

Machine Learning Project: HR Dataset Analysis

Student Name: Abdullah Hadi Alfaifi
Course: Master of Science in Artificial Intelligence
Date: April 17, 2026

```
In [1]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.feature_selection import RFE, SelectKBest, f_classif

In [2]: df = pd.read_csv('HRDataset_v14.csv')

In [3]: columns_to_drop = ['Employee_Name', 'EmpID', 'DateofHire', 'DateofTermination',
                          'TermReason', 'DOB', 'LastPerformanceReview_Date', 'ManagerName']

df_cleaned = df.drop(columns=[col for col in columns_to_drop if col in df.columns])

y = df_cleaned['Termd']
X = df_cleaned.drop(columns=['Termd'])

numeric_cols = X.select_dtypes(include=[np.number]).columns
categorical_cols = X.select_dtypes(exclude=[np.number]).columns

imputer_num = SimpleImputer(strategy='mean')
X[numeric_cols] = imputer_num.fit_transform(X[numeric_cols])

imputer_cat = SimpleImputer(strategy='most_frequent')
X[categorical_cols] = imputer_cat.fit_transform(X[categorical_cols])

le = LabelEncoder()
for col in categorical_cols:
    X[col] = le.fit_transform(X[col].astype(str))

In [4]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [None, 5, 10],
    'random_state': [42]
}

grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=3)
grid_search.fit(X_train, y_train)

print(f"--- Best Parameters found by GridSearch ---")
print(grid_search.best_params_)

--- Best Parameters found by GridSearch ---
{'max_depth': 5, 'n_estimators': 50, 'random_state': 42}

In [5]: filter_sel = SelectKBest(score_func=f_classif, k=5)
filter_sel.fit(X_train, y_train)
selected_filter = list(X.columns[filter_sel.get_support()])
