**AI Featuring Engineering Assignment**

Name: Krishna Kushwah

PRN: 202401110039

Batch: A2(AIML)

Dataset: IBM HR Analytics Attrition Dataset

```
# ----------------------------
# STEP 1: Install Kaggle API (only needed once per runtime)
# ----------------------------
!pip install -q kaggle
import os

# Directly store Kaggle API key (avoid manual upload every time)
os.environ['KAGGLE_USERNAME'] = "krishnakus"  # 🔹 Replace with your Kaggle username
os.environ['KAGGLE_KEY'] = "1c17326f6c19a146780799da2336ac06"    # 🔹 Replace with your actual Kaggle API key

# ----------------------------
# STEP 2: Download IBM HR Analytics dataset from Kaggle
# ----------------------------
!kaggle datasets download -d pavansubhasht/ibm-hr-analytics-attrition-dataset

# ----------------------------
# STEP 3: Unzip dataset file
# ----------------------------
!unzip -o ibm-hr-analytics-attrition-dataset.zip
```

```
Dataset URL: https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset
License(s): DbCL-1.0
ibm-hr-analytics-attrition-dataset.zip: Skipping, found more recently modified local copy (use --force to force download)
Archive:  ibm-hr-analytics-attrition-dataset.zip
  inflating: WA_Fn-UseC_-HR-Employee-Attrition.csv
```

```
# ----------------------------
# Import all required libraries
# ----------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
```

```
# ----------------------------
# Load the dataset into a DataFrame
# ----------------------------
df = pd.read_csv('/content/WA_Fn-UseC_-HR-Employee-Attrition.csv')

# Display first 5 rows
df.head()
```

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

5 rows × 35 columns

```python
# -----------------------------
# Check general information about the dataset
# -----------------------------
print(" 🔹 Basic Info:")
df.info()

print("\n 🔹 Dataset Shape:", df.shape)
print("\n 🔹 Missing Values per Column:")
print(df.isnull().sum())

print("\n 🔹 Duplicate Rows:", df.duplicated().sum())
```

```
 🔹 Basic Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB

 🔹 Dataset Shape: (1470, 35)

 🔹 Missing Values per Column:
Age                        0
Attrition                  0
BusinessTravel             0
DailyRate                  0
Department                 0
DistanceFromHome           0
Education                  0
EducationField             0
EmployeeCount              0
EmployeeNumber             0
EnvironmentSatisfaction    0
```

The dataset contains employee details for attrition prediction. It has no missing values or duplicate rows. Data types are mostly numeric, with some categorical fields like “Gender”, “JobRole”, and “MaritalStatus”.

```python
# -----------------------------
# Statistical Summary of all columns
# -----------------------------
df.describe(include='all').T
```

| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | ma |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Age** | 1470.0 | NaN | NaN | NaN | 36.92381 | 9.135373 | 18.0 | 30.0 | 36.0 | 43.0 | 60 |
| **Attrition** | 1470 | 2 | No | 1233 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **BusinessTravel** | 1470 | 3 | Travel_Rarely | 1043 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **DailyRate** | 1470.0 | NaN | NaN | NaN | 802.485714 | 403.5091 | 102.0 | 465.0 | 802.0 | 1157.0 | 1499 |
| **Department** | 1470 | 3 | Research & Development | 961 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **DistanceFromHome** | 1470.0 | NaN | NaN | NaN | 9.192517 | 8.106864 | 1.0 | 2.0 | 7.0 | 14.0 | 29 |
| **Education** | 1470.0 | NaN | NaN | NaN | 2.912925 | 1.024165 | 1.0 | 2.0 | 3.0 | 4.0 | 5 |
| **EducationField** | 1470 | 6 | Life Sciences | 606 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **EmployeeCount** | 1470.0 | NaN | NaN | NaN | 1.0 | 0.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1 |
| **EmployeeNumber** | 1470.0 | NaN | NaN | NaN | 1024.865306 | 602.024335 | 1.0 | 491.25 | 1020.5 | 1555.75 | 2068 |
| **EnvironmentSatisfaction** | 1470.0 | NaN | NaN | NaN | 2.721769 | 1.093082 | 1.0 | 2.0 | 3.0 | 4.0 | 4 |
| **Gender** | 1470 | 2 | Male | 882 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **HourlyRate** | 1470.0 | NaN | NaN | NaN | 65.891156 | 20.329428 | 30.0 | 48.0 | 66.0 | 83.75 | 100 |
| **JobInvolvement** | 1470.0 | NaN | NaN | NaN | 2.729932 | 0.711561 | 1.0 | 2.0 | 3.0 | 3.0 | 4 |
| **JobLevel** | 1470.0 | NaN | NaN | NaN | 2.063946 | 1.10694 | 1.0 | 1.0 | 2.0 | 3.0 | 5 |
| **JobRole** | 1470 | 9 | Sales Executive | 326 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **JobSatisfaction** | 1470.0 | NaN | NaN | NaN | 2.728571 | 1.102846 | 1.0 | 2.0 | 3.0 | 4.0 | 4 |
| **MaritalStatus** | 1470 | 3 | Married | 673 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **MonthlyIncome** | 1470.0 | NaN | NaN | NaN | 6502.931293 | 4707.956783 | 1009.0 | 2911.0 | 4919.0 | 8379.0 | 19999 |
| **MonthlyRate** | 1470.0 | NaN | NaN | NaN | 14313.103401 | 7117.786044 | 2094.0 | 8047.0 | 14235.5 | 20461.5 | 26999 |
| **NumCompaniesWorked** | 1470.0 | NaN | NaN | NaN | 2.693197 | 2.498009 | 0.0 | 1.0 | 2.0 | 4.0 | 9 |
| **Over18** | 1470 | 1 | Y | 1470 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **OverTime** | 1470 | 2 | No | 1054 | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| **PercentSalaryHike** | 1470.0 | NaN | NaN | NaN | 15.209524 | 3.659938 | 11.0 | 12.0 | 14.0 | 18.0 | 25 |
| **PerformanceRating** | 1470.0 | NaN | NaN | NaN | 3.153741 | 0.360824 | 3.0 | 3.0 | 3.0 | 3.0 | 4 |
| **RelationshipSatisfaction** | 1470.0 | NaN | NaN | NaN | 2.712245 | 1.081209 | 1.0 | 2.0 | 3.0 | 4.0 | 4 |
| **StandardHours