Project Title: IBM HR Analytics - Employee Attrition & Performance

Tools: Google Colab, Github

Technologies: Machine Learning, Data Analytics

Domain: Human Resources

---

1. Project Overview

The IBM HR Analytics Employee Attrition & Performance project aims to analyze employee attrition and performance patterns using machine learning. The goal is to identify key factors influencing attrition, predict employee churn, and provide actionable insights for HR decision-making.

2. Understanding the Problem

- High employee attrition can lead to increased hiring costs and loss of organizational knowledge.
- Understanding factors contributing to attrition can help businesses improve retention strategies.
- Predicting attrition enables HR teams to proactively address potential issues.

3. Dataset Preparation

- **Data Source:** IBM HR Analytics dataset
- **Features:** Employee demographics, job role, work environment, satisfaction levels, salary details, promotions, overtime, and performance ratings.
- **Target Variable:** 'Attrition' (Yes/No - indicating whether an employee left the company or stayed).

4. Data Exploration and Visualization

- **Attrition Distribution:** Understanding the number of employees who left vs. stayed.
- **Feature Analysis:** Boxplots and countplots for key variables like Age, Monthly Income, and Job Satisfaction.
- **Correlation Matrix:** Identifying relationships between features.

5. Data Preprocessing

- Handle missing values (if any).
- Encode categorical variables (e.g., department, education field, marital status).
- Standardize numerical features for better model performance.

6. Feature Engineering

- Identify the most important factors contributing to attrition.

- Create additional features that may enhance model performance.
- Feature selection to reduce dimensionality and improve efficiency.

7. Model Selection and Training

- **Algorithms Used:**
    - Decision Trees
    - Random Forest (with Hyperparameter Tuning using GridSearchCV)
    - Gradient Boosting (optional for comparison)
- Train-test split (80% training, 20% testing).

8. Model Evaluation

- **Accuracy Score:** Measures how well the model predicts attrition.
- **Classification Report:** Precision, recall, and F1-score.
- **Confusion Matrix:** Evaluates correct and incorrect predictions.
- **Feature Importance Visualization:** Identifies top contributing factors.

9. Attrition Prediction Implementation

- Develop a function to predict whether a new employee is likely to leave.
- Input employee details and return a prediction based on the trained model.

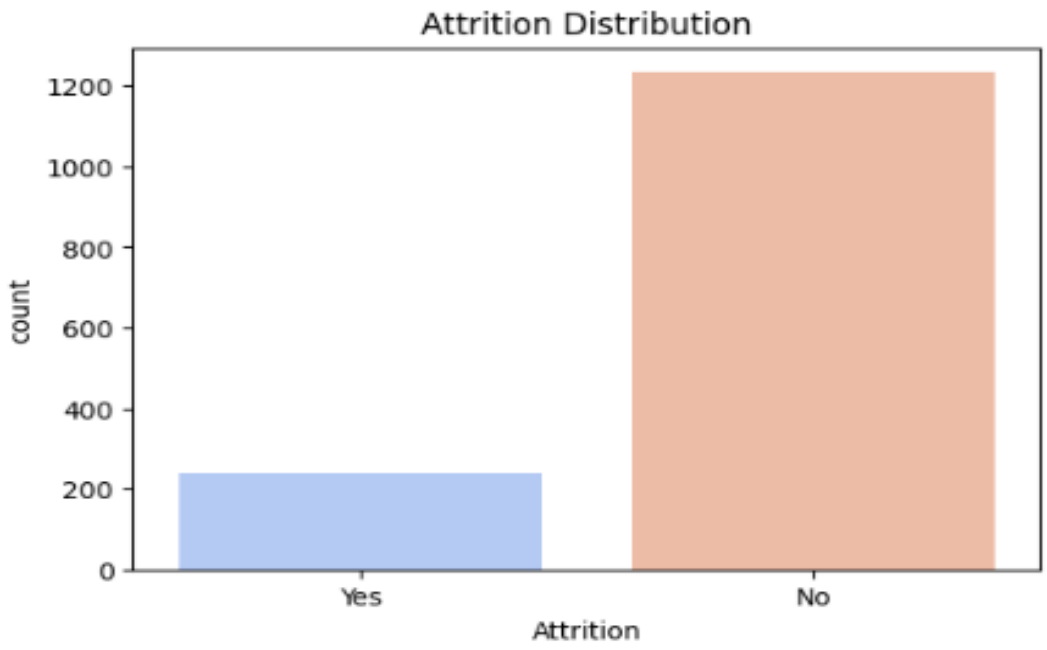
10. Deployment (Optional)

- Create an interactive web app using Flask or Streamlit for HR teams to input employee details and get predictions.