print(f"\nSelected by Filter Method: {selected_filter}")

Selected by Filter Method: ['EmpStatusID', 'MaritalDesc', 'EmploymentStatus', 'ManagerID', 'RecruitmentSource']

In [7]: X_train_final = X_train[selected_wrapper]
X_test_final = X_test[selected_wrapper]

final_model = RandomForestClassifier(n_estimators=100, random_state=42)
final_model.fit(X_train_final, y_train)
y_pred = final_model.predict(X_test_final)

print("\n--- Final Model Performance Report ---")
print(classification_report(y_test, y_pred))

--- Final Model Performance Report ---
              precision    recall  f1-score   support

      0       1.00      1.00      1.00         41
      1       1.00      1.00      1.00         22

   accuracy          1.00
  macro avg          1.00
weighted avg          1.00

In [9]: import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))
sns.countplot(x='Termd', data=df)
plt.title('Distribution of Employee Termination (Target)')
plt.show()

Distribution of Employee Termination (Target)

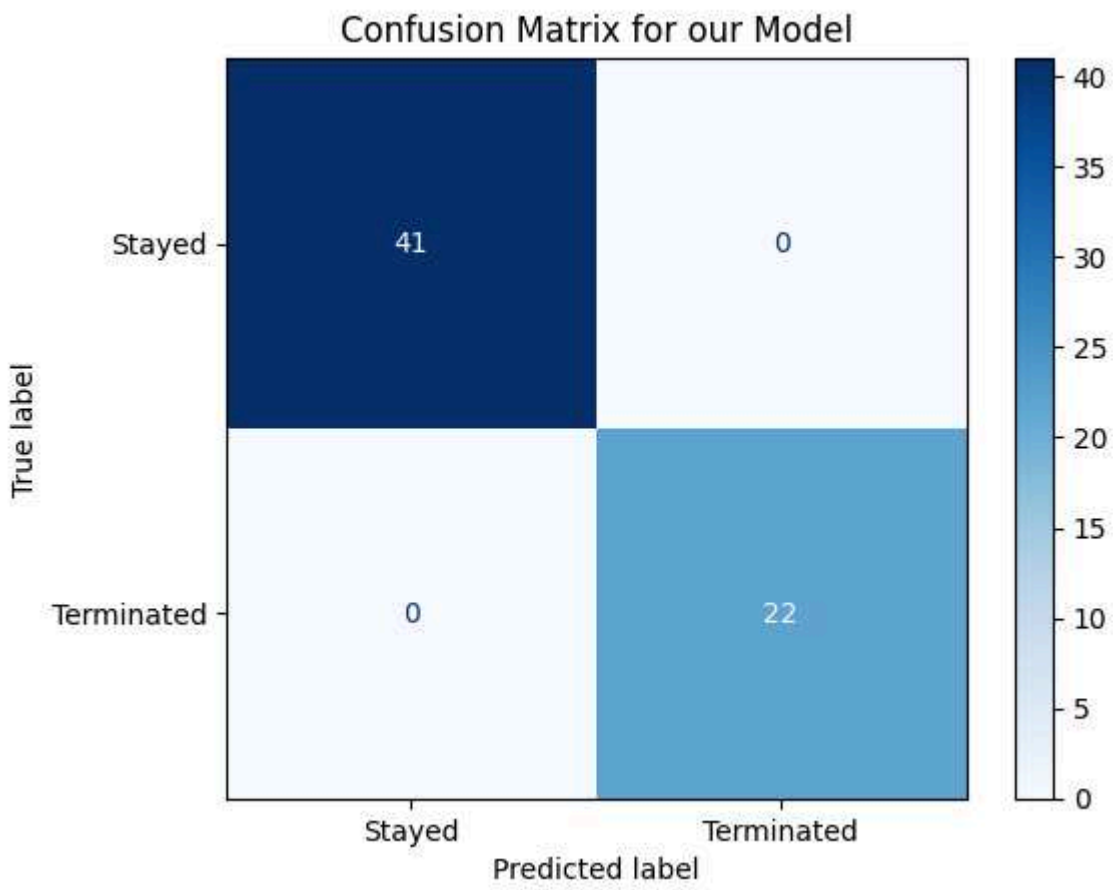
count
200
175
150
125
100
75
50
25
0
0      1
Termd
```

```
In [10]: from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Stayed', 'Terminated'])

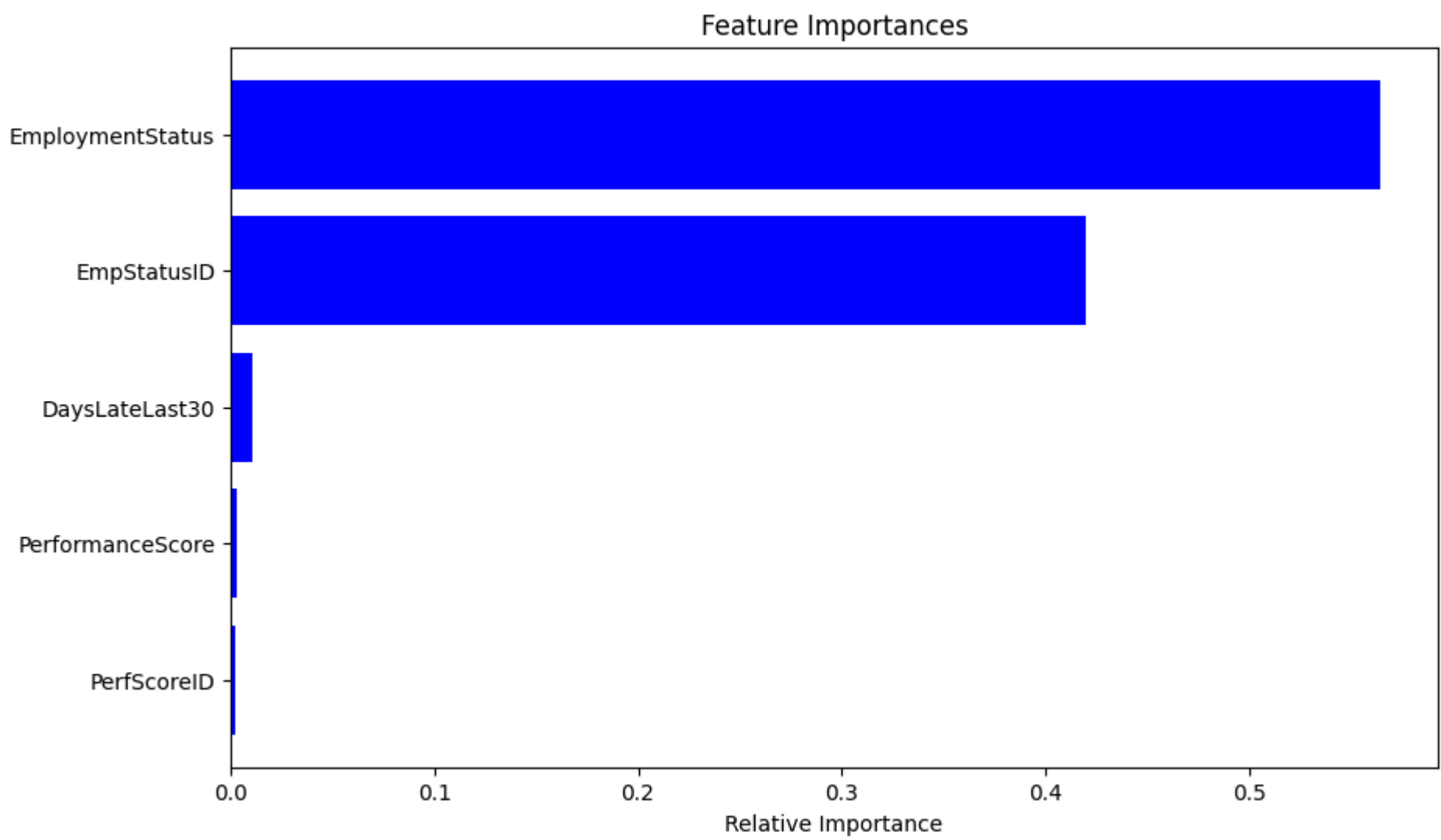
disp.plot(cmap='Blues')
```

```
plt.title('Confusion Matrix for our Model')
plt.show()
```



```
In [11]: importances = final_model.feature_importances_
indices = np.argsort(importances)

plt.figure(figsize=(10, 6))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [selected_wrapper[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



In []: