MPBA G513 – PREDICTIVE ANALYTICS
PROJECT REPORT

MBA BUSINESS ANALYTICS
SECOND SEMESTER 2024-25

TOPIC - EMPLOYEE ATTRITION PREDICTION

SUPERVISOR: PROF. SATANIK MITRA

DATE: 18th APRIL 2025

SUBMITTED BY-

1. NIRJA RAJEEV – 2024H1540801P
2. KARTIK VIJAY BADKAS - 2024H1540809P
3. SATHEESH M K - 2024H1540810P
4. SARVESH KULKARNI - 2024H1540820P
5. SAKET PITALE - 2024H1540836P

Table of Contents:

1. Abstract:

Employee attrition can have significant impacts on business productivity, morale, and costs. This project aims to develop a predictive analytics model that can accurately forecast which employees are at risk of leaving the organization using historical HR data. Machine learning models such as Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine are applied and evaluated.

2. Introduction:

Employee turnover is one of the critical challenges faced by organizations. Proactively identifying potential attrition cases allows HR departments to design retention strategies that improve employee satisfaction and organizational performance. This study explores predictive analytics to gain insights into factors influencing attrition and develop robust prediction models.

3. Problem Statement:

Predict which employees are likely to leave the company using historical HR data.

4. Objectives:

- To explore the dataset and understand attrition trends
- To preprocess and prepare data for machine learning models
- To train and evaluate multiple models for predicting attrition
- To identify key features affecting employee attrition
- To visualize insights and model performance

5. Methodology

Data Description-

The dataset contains 1470 employee records with 35 features, including demographic, educational, professional, and compensation-related attributes.

Target Variable: Attrition (Yes/No)

Data Loading and Initial Exploration-

```
[1]: import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder, StandardScaler
     from sklearn.svm import SVC
     from sklearn.metrics import classification_report, accuracy_score,confusion_matrix,auc,roc_curve
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.linear_model import LogisticRegression
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[2]: # Load the dataset
     file_path = "C:/Users/Saket/OneDrive/Desktop/Predictive Project/Employee-Attrition.csv"
     df = pd.read_csv(file_path)
     # Display basic information and the first few rows
     df.info(), df.head()
```

Data Preprocessing

```
[3]: df_cleaned = df.drop(['EmployeeNumber', 'EmployeeCount', 'Over18', 'StandardHours'], axis=1)
     # Encode categorical variables
     for column in df_cleaned.select_dtypes(include=['object']).columns:
         le = LabelEncoder()
         df_cleaned[column] = le.fit_transform(df_cleaned[column])
```

Feature Scaling and Train-Test Split

```
[6]: # Split data into features and target variable
     X = df.drop(columns=["Attrition"])  # Features
     y = df["Attrition"]  # Target
```

