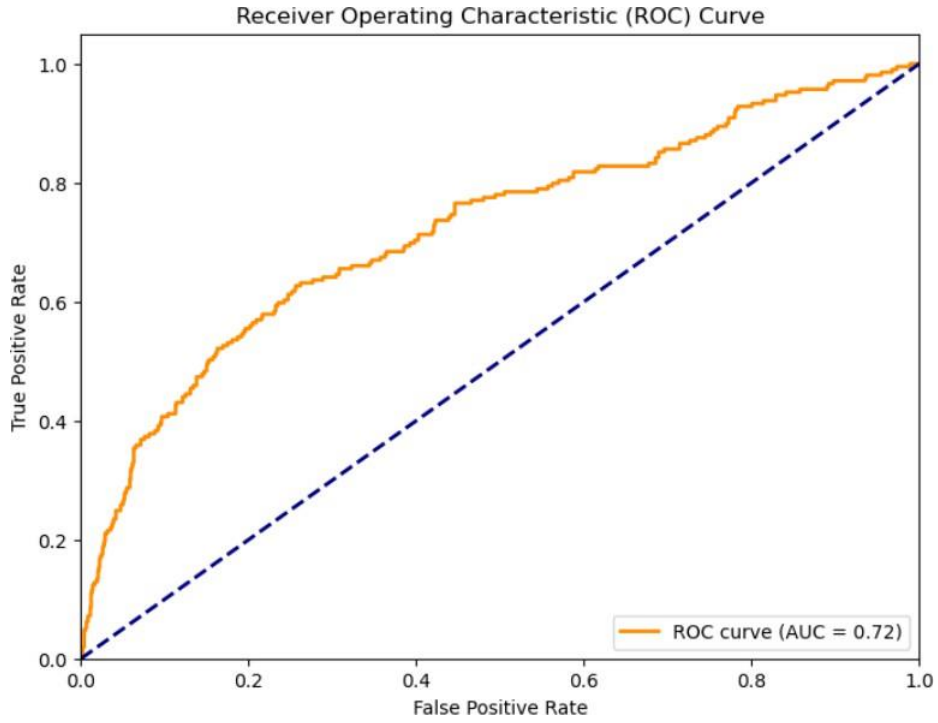
Machine Learning Models:

ANN:

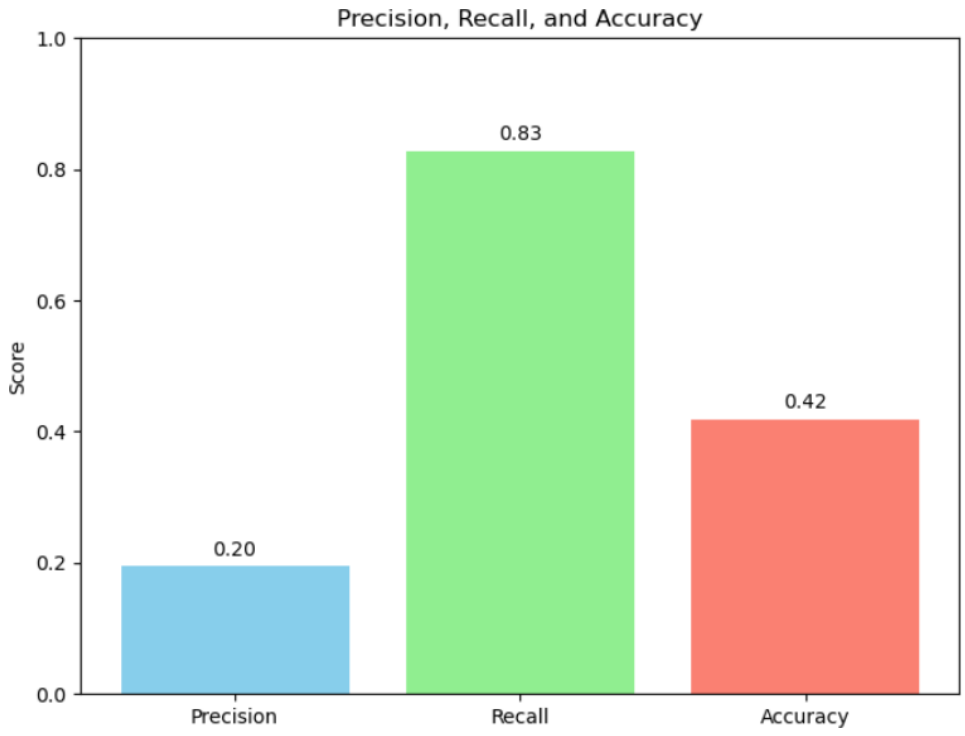
Confusion Matrix:



ROC Curve:

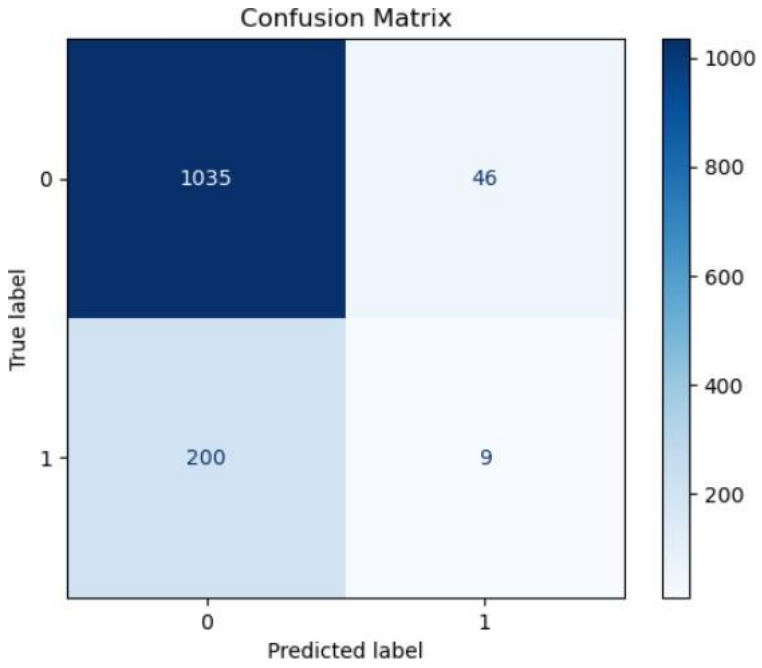


Precision, Recall & Accuracy:

