PORTOFOLIO

Employee Attrition Prediction using ML

Nabilla Salsa Billa
AI Hacker

PORTOFOLIO

Problem

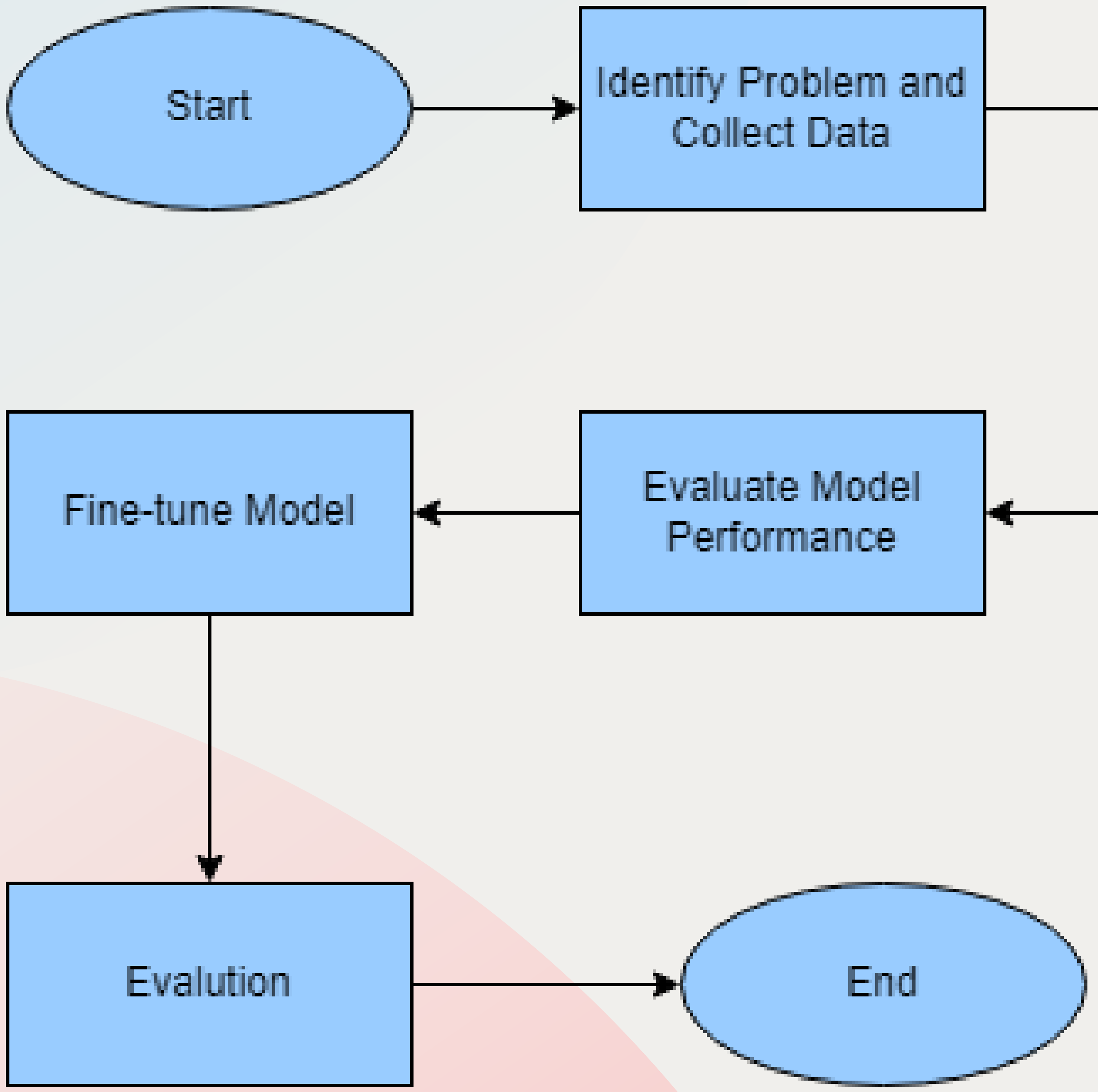
Employee is one of the most important resource in company, where a high attrition rate indicates that the company is unable to maintain their employees. In a short term, with high attrition rate, company must pay a great money to cover the cost of turnover. While in a long term, this will affect the company's performance as employees come and go the company's performance will decline.

Goal

To analyze the factors lead to employee attrition and make prediction of it, therefore company could give an appropriate treatment for the likely attrition employee.