Code Snippet:

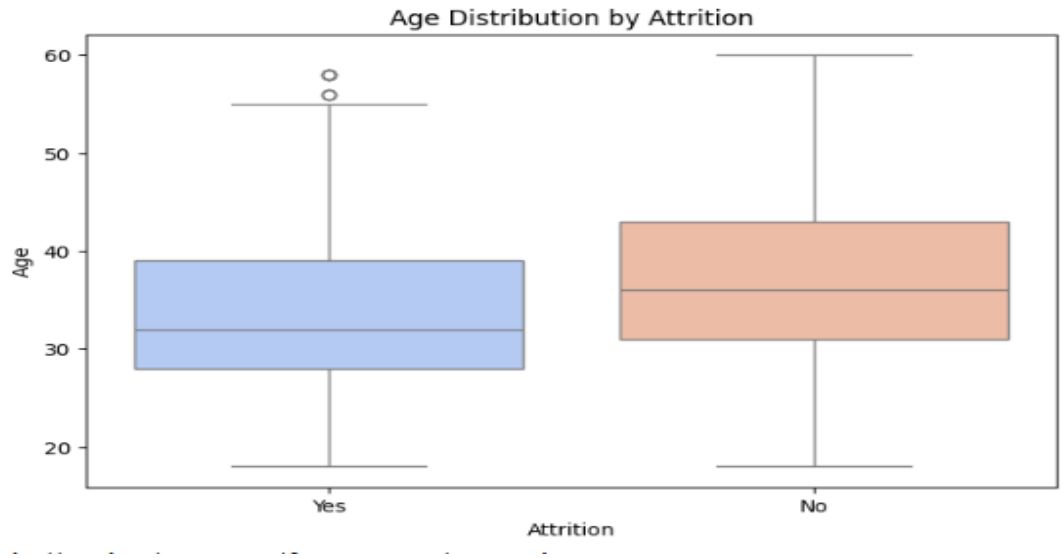
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
# Load the dataset
df = pd.read_csv("IBM Employee Attrition Data.csv")

