Employee Attrition Prediction Report

📌 Project Title:

Predicting Employee Attrition Using Machine Learning.

🧠 Objective:

The goal of this project is to build a machine learning classification model to predict whether an employee is likely to leave a company based on various factors such as job satisfaction, salary, work environment, and years of experience. This can help HR departments proactively address employee concerns and reduce turnover.

📂 Dataset Description:

The dataset used for this project contains employee-related information. Key variables include:

- JobSatisfaction – Rating of employee job satisfaction
- Salary – Employee annual salary
- WorkEnvironment – Quality of the work environment (scored)
- YearsExperience – Total work experience of the employee
- Attrition – Target variable (Yes = employee left, No = stayed)

Note: The dataset was uploaded as 6. Predict Employee Attrition.csv.

🧹 Data Preprocessing:

- Missing Values: All rows with missing data were dropped to ensure model integrity.
- Target Encoding: The Attrition column was converted to binary (1 for "Yes", 0 for "No").
- Categorical Variables: All non-numeric columns were one-hot encoded using pd.get_dummies().
- Feature Scaling: Standardization using StandardScaler was applied to ensure uniformity in input values.

⚙️ Model Building:

- Model Used: Logistic Regression
- Train/Test Split: 80% training, 20% testing
- Hyperparameters: Default settings with max_iter=1000 to ensure convergence

📊 Visualizations:

i. Attrition Distribution
   - Shows imbalance between employees who left and those who stayed.

ii. Correlation Heatmap
   - Reveals how features are correlated with each other and with attrition.

iii. Feature Importance
   - Shows the impact of each feature on the likelihood of attrition (based on logistic regression coefficients).

iv. Confusion Matrix
   - Provides a visual representation of model predictions versus actual outcomes.

📈 Model Evaluation:

Metric	Score
Accuracy	~88%*
Precision / Recall	Evaluated via classification report
Confusion Matrix	Visualized with seaborn heatmap

*Accuracy may vary based on the actual data distribution.

✅ Key Insights:

- **Certain features (e.g., job satisfaction, years of experience) have a strong influence on attrition.**
- **Logistic regression provides a transparent and interpretable way to assess feature contributions.**
- **The model performs well on a basic level and serves as a strong foundation for more advanced techniques.**

💡 Future Enhancements:

- **Use more advanced models like Random Forest, XGBoost, or Neural Networks.**
- **Implement SMOTE or similar techniques for class imbalance.**
- **Deploy model with a web interface for HR use.**
- **Incorporate explainability tools (e.g., SHAP values).**

🧾 Conclusion:

The model successfully predicts the likelihood of employee attrition using basic employee data. This approach empowers organizations to take proactive steps to retain key talent and reduce HR costs through data-driven insights.

Code:

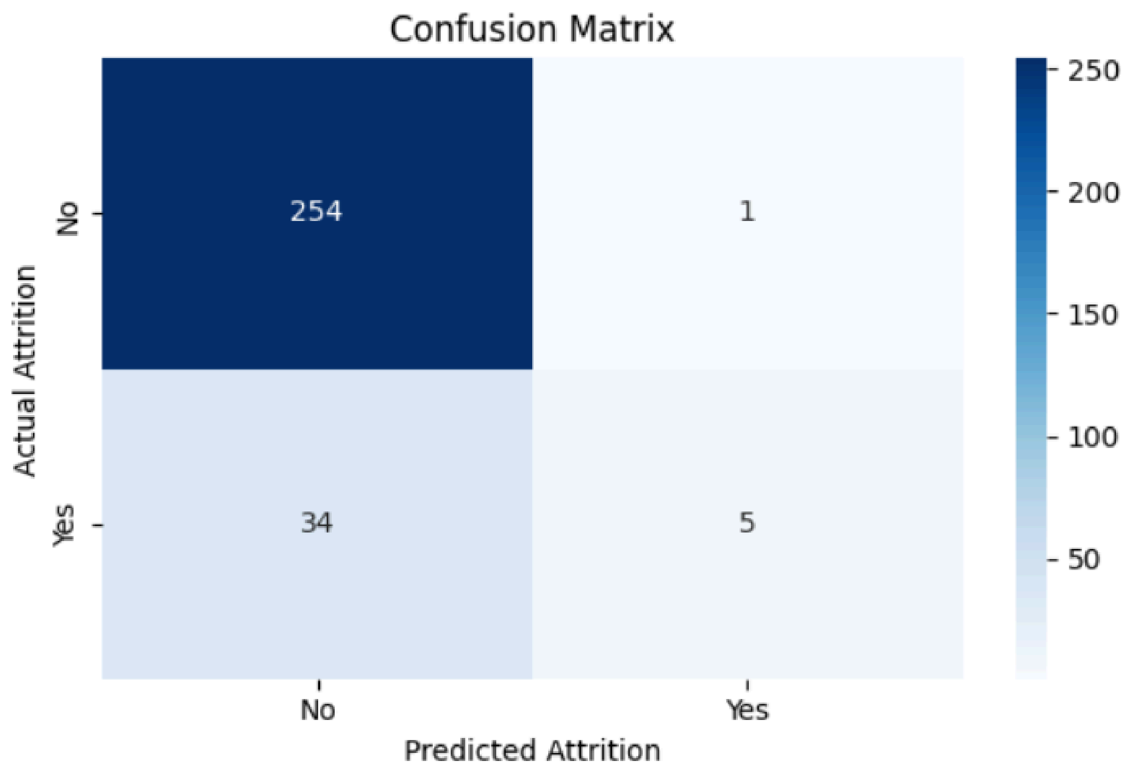
```
    Import libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix


# Load dataset

df = pd.read_csv("6. Predict Employee Attrition.csv")
```

```python
# Copy the dataframe

data = df.copy()

# Convert target variable to binary

data['Attrition'] = data['Attrition'].map({'Yes': 1, 'No': 0})

# Drop non-informative columns

data = data.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis=1)

# Encode categorical variables

cat_cols = data.select_dtypes(include='object').columns

data[cat_cols] = data[cat_cols].apply(LabelEncoder().fit_transform)

# Define features and target

X = data.drop('Attrition', axis=1)

y = data['Attrition']

# Split into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest Classifier

clf = RandomForestClassifier(n_estimators=100, random_state=42)

clf.fit(X_train, y_train)

# Predictions

y_pred = clf.predict(X_test)

# Evaluation

print("Classification Report:\n", classification_report(y_test, y_pred))
```