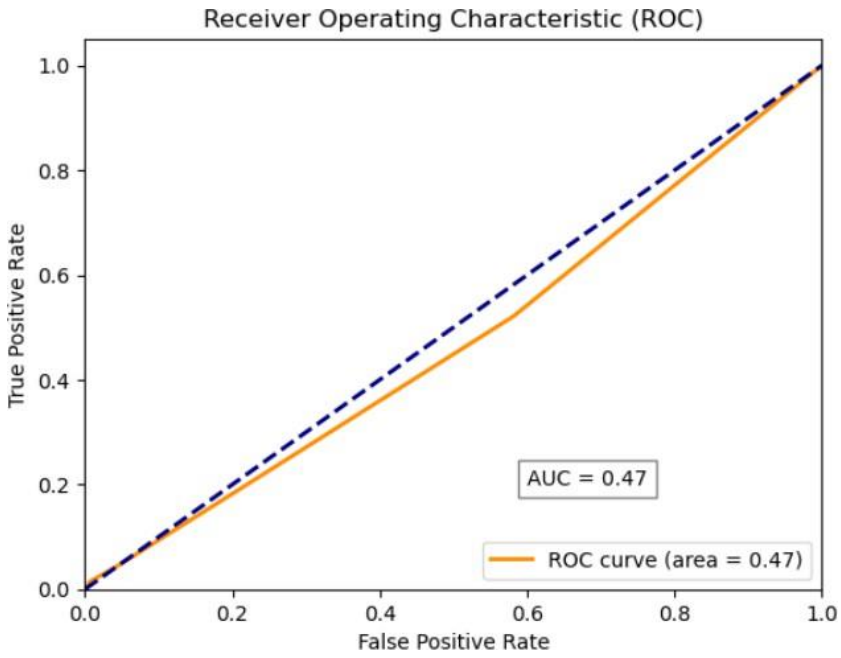


KNN:

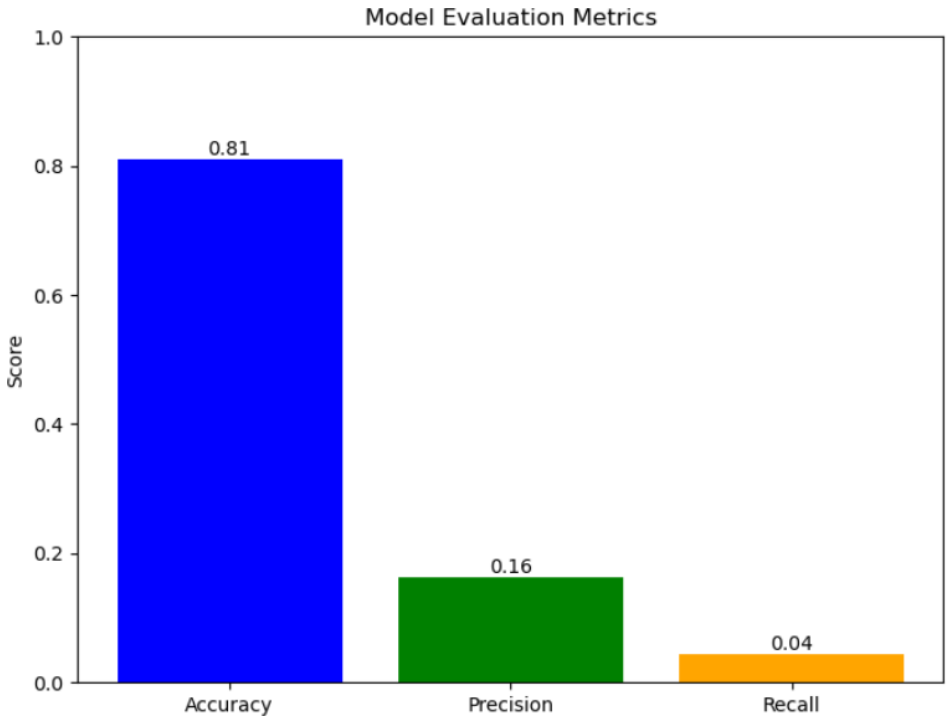
Confusion Matrix:



ROC Curve:

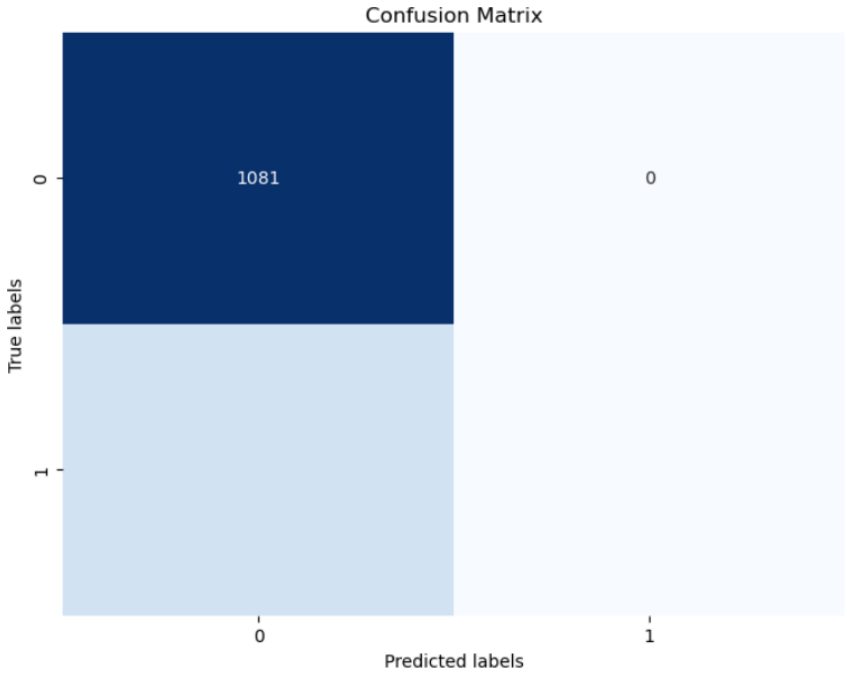


Precision, Recall & Accuracy:

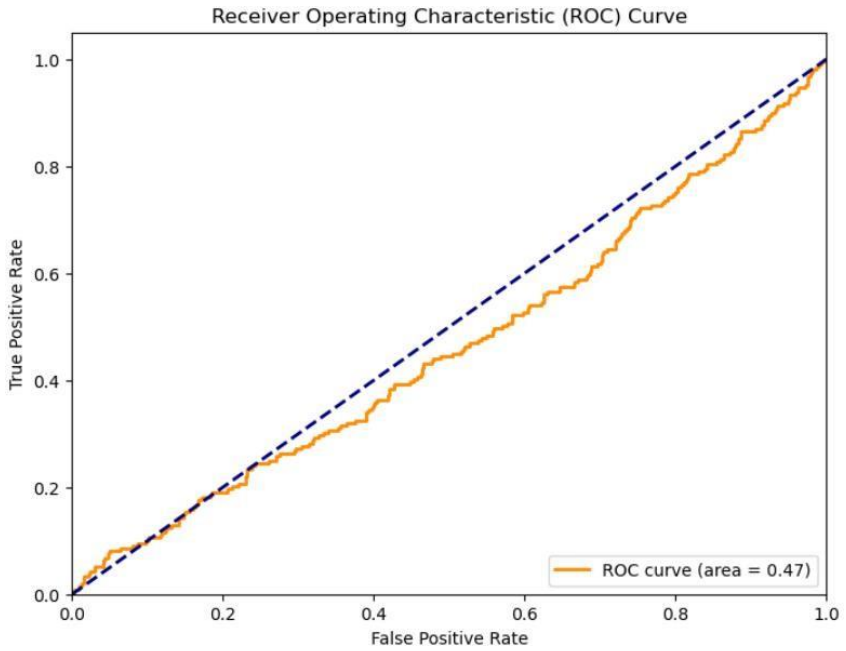


SVM:

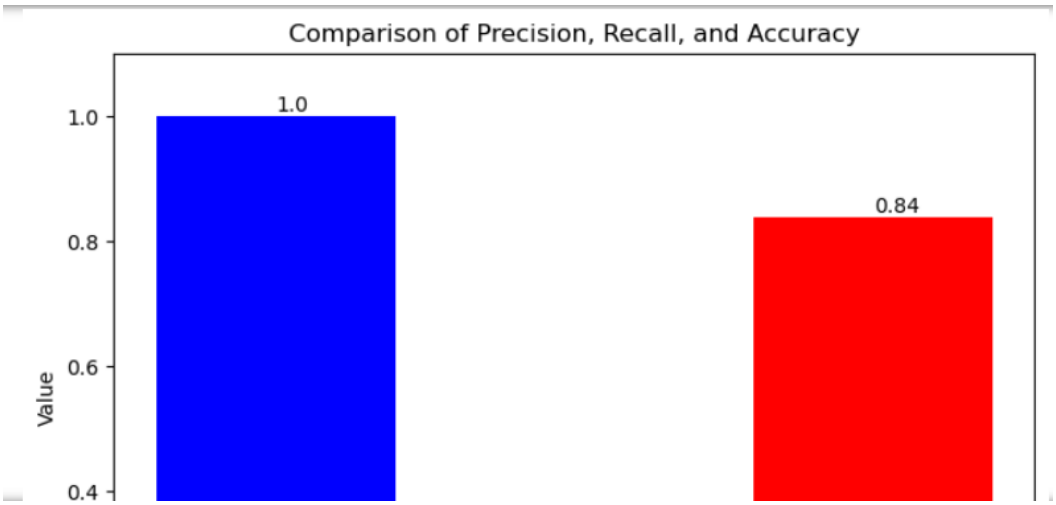
Confusion Matrix:



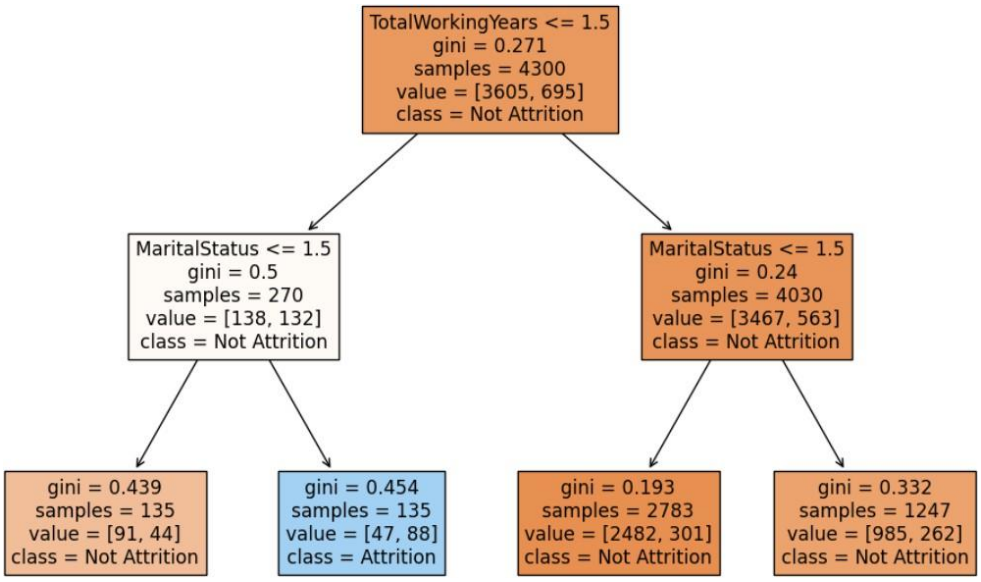
ROC Curve:



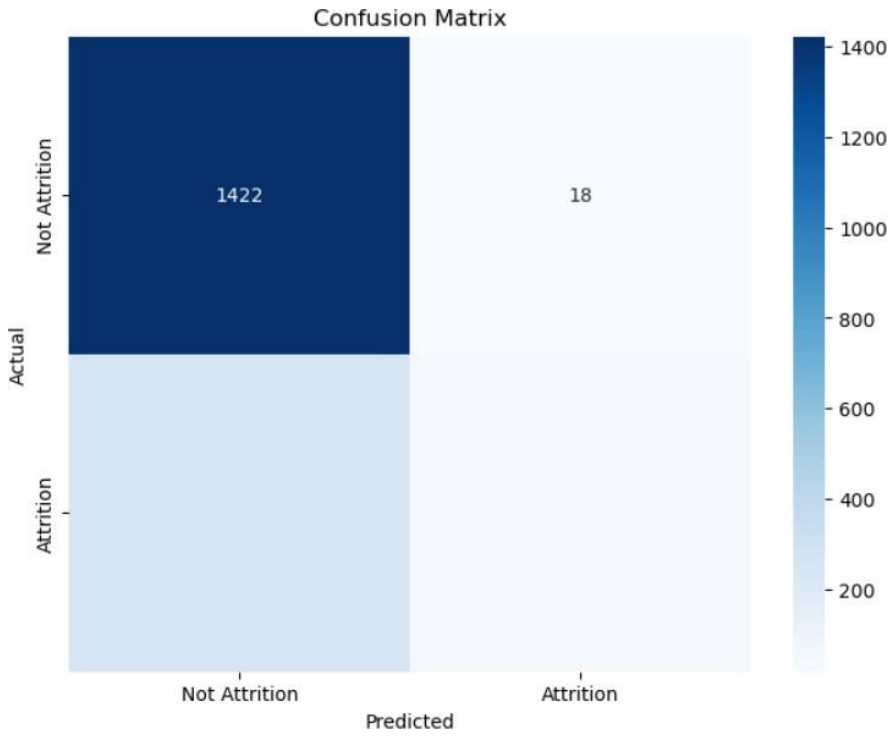
Precision, Recall & Accuracy:



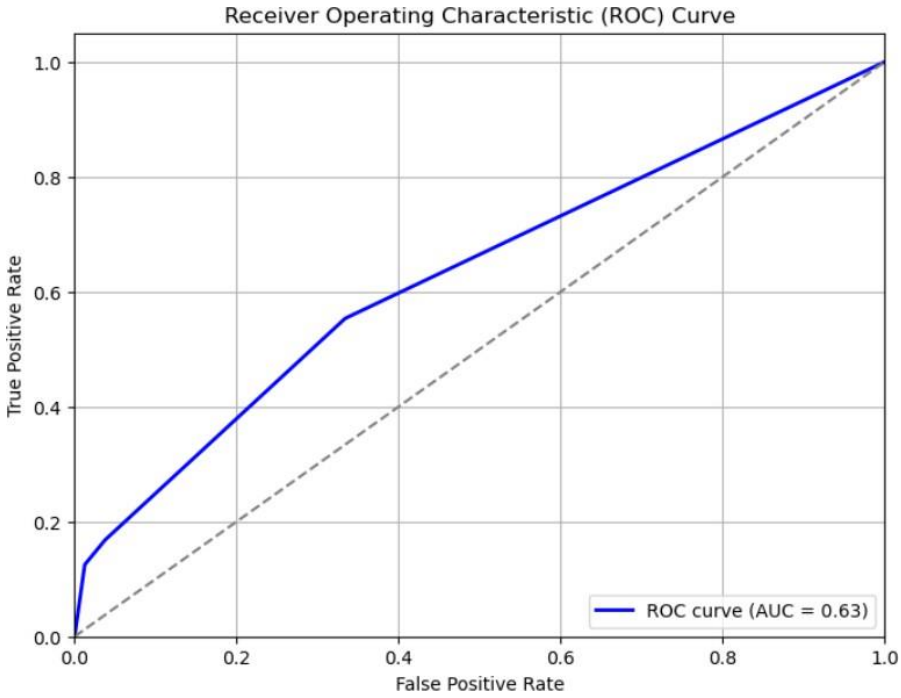
Decision Tree:



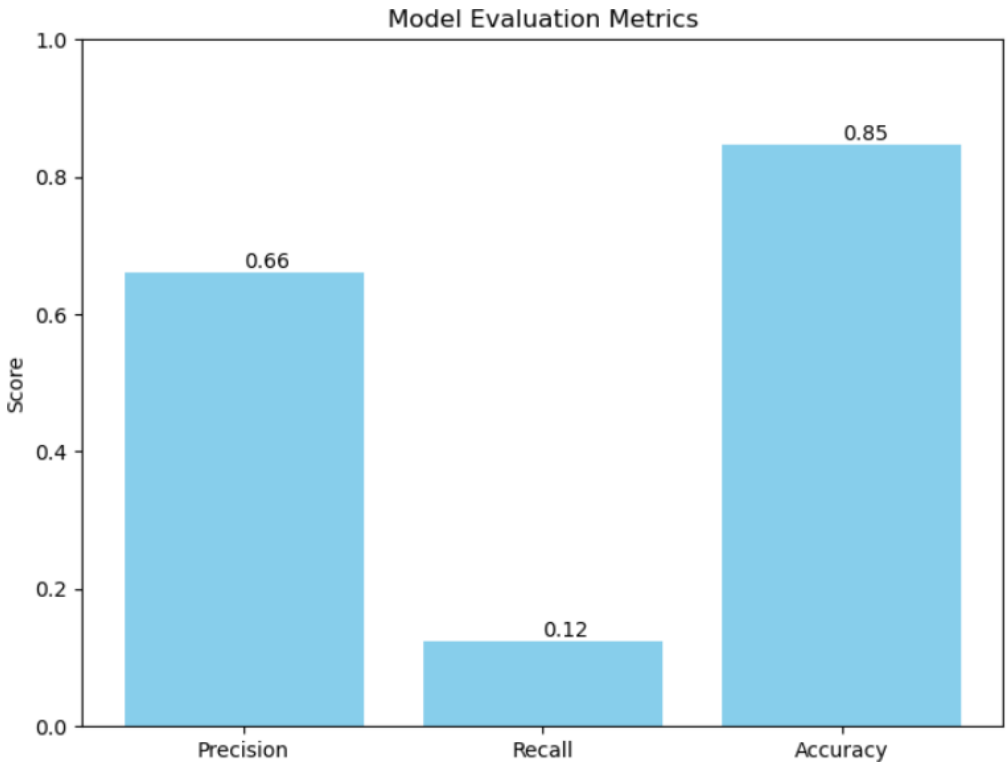
Confusion Matrix:



ROC Curve:

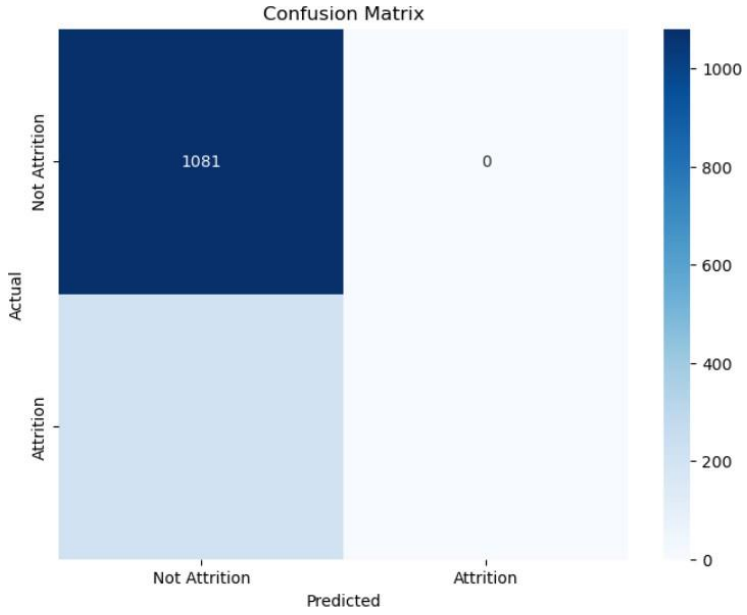


Precision, Recall & Accuracy:

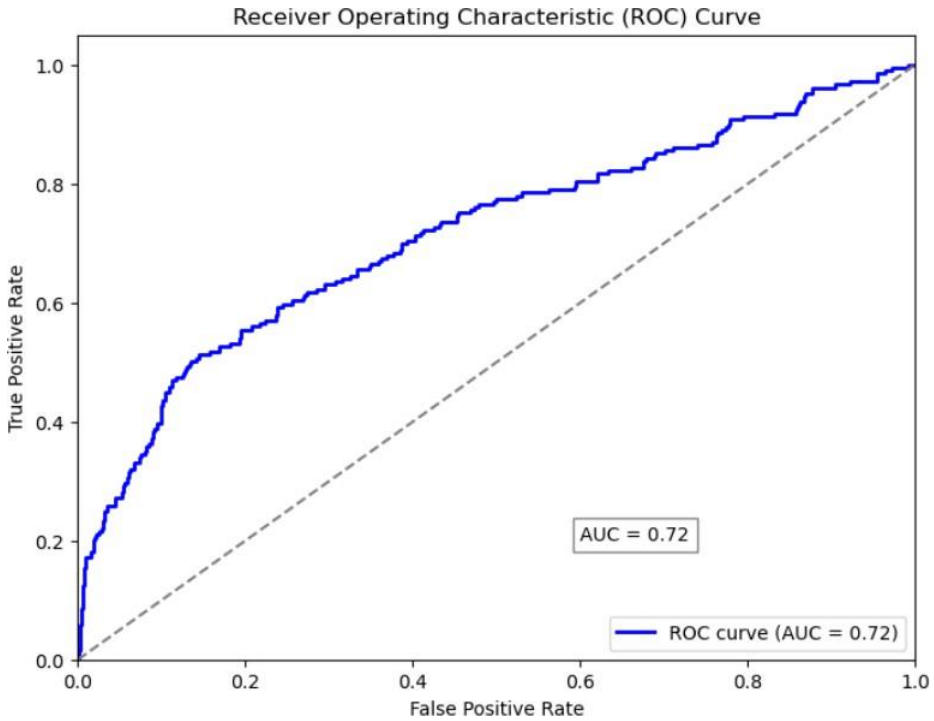


Naive Bayes:

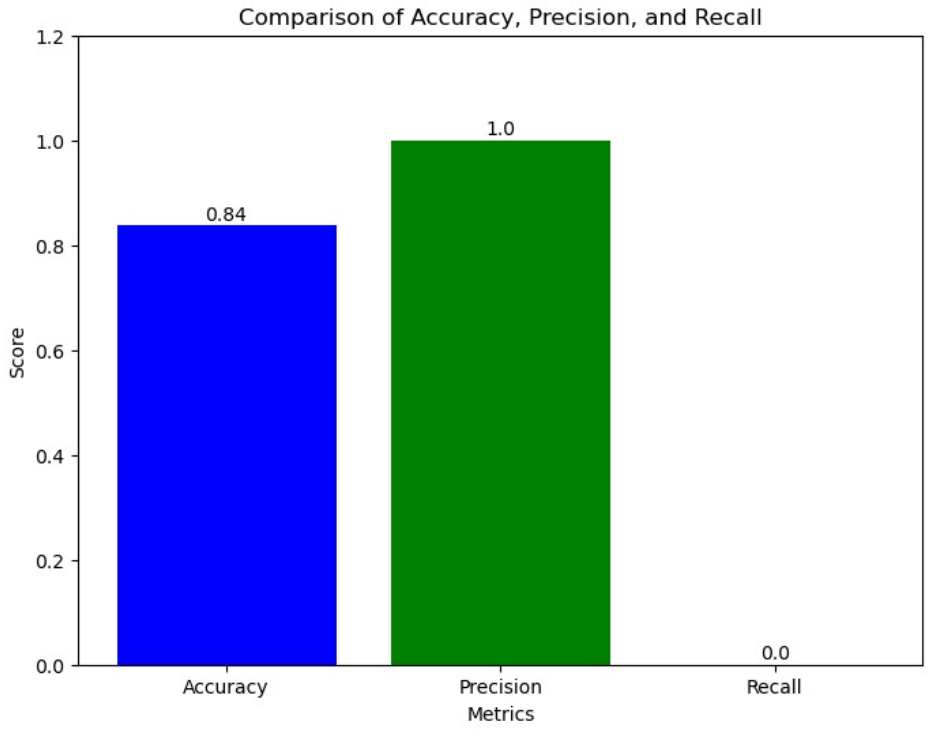
Confusion Matrix:



ROC Curve:

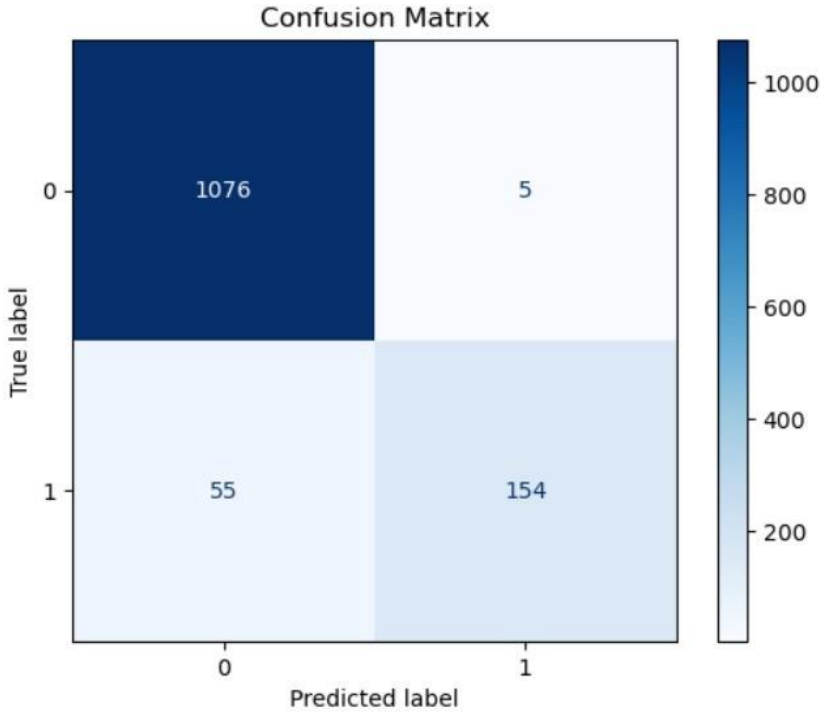


Precision, Recall & Accuracy:

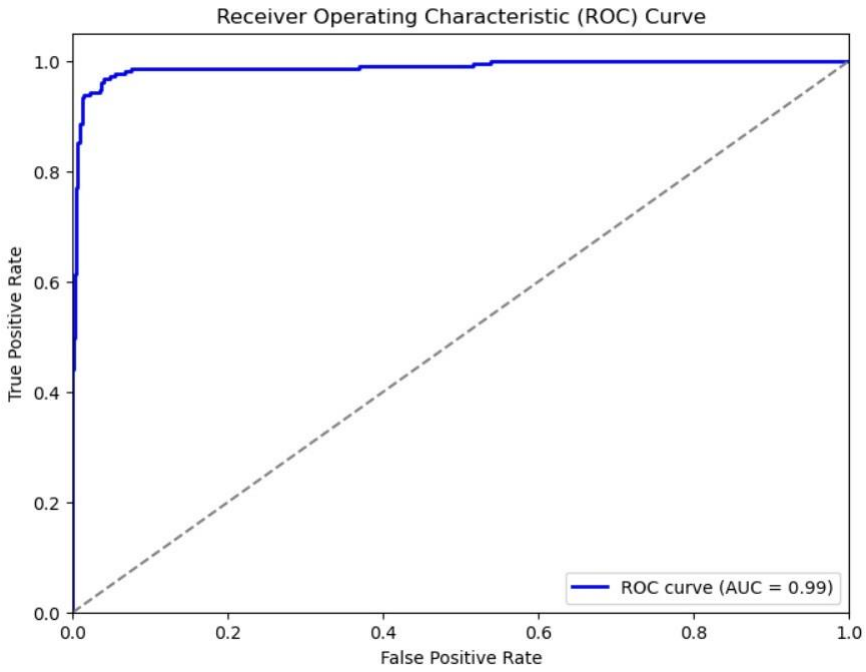


Random Forest:

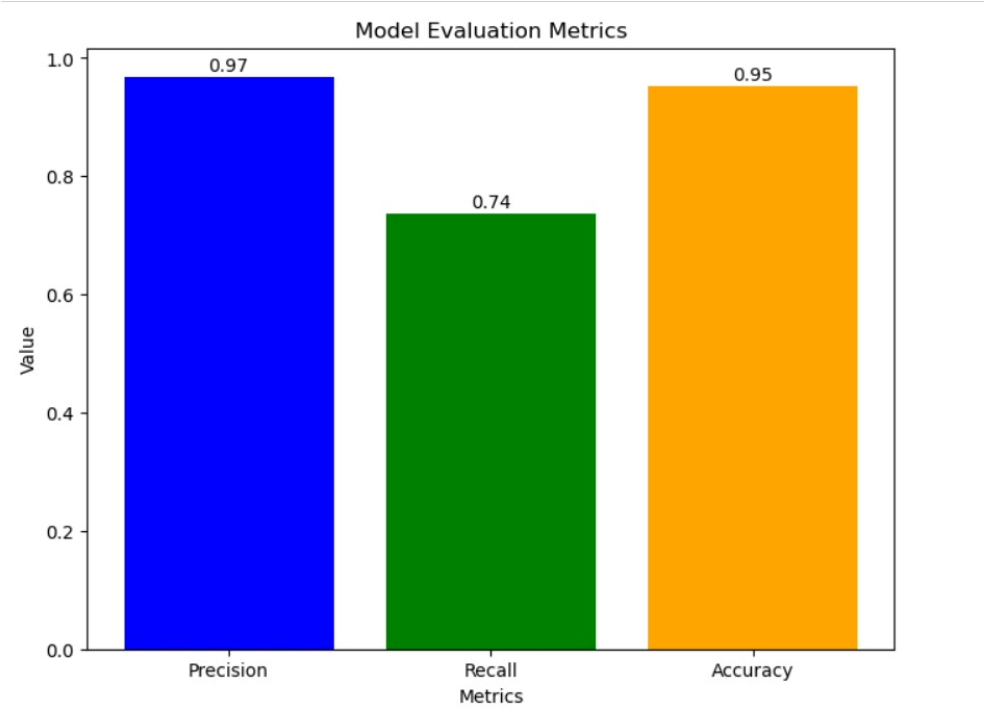
Confusion Matrix:



ROC Curve:

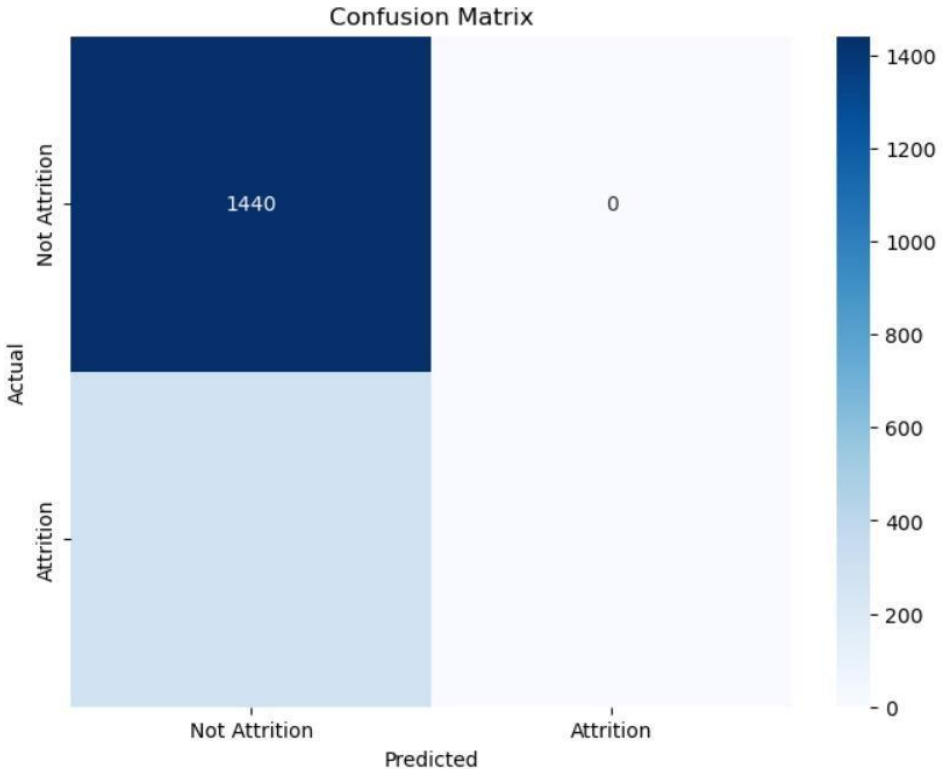


Precision, Recall & Accuracy:

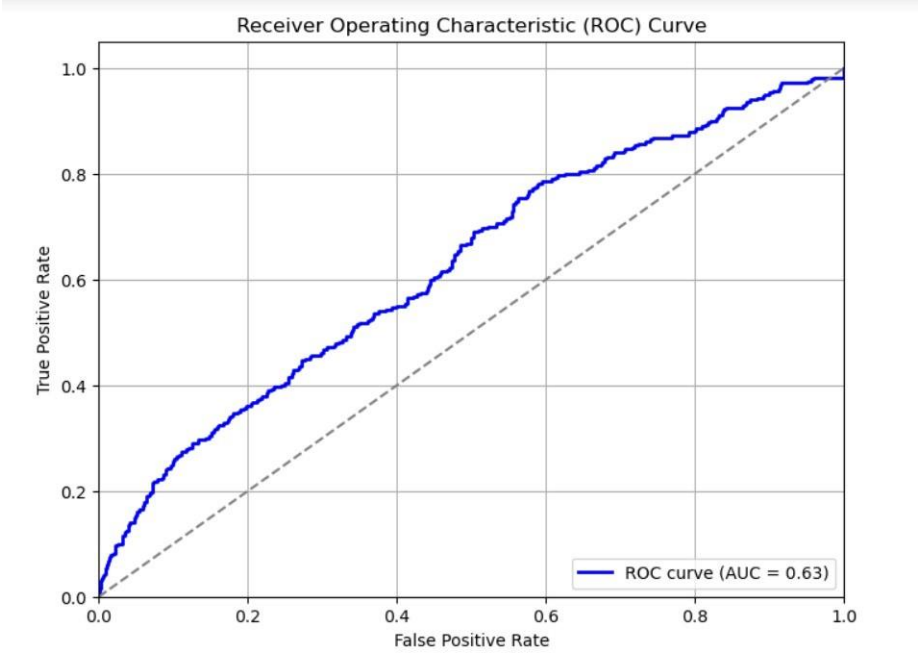


Logistic Regression:

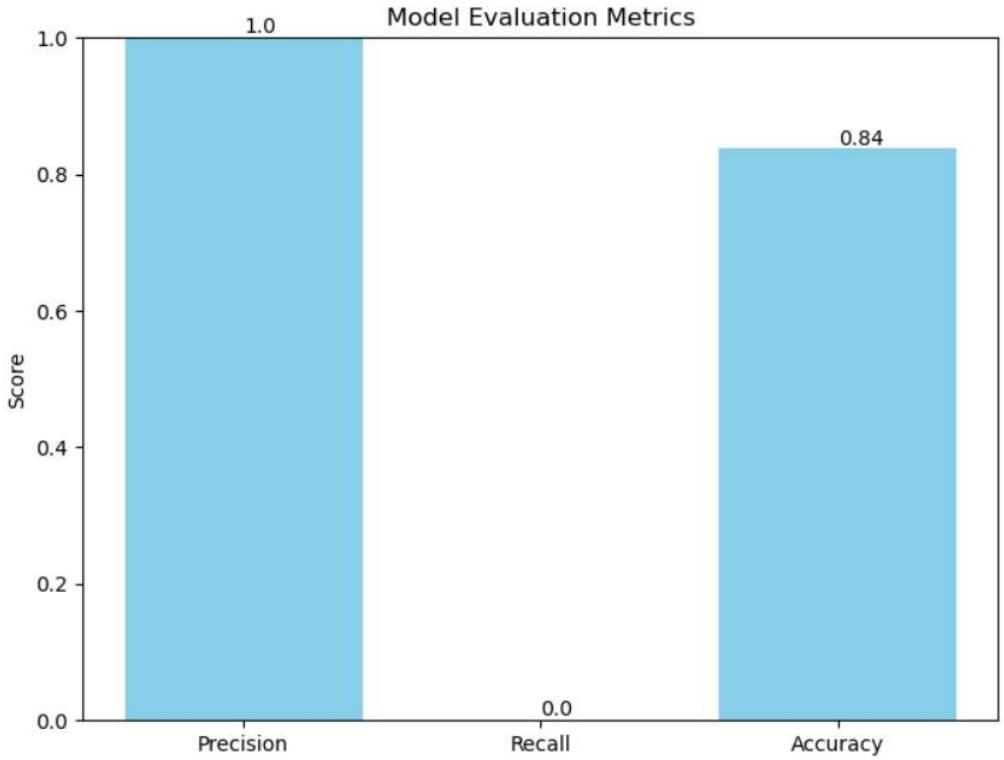
Confusion Matrix:



ROC Curve:



Precision, Recall & Accuracy:



Findings:

The Evaluation of supervised machine learning models for predicting attrition reveals a nuanced landscape of performance metrics, including accuracy, precision, recall, and AUC scores. While certain models excel in specific metrics, none achieve perfection in predicting attrition, underlining the necessity of weighing trade-offs between accuracy, precision, and recall based on organizational priorities. Models like Logistic Regression demonstrate high accuracy but may sacrifice recall, crucial for identifying actual attrition cases. Conversely, models with high recall, such as ANN, often incur lower precision, leading to more false positives. To address these trade-offs and enhance predictive performance, ensemble modeling emerges as a promising approach. By combining predictions from multiple models, ensemble methods leverage individual strengths and mitigate weaknesses, potentially outperforming any single model. However, further analysis is imperative for model refinement. This involves feature engineering to select or create

relevant predictors, gathering additional data to capture nuanced factors influencing attrition, and exploring advanced techniques like deep learning or NLP for deeper insights. Ultimately, the iterative process of model refinement and exploration ensures the development of robust attrition prediction models aligned with organizational objectives.