# Display basic info
print("Dataset Info:\n")
df.info()
print("\nFirst 5 Rows:\n", df.head())

# Check for missing values
print("\nMissing Values:\n", df.isnull().sum())

# Summary statistics
print("\nSummary Statistics:\n", df.describe())

# Visualizing attrition distribution
plt.figure(figsize=(6, 4))
sns.countplot(x='Attrition', data=df, palette='coolwarm')
plt.title("Attrition Distribution")
plt.show()
```

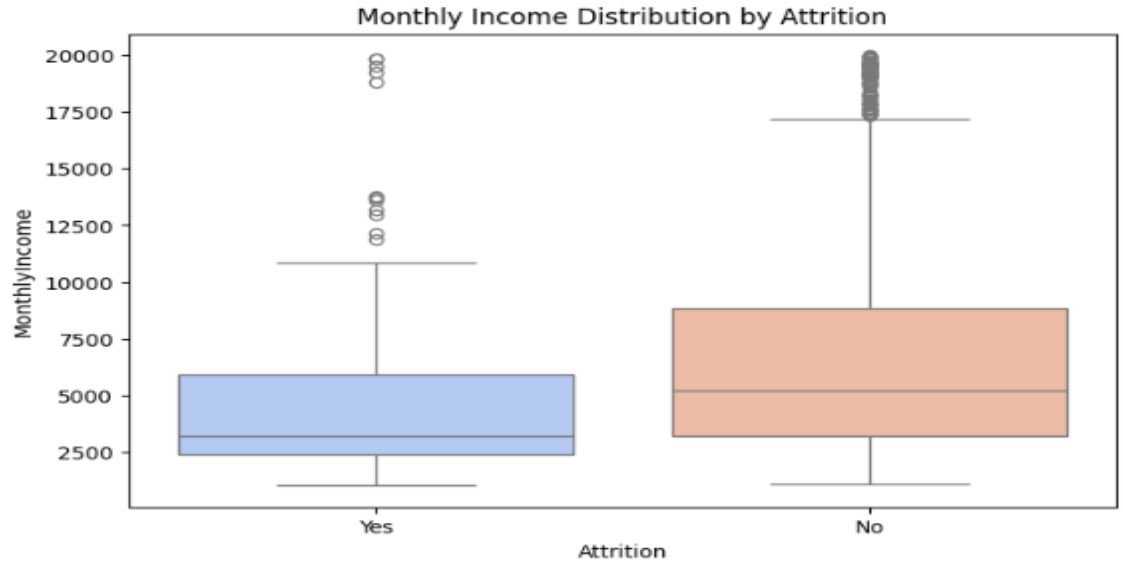


Additional Visualizations

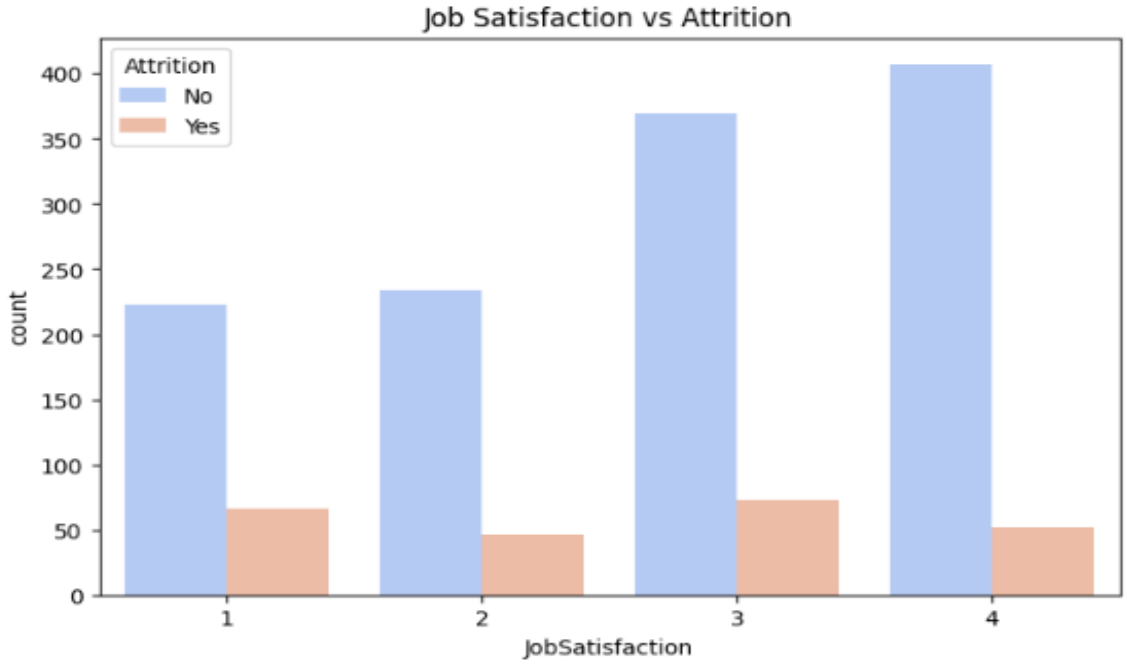
```
plt.figure(figsize=(8, 5))
sns.boxplot(x='Attrition', y='Age', data=df, palette='coolwarm')
plt.title("Age Distribution by Attrition")
plt.show()
```