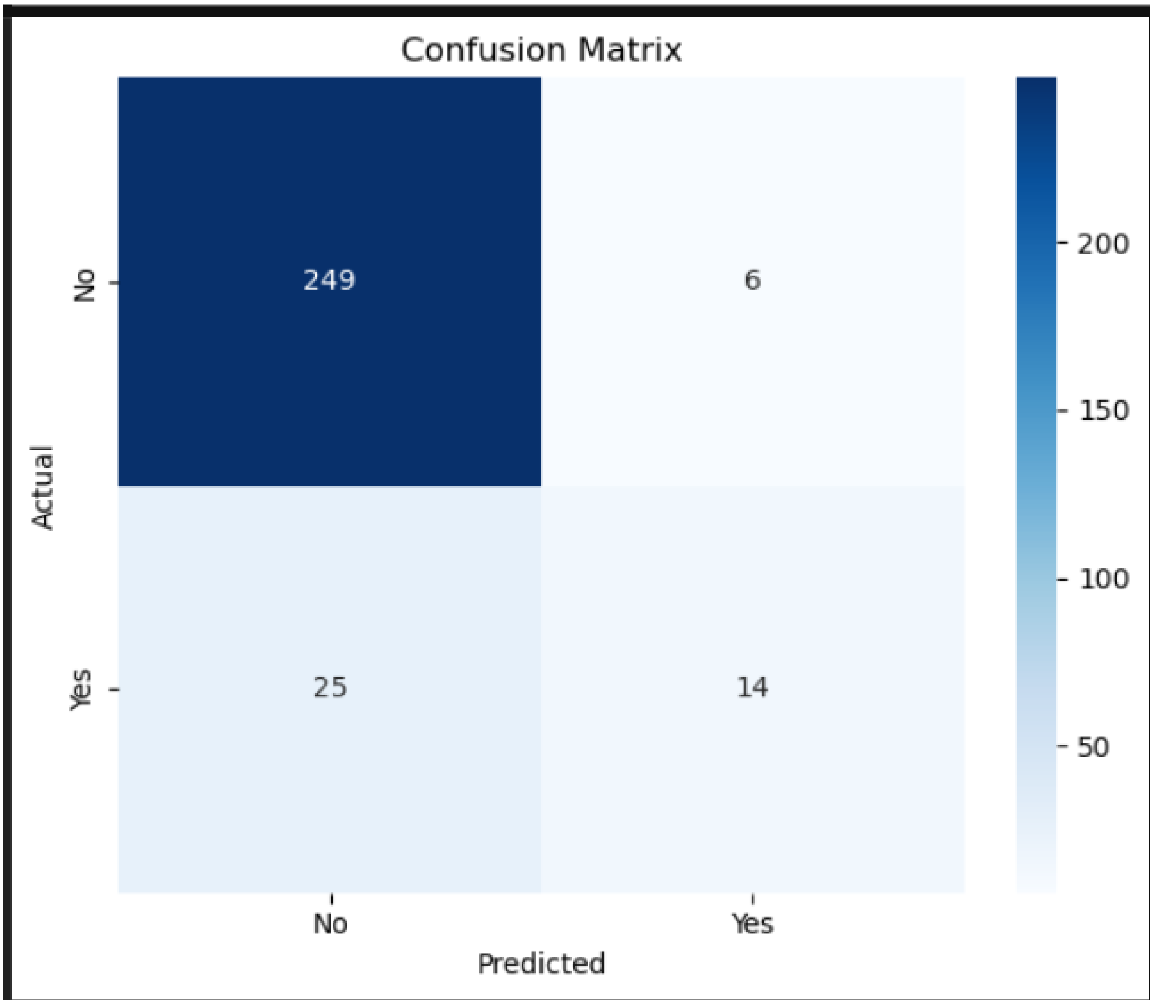
```
[7]: # Normalize numerical features
     scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)

     # Split into train and test sets
     X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

6. Model Evaluation:

Logistic Regression

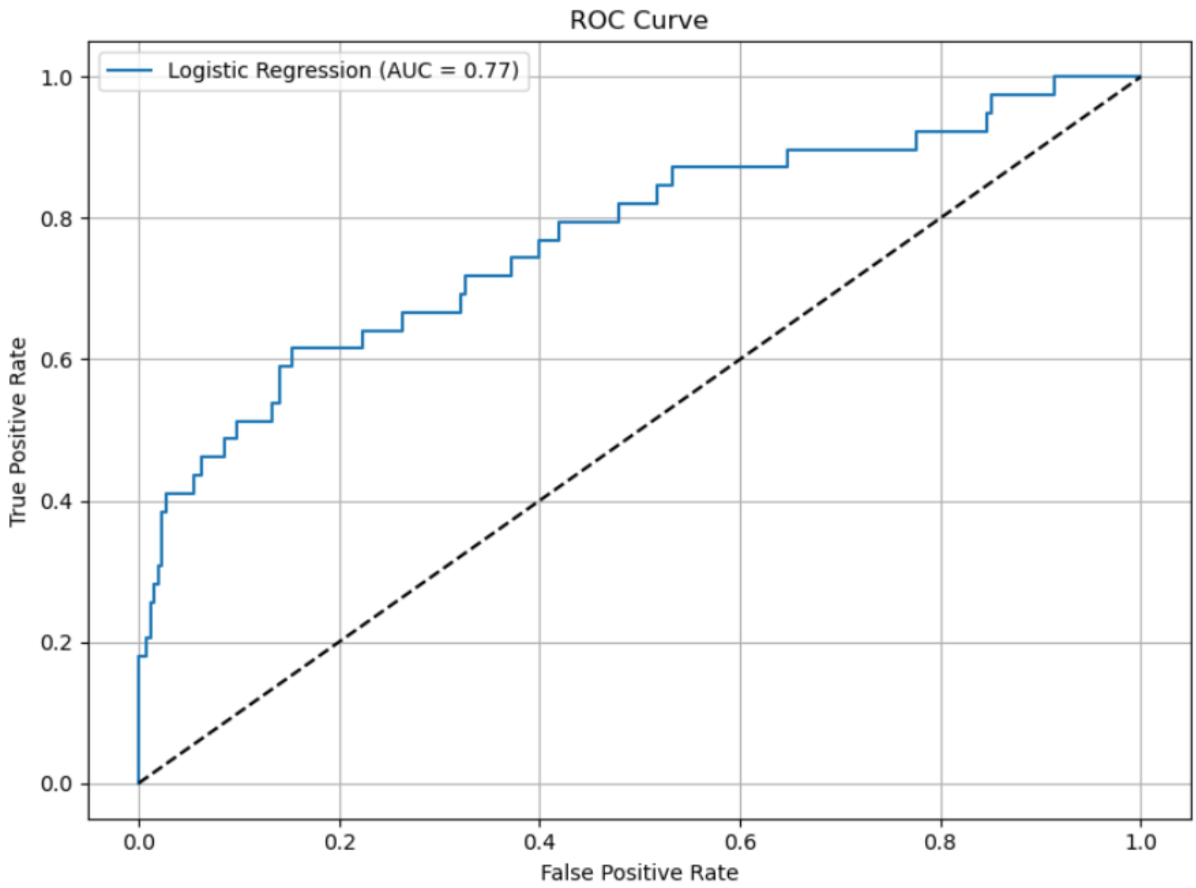
```
[8]: # Fit logistic regression model
     log_model = LogisticRegression(max_iter=2000)
     log_model.fit(X_train, y_train)

     # Predictions
     y_pred = log_model.predict(X_test)
     y_prob = log_model.predict_proba(X_test)[:, 1]
```

Confusion Matrix




```
              precision    recall  f1-score   support

          No       0.91      0.98      0.94       255
         Yes       0.70      0.36      0.47        39

    accuracy                           0.89       294
   macro avg       0.80      0.67      0.71       294
weighted avg       0.88      0.89      0.88       294
```

ROC Curve



Decision Tree

```
[13]: # Train Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Predictions
y_pred_dt = dt_model.predict(X_test)

# Evaluate model
accuracy_dt = accuracy_score(y_test, y_pred_dt)
report_dt = classification_report(y_test, y_pred_dt)

accuracy_dt, report_dt

print(accuracy_dt)
print(report_dt)
```

```
0.7993197278911565
              precision    recall  f1-score   support

           0       0.88      0.89      0.88       255
           1       0.24      0.23      0.23        39

    accuracy                           0.80       294
   macro avg       0.56      0.56      0.56       294
weighted avg       0.80      0.80      0.80       294
```

Random Forest

```
[14]: # Train Random Forest model
      rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
      rf_model.fit(X_train, y_train)

      # Predictions
      y_pred = rf_model.predict(X_test)
```

```
[15]: # Evaluate model
      accuracy = accuracy_score(y_test, y_pred)
      report = classification_report(y_test, y_pred)

      accuracy, report

      print(accuracy)
      print(report)
```

```
0.8809523809523809
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       255
           1       0.83      0.13      0.22        39

    accuracy                           0.88       294
   macro avg       0.86      0.56      0.58       294
weighted avg       0.88      0.88      0.84       294
```

Support Vector Machine

```
[11]: # Train SVM model
      svm_model = SVC(kernel="rbf", random_state=42)
      svm_model.fit(X_train, y_train)

      # Predictions
      y_pred = svm_model.predict(X_test)
```

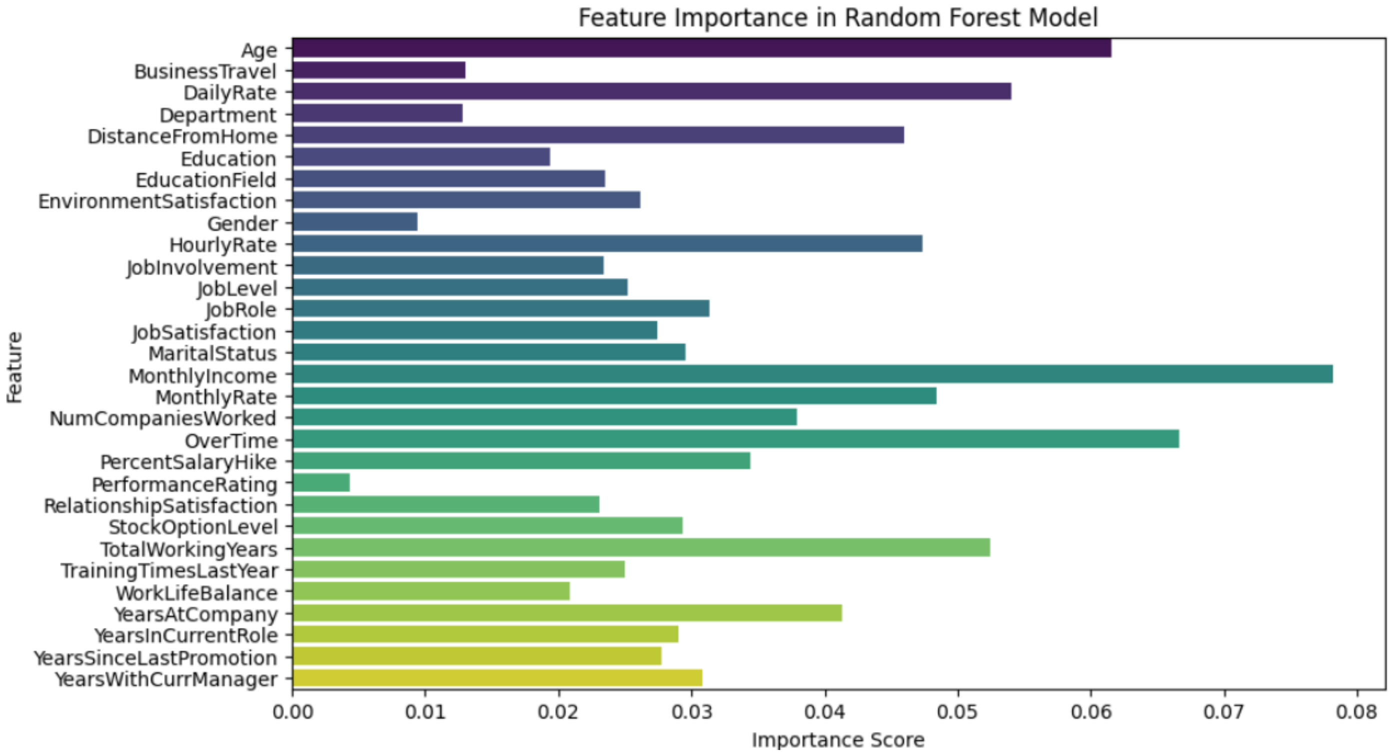
```
[12]: # Evaluate model
      accuracy = accuracy_score(y_test, y_pred)
      report = classification_report(y_test, y_pred)

      accuracy, report

      print(report)
```

```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94       255
           1       1.00      0.15      0.27        39