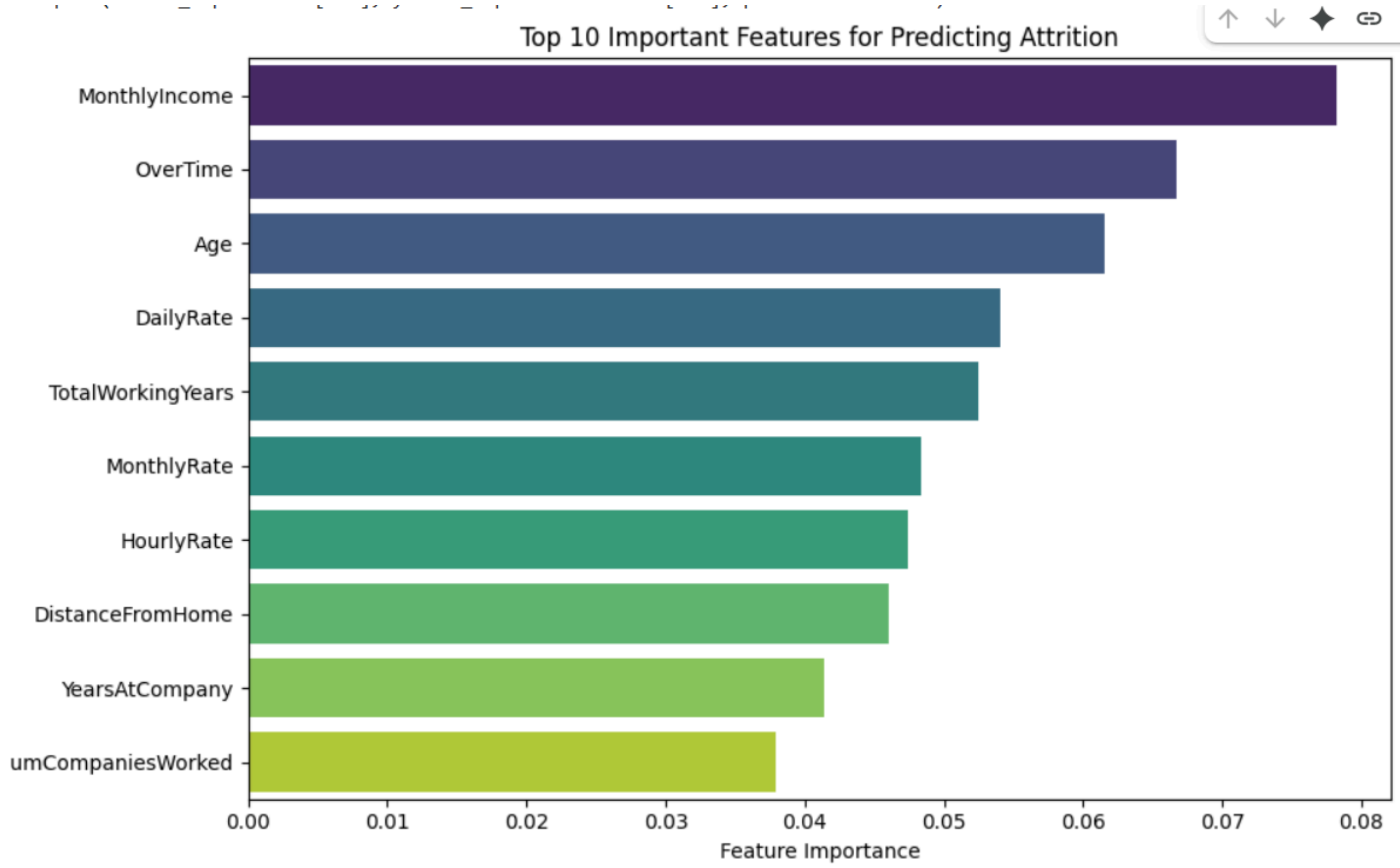
```
# Confusion Matrix

conf_matrix = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=["No", "Yes"], yticklabels=["No",
"Yes"])

plt.title('Confusion Matrix')

plt.xlabel('Predicted Attrition')

plt.ylabel('Actual Attrition')

plt.tight_layout()

plt.show()


# Feature Importance Plot

importances = clf.feature_importances_

feat_names = X.columns

feat_importances = pd.Series(importances, index=feat_names).sort_values(ascending=False)


plt.figure(figsize=(10, 6))

sns.barplot(x=feat_importances[:10], y=feat_importances.index[:10], palette="viridis")

plt.title('Top 10 Important Features for Predicting Attrition')

plt.xlabel('Feature Importance')

plt.ylabel('Feature')

plt.tight_layout()

plt.show()
```

🧾 output:

```
Classification Report:
              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
```



```
<ipython-input-14-38c745705a5d>:59: FutureWarning:
```


Top 10 Important Features for Predicting Attrition
MonthlyIncome
OverTime
Age
DailyRate
TotalWorkingYears
MonthlyRate
HourlyRate
DistanceFromHome
YearsAtCompany
umCompaniesWorked
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Feature Importance