PORTOFOLIO

Method

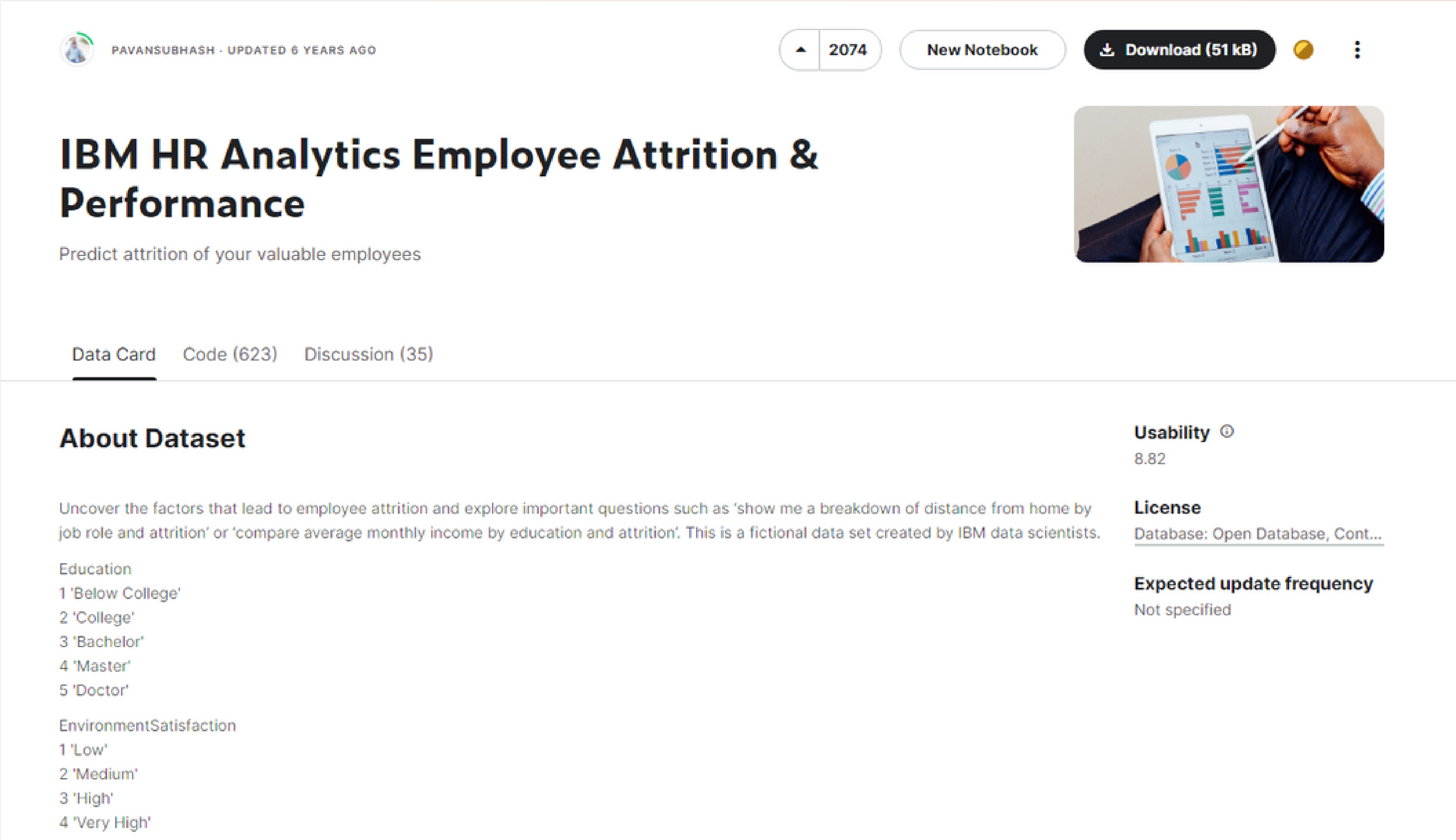


PORTOFOLIO

Dataset

IBM HR Analytics Employee Attrition & Performance

The data for this project was obtained from Kaggle, a popular online platform for machine learning and data science competitions.



https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset

PORTOFOLIO

Data Exploration

```
df = pd.read_csv("/content/drive/My Drive/dataset/WA_Fn-UseC_-HR-Employee-Attrition.csv")
df.head(5)
```

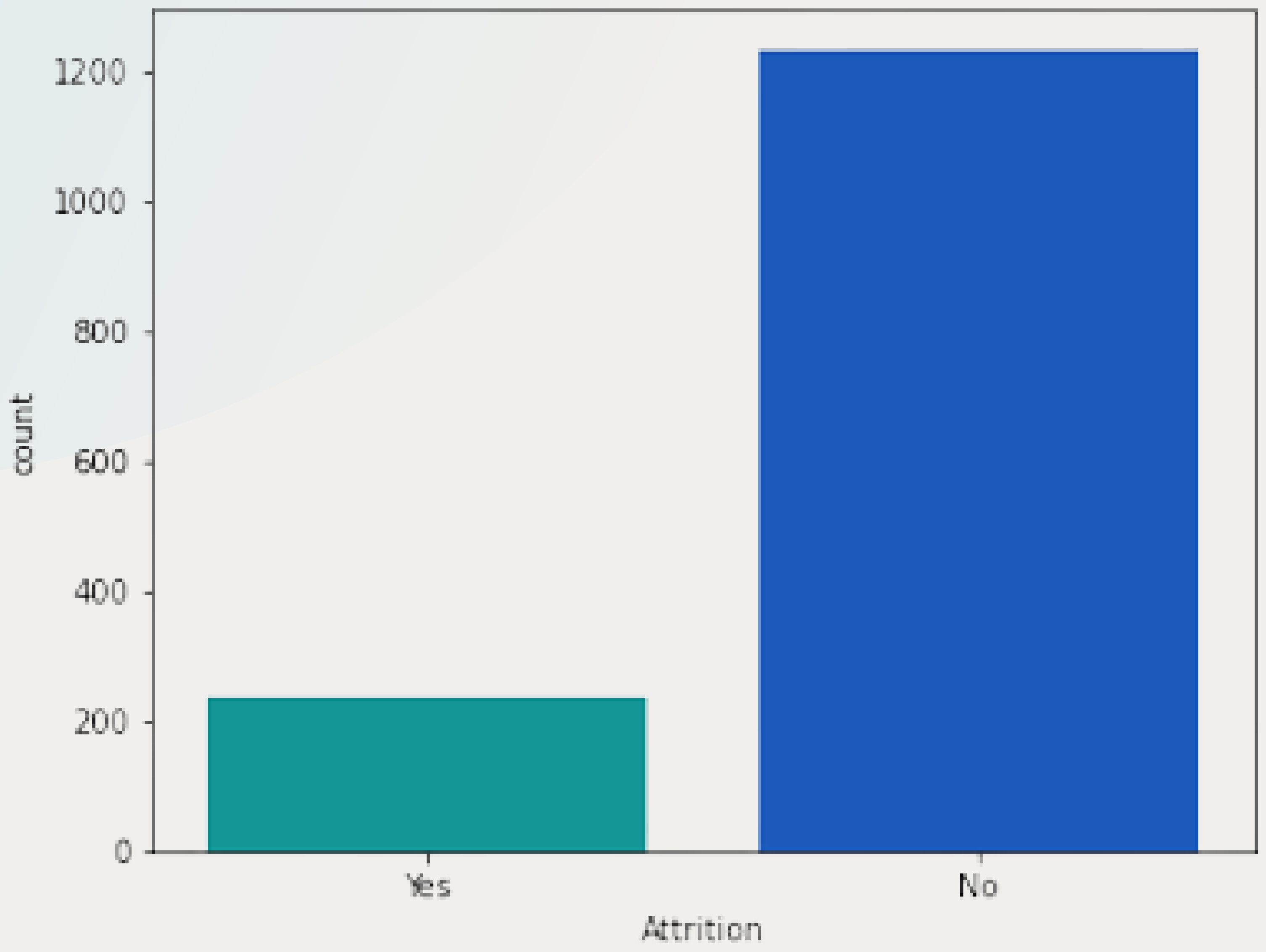
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...

5 rows × 35 columns

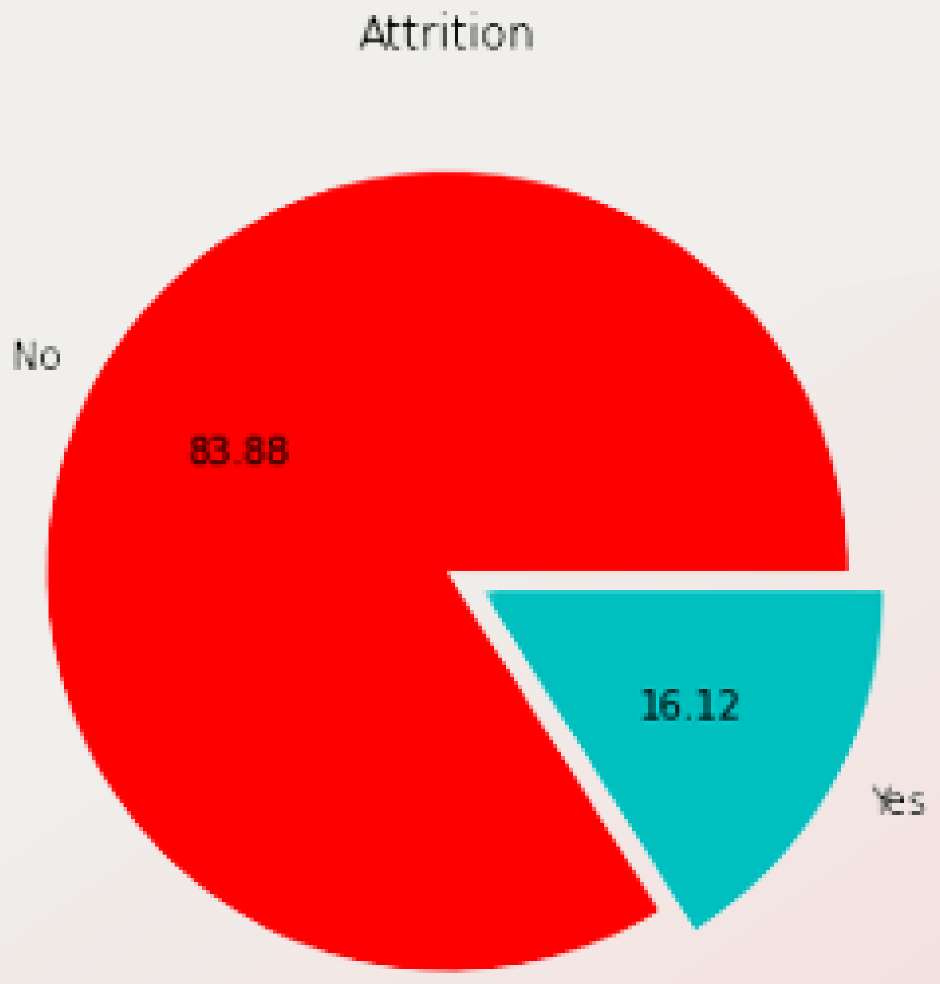
The dataset contains 1470 rows representing individual employees and 35 attributes or columns representing various characteristics or features of the employees.

PORTOFOLIO

Data Exploration

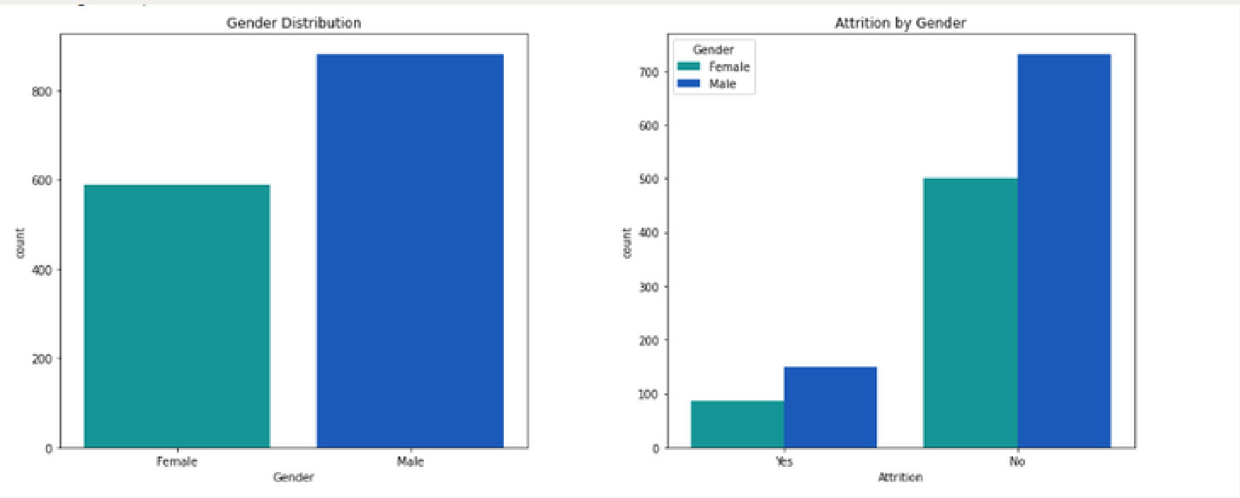


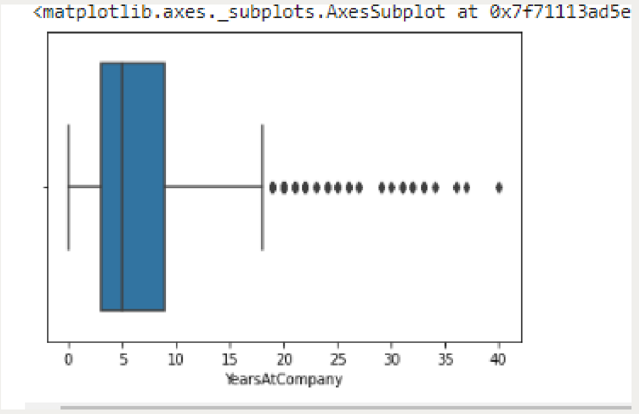
NO 1233
YES 237

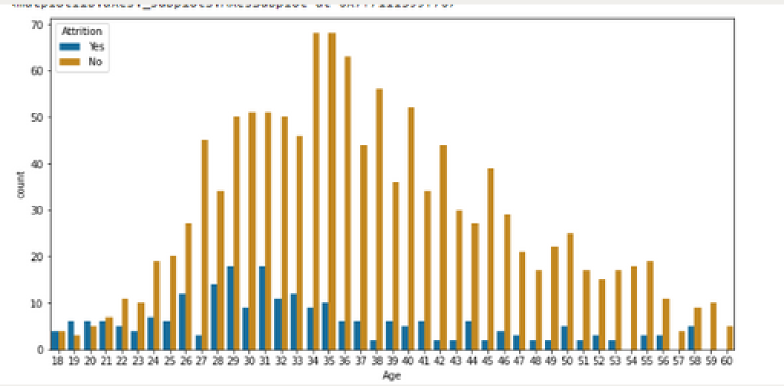


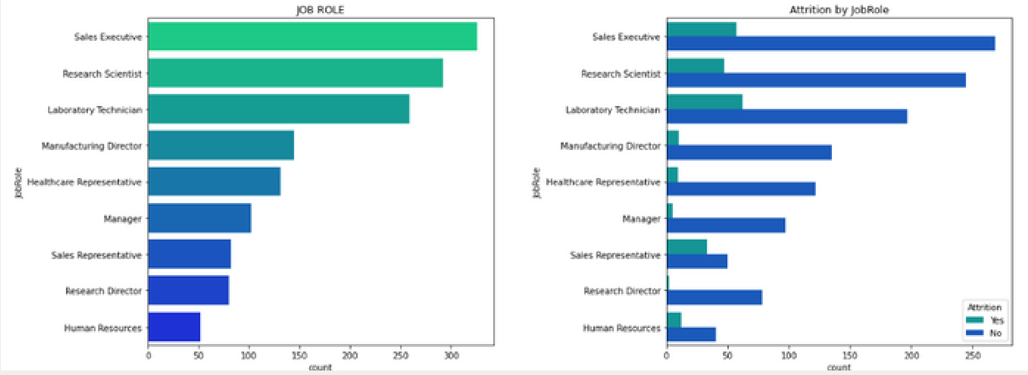
Almost 84% of the employees in the dataset have not left the company, indicating a relatively low attrition rate.

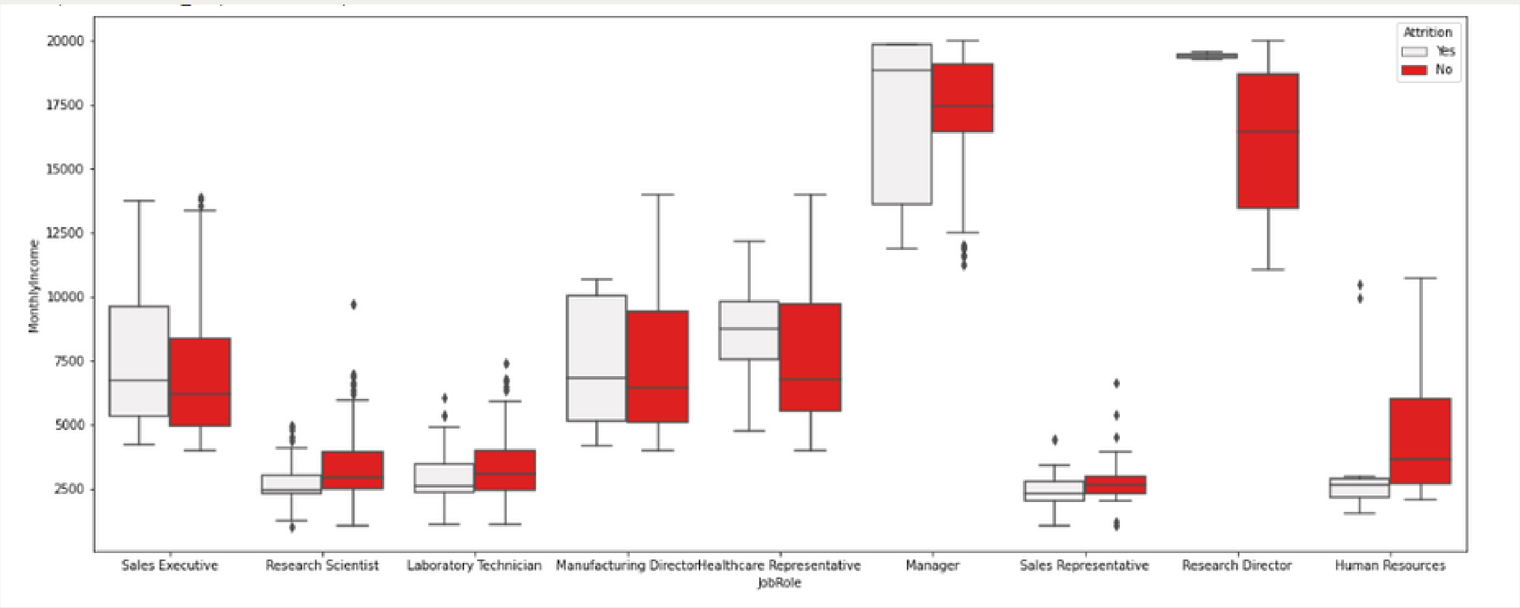
Data Exploration











- Male Employee has maximum Attrition rate when compared to females
- Employee Attrition is most in Sales Department and then Human Resources Department
- Employees having less salaries have more Attrition Rate
- Employees who has low job satisfaction and low environment satisfaction has maximum attrition rate.

Data Preparation

Drop Columns

```
df = df.drop(['EmployeeCount',
              'EmployeeNumber',
              'Over18',
               'StandardHours'],axis = 1)
df.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
       'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'OverTime',
       'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager'],
      dtype='object')
```

Categorical Encoding

```
#categorical features encoding

others = df.select_dtypes('object').columns
others
```

```
Index(['BusinessTravel', 'Department', 'EducationField', 'JobRole',
       'MaritalStatus'],
      dtype='object')
```

```
le = LabelEncoder()
for col in others:
    df[col] = le.fit_transform(df[col])
```

Data Preparation

Categorical Encoding

```python
#Binary Features Encoding

y_n_type = []
others =[]
for col in df.select_dtypes('object').columns:
    if(len(df[col].unique()) ==2):
        y_n_type.append(col)

y_n_type
```

```
['Attrition', 'Gender', 'OverTime']
```

```python
df['Gender'].replace({'Male':1 ,'Female':0} ,inplace = True)
df['OverTime'].replace({'Yes':1 ,'No':0} ,inplace = True)
df['Attrition'].replace({'Yes':1 ,'No':0} ,inplace = True)
```

Feature Scaling

```python
# Rescaling Data
Scaler = StandardScaler()
Scaling_Cols = ['TrainingTimesLastYear','YearsAtCompany','TotalWorkingYears',
                'YearsInCurrentRole','YearsSinceLastPromotion','YearsWithCurrManager',
                'PercentSalaryHike','Age','DailyRate','DistanceFromHome','HourlyRate',
                'MonthlyIncome','MonthlyRate','NumCompaniesWorked']
X[Scaling_Cols] = Scaler.fit_transform(X[Scaling_Cols])
```

```python
X
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate	...
0	0.446350	2	0.742527	2	-1.010909	2	1	2	0	1.383138	...
1	1.322365	1	-1.297775	1	-0.147150	1	1	3	1	-0.240677	...
2	0.008343	2	1.414363	1	-0.887515	2	4	4	1	1.284725	...
3	-0.429664	1	1.461466	1	-0.764121	4	1	4	0	-0.486709	...
4	-1.086676	2	-0.524295	1	-0.887515	1	3	1	1	-1.274014	...
...	...	...	...	...	...	...	...	...	...	...	...
1465	-0.101159	1	0.202082	1	1.703764	2	3	3	1	-1.224807	...
1466	0.227347	2	-0.469754	1	-0.393938	1	3	4	1	-1.175601	...
1467	-1.086676	2	-1.605183	1	-0.640727	3	1	2	1	1.038693	...
1468	1.322365	1	0.546677	2	-0.887515	3	3	4	1	-0.142264	...
1469	-0.320163	2	-0.432568	1	-0.147150	3	3	2	1	0.792660	...

1470 rows × 30 columns

Splitting Training and Testing Sets

```
# Splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.3)
print('X train size: ', len(X_train))
print('X test size: ', len(X_test))
print('y train size: ', len(y_train))
print('y test size: ', len(y_test))
```

```
X train size:  1726
X test size:  740
y train size:  1726
y test size:  740
```

Training Set

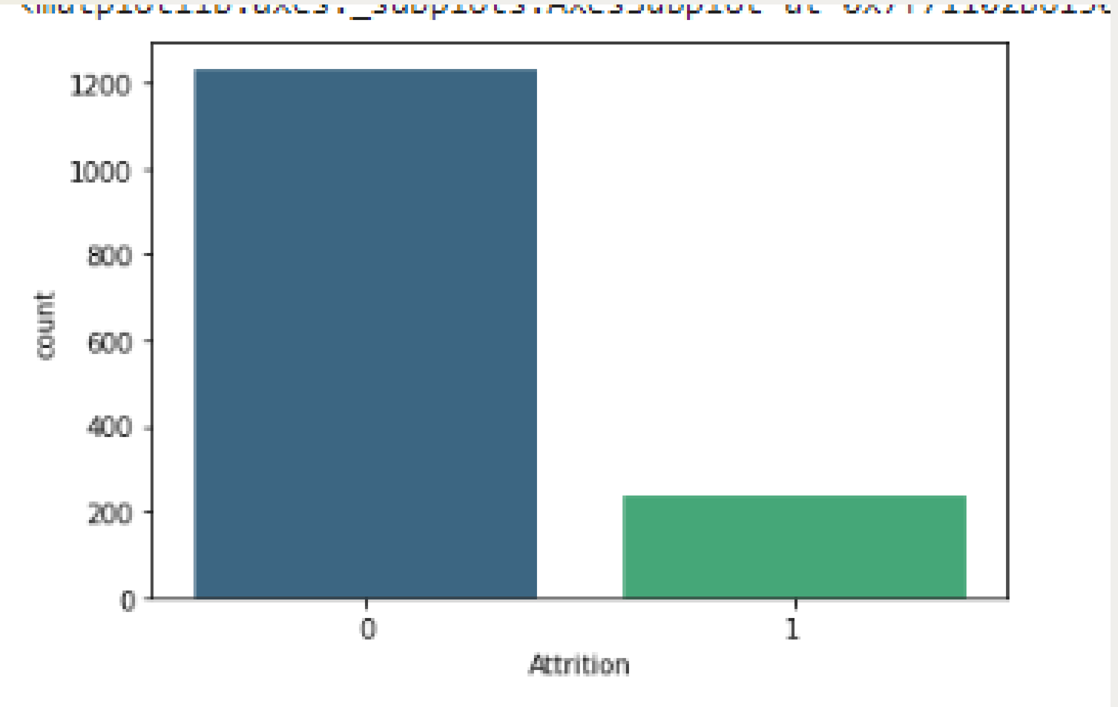
The training set is used to fit the model

Testing Set

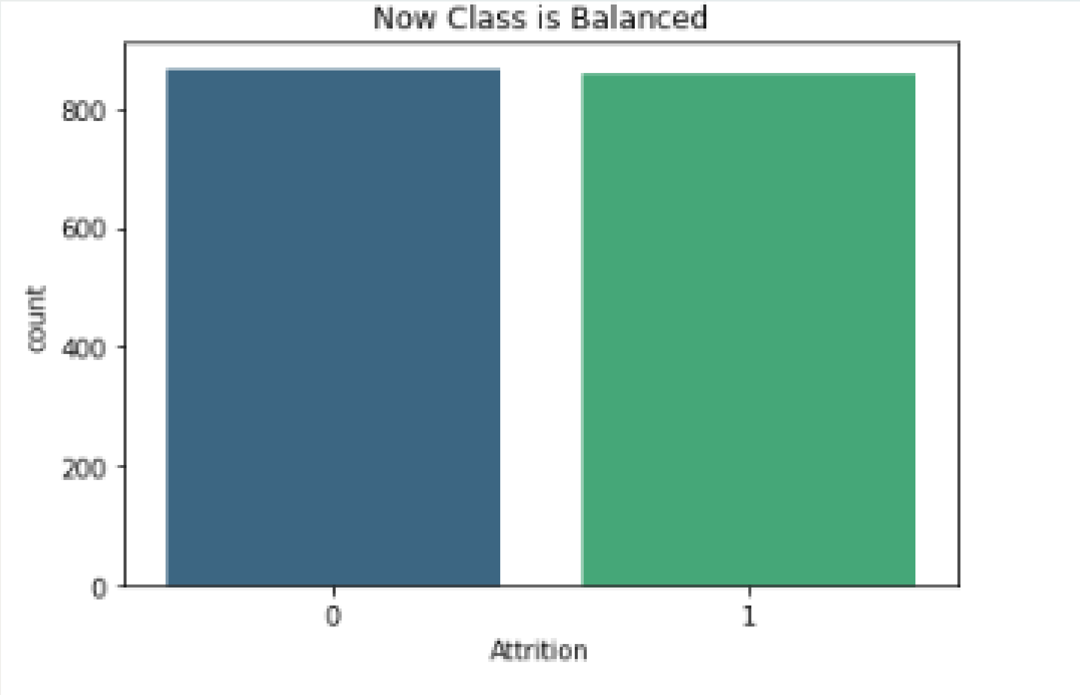
The test set is used to evaluate the model's performance on unseen data.

PORTOFOLIO

Handling Imbalance Dataset



BEFORE OVERSAMPLING



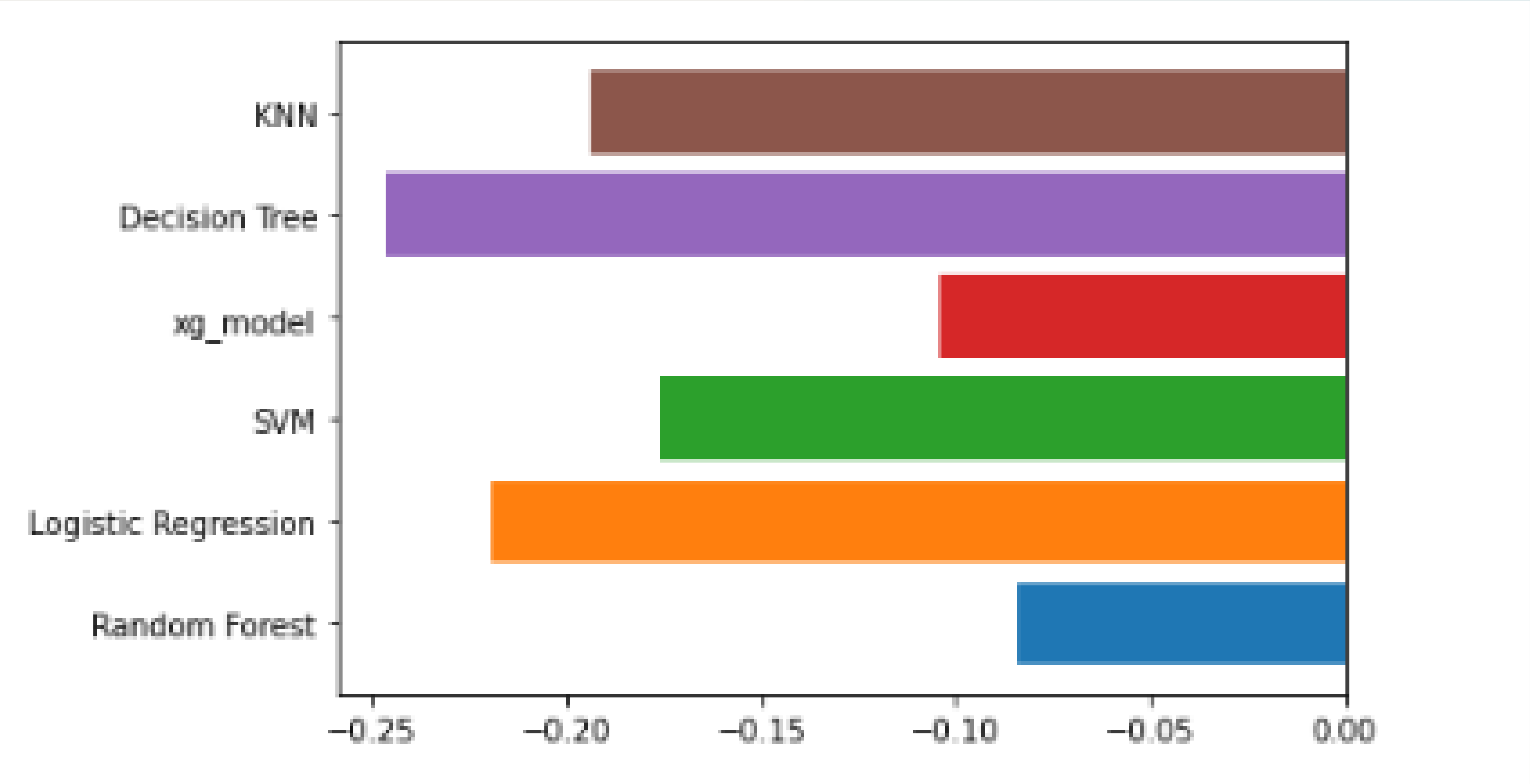
AFTER OVERSAMPLING

PORTOFOLIO

Model Selection

Random Forest:
neg_mean_squared_error = -0.076
Standard deviation = 0.007
Logistic Regression
neg_mean_squared_error = -0.191
Standard deviation = 0.017
SVM
neg_mean_squared_error = -0.149
Standard deviation = 0.017
xG Boost:
neg_mean_squared_error = -0.095
Standard deviation = 0.006
Decision Tree:
neg_mean_squared_error = -0.210
Standard deviation = 0.011
KNN:
neg_mean_squared_error = -0.176
Standard deviation = 0.015

Best model: Random Forest



PORTOFOLIO

Evaluating the model's performance on the test set

	model	accuracy	f1 score	roc auc score
0	Logistic Regression	0.831081	0.828532	0.831434
1	KNN	0.821622	0.848624	0.819361
2	Random Forest	0.924324	0.923077	0.924712
3	SVM	0.860811	0.859097	0.861132
4	XGBoost	0.901351	0.898752	0.901863

Fine-tune Model

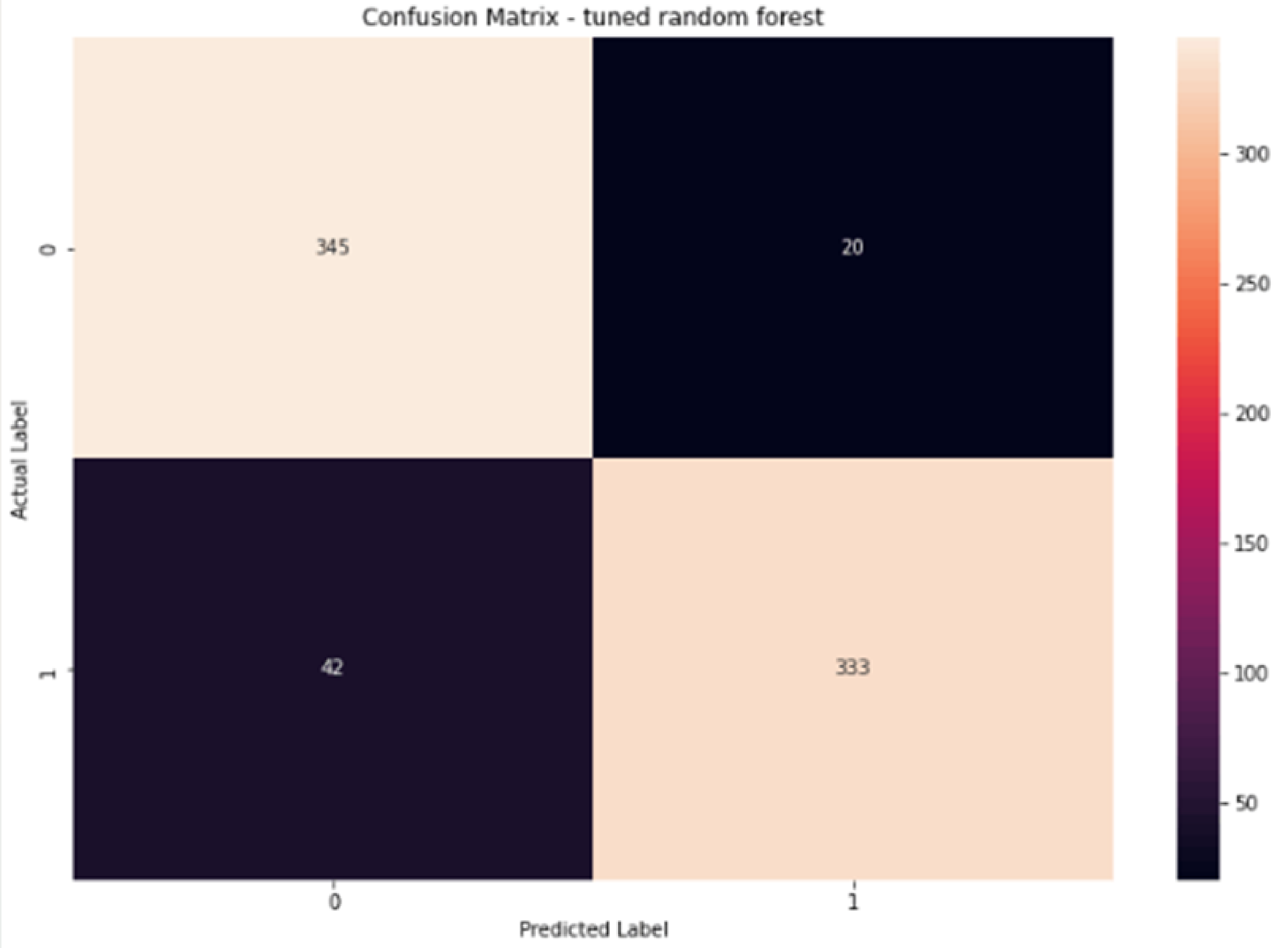
Random Forest

```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92       365
           1       0.94      0.89      0.91       375

    accuracy                           0.92       740
   macro avg       0.92      0.92      0.92       740
weighted avg       0.92      0.92      0.92       740

[Text(0, 0.5, '0'), Text(0, 1.5, '1')]
```



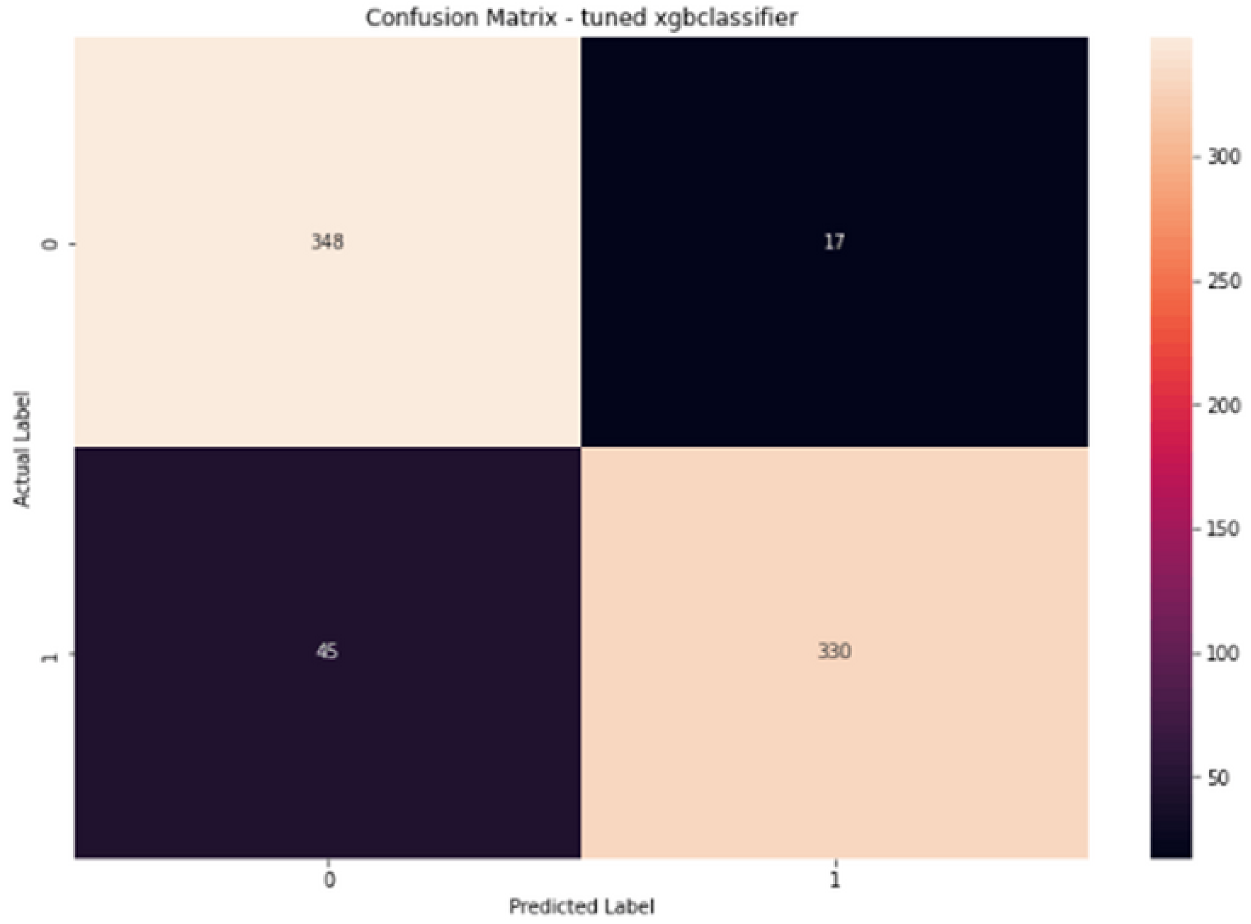
XGBoost

```
Accuracy: 91.62%
              precision    recall  f1-score   support

           0       0.89      0.95      0.92       365
           1       0.95      0.88      0.91       375

    accuracy                           0.92       740
   macro avg       0.92      0.92      0.92       740
weighted avg       0.92      0.92      0.92       740

[Text(0, 0.5, '0'), Text(0, 1.5, '1')]
```



Fine-tune Model

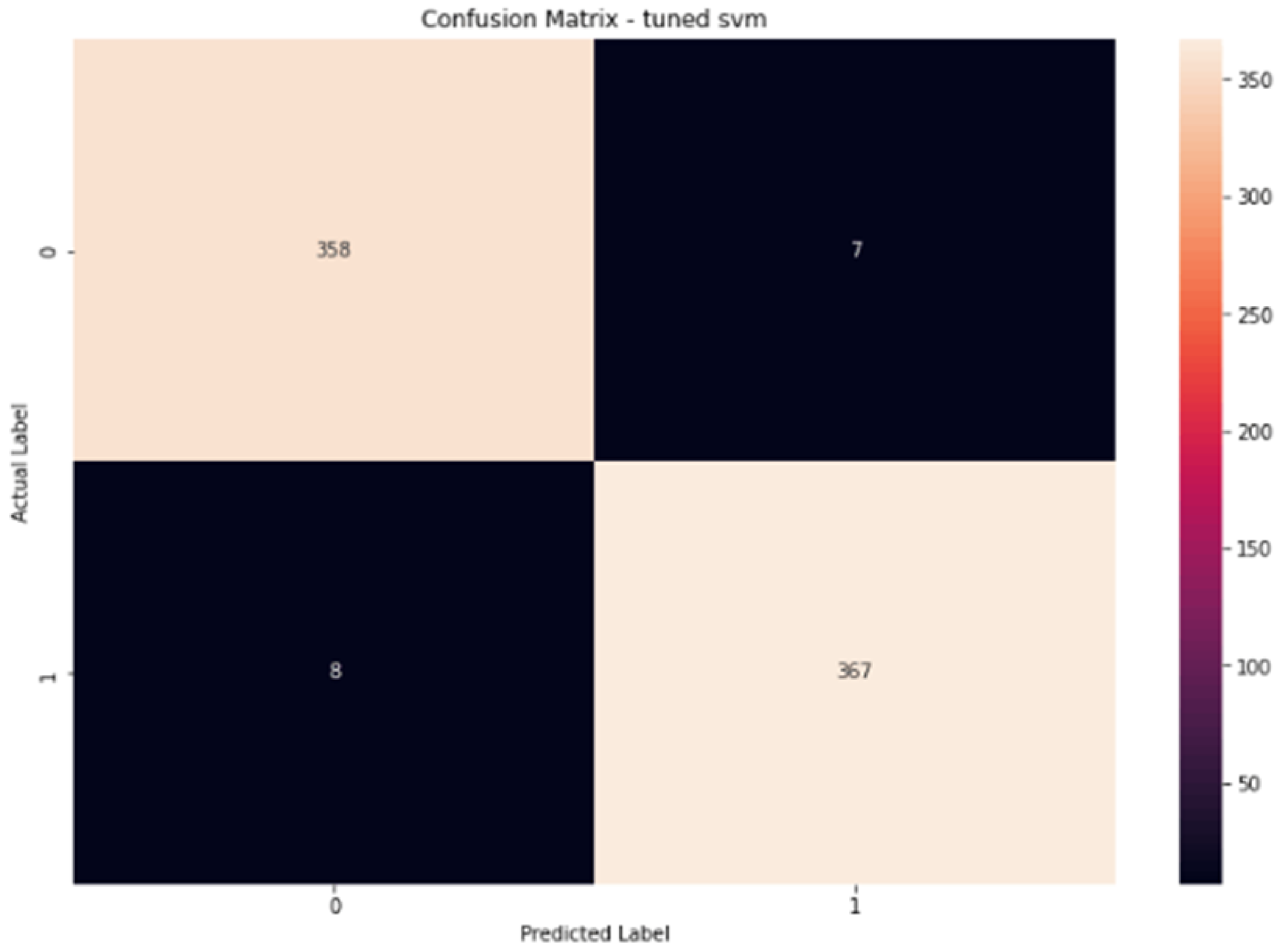
SVM

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       365
           1       0.98      0.98      0.98       375

    accuracy                           0.98       740
   macro avg       0.98      0.98      0.98       740
weighted avg       0.98      0.98      0.98       740

[Text(0, 0.5, '0'), Text(0, 1.5, '1')]
```



PORTOFOLIO

Conclusion

In conclusion, the results of this study indicate that the Random Forest model performed the best among the tested models in terms of both accuracy 92.4% and f1-score 0.92% when no hyperparameters were used. The XGBoost model also showed good performance, but was slightly outperformed by the Random Forest model. The SVM model performed the worst among the three models without hyperparameters. However, when hyperparameters were used, the SVM model outperformed the other two models improved with 98% both accuracy and f1-score.

Discussion

These results suggest that the SVM model may be the most suitable for this problem when hyperparameters are properly tuned. Further research and evaluation will be needed to confirm these findings and to explore other potential applications.

PORTOFOLIO

Thank You

nbilasals

www.linkedin.com/in/nbilasals