```
plt.figure(figsize=(8, 5))
sns.boxplot(x='Attrition', y='MonthlyIncome', data=df, palette='coolwarm')
plt.title("Monthly Income Distribution by Attrition")
plt.show()
```



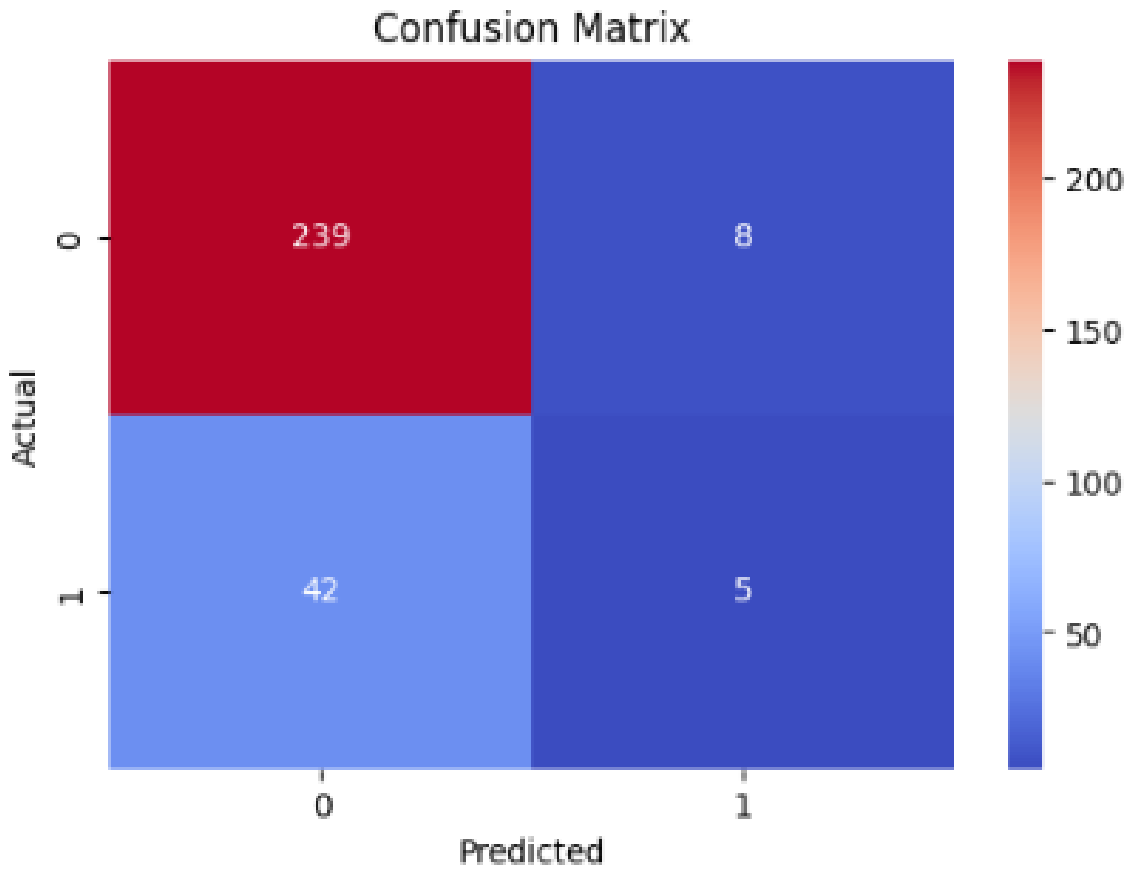
```
plt.figure(figsize=(8, 5))
sns.countplot(x='JobSatisfaction', hue='Attrition', data=df, palette='coolwarm')
plt.title("Job Satisfaction vs Attrition")
plt.show()
```



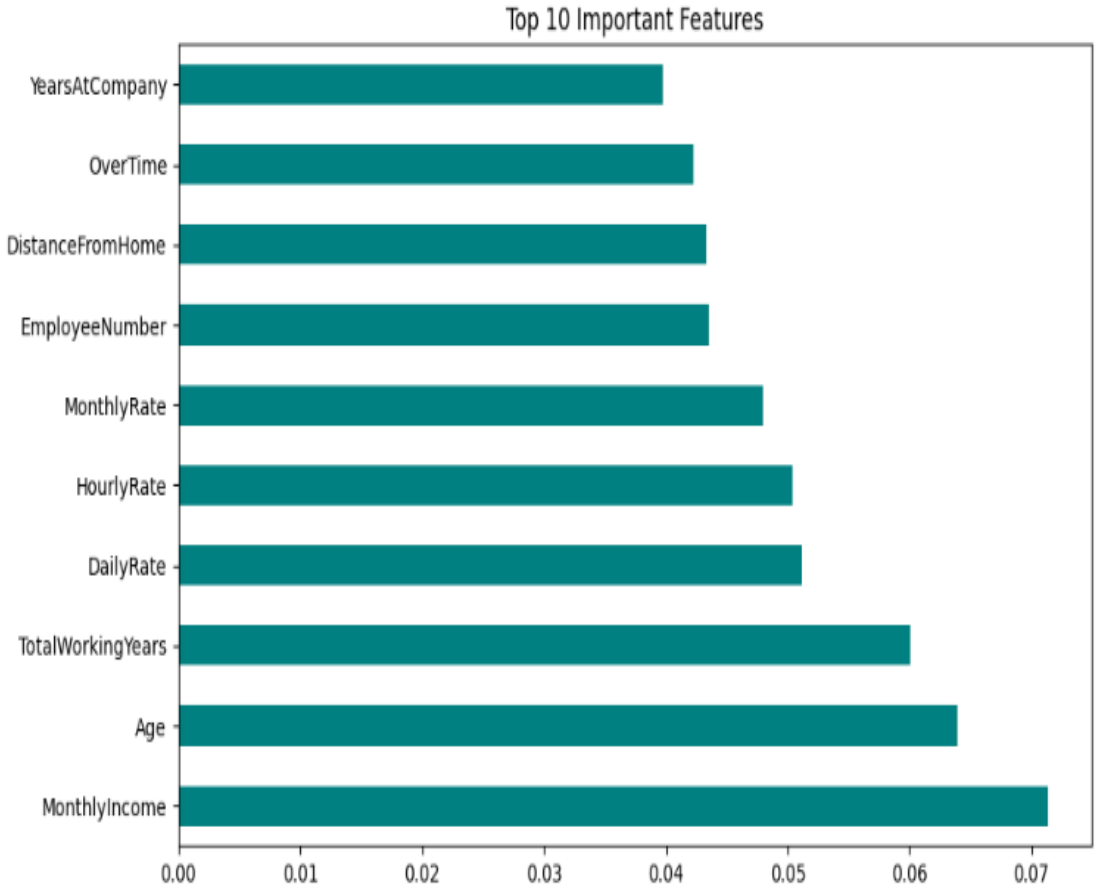