    accuracy                           0.89       294
   macro avg       0.94      0.58      0.60       294
weighted avg       0.90      0.89      0.85       294
```

7. Results and Discussion:

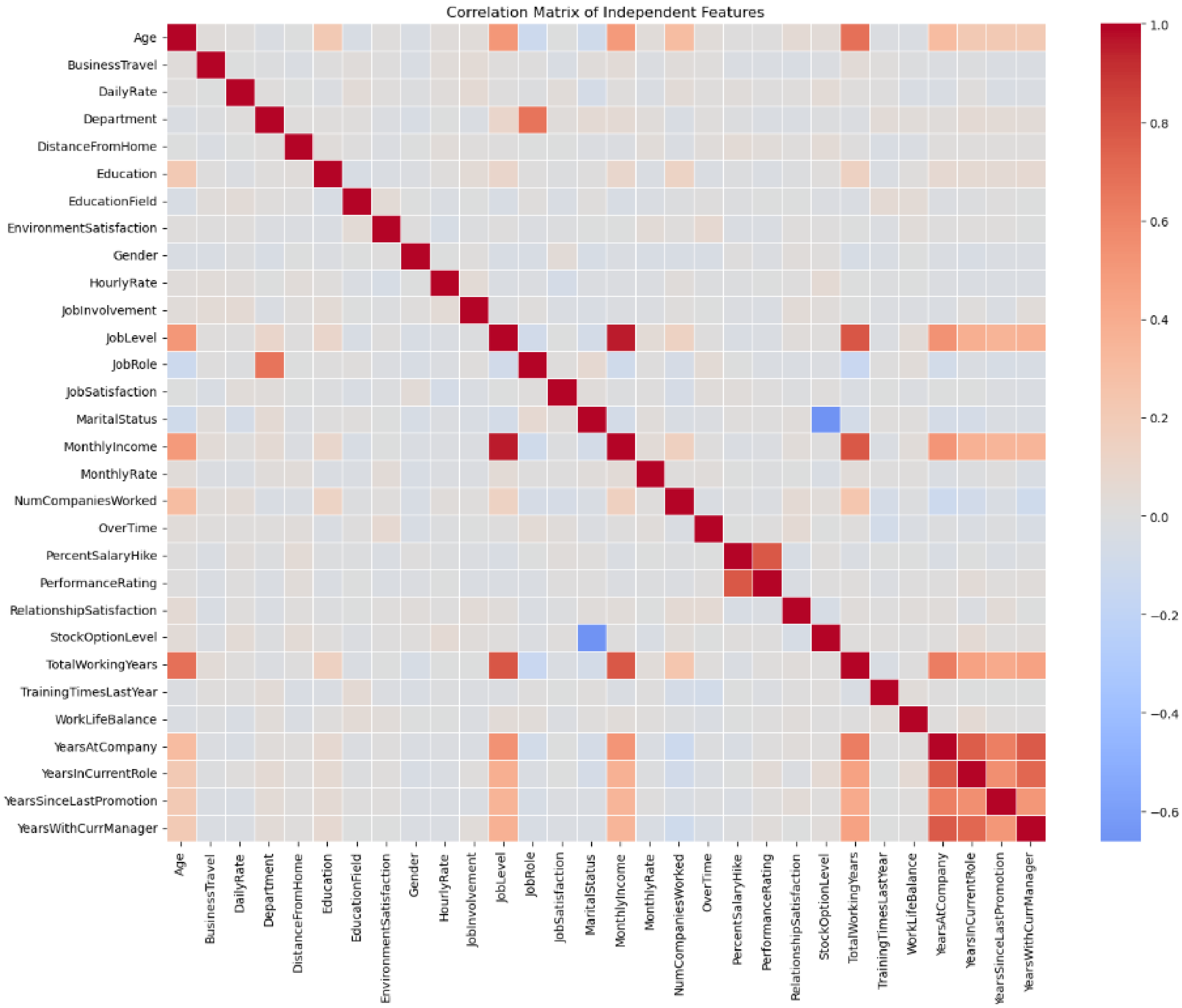
Feature Importance

```
[ ]: # Feature Importance Plot
feature_importances = rf_model.feature_importances_
features = X.columns

plt.figure(figsize=(10, 6))
sns.barplot(x=feature_importances, y=features, palette="viridis")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.title("Feature Importance in Random Forest Model")
plt.show()
```



Correlation Heatmap

```
[4]: C = df_cleaned.drop("Attrition", axis=1)

# Compute the correlation matrix
corr_matrix = C.corr()

# Plot the heatmap
plt.figure(figsize=(16, 12))
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm', center=0, linewidths=0.5)
plt.title("Correlation Matrix of Independent Features")
plt.show()
```



8. Conclusion:

The project successfully built and evaluated predictive models for employee attrition. Random Forest and SVM showed strong performance. Insights derived from feature importance can help HR professionals design effective retention strategies.

9. Future Scope

- Incorporate external data like employee satisfaction surveys
- Use deep learning models for potentially higher accuracy
- Perform real-time attrition monitoring in production systems

10. Appendix

- Dataset Size: 1470 records
- Key Features: Age, OverTime, MonthlyIncome, etc.
- Tools Used: Python, pandas, scikit-learn, seaborn, matplotlib