```
# Encoding categorical features
label_encoders = {}
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Splitting data into features and target
X = df.drop(columns=['Attrition'])  # Assuming 'Attrition' is the target variable
y = df['Attrition']
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Hyperparameter tuning for RandomForestClassifier
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best model
model = grid_search.best_estimator_

# Model training
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Model evaluation
accuracy = accuracy_score(y_test, y_pred)
print("\nModel Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
# Confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='coolwarm')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```




```
# Feature Importance
plt.figure(figsize=(10, 6))
feature_importances = pd.Series(model.feature_importances_, index=X.columns)
feature_importances.nlargest(10).plot(kind='barh', color='teal')
plt.title("Top 10 Important Features")
plt.show()
```




```
# Example function to predict attrition for a new employee
def predict_attrition(employee_data):
    employee_df = pd.DataFrame([employee_data], columns=X.columns)
    employee_df = scaler.transform(employee_df)
    prediction = model.predict(employee_df)
    return "Likely to Leave" if prediction[0] == 1 else "Likely to Stay"

# Example Usage
sample_employee = X_test[0]  # Using a sample from the test set
prediction = predict_attrition(sample_employee)
print("\nAttrition Prediction for Sample Employee:\n", prediction)
```

---

11. Results and Insights

- **Key Findings:** Major factors influencing attrition (e.g., job satisfaction, salary, overtime, work-life balance).
- **Recommendations:** Strategies for reducing attrition based on data insights.
- **Model Performance:** High accuracy and explainable results.

12. Future Scope

- Incorporate deep learning models for improved prediction.
- Implement real-time attrition monitoring with dynamic dashboards.
- Expand dataset to include industry-specific attrition factors.

13. References

- Research papers on employee attrition analysis.
- Machine learning algorithms and best practices.
- Data sources and preprocessing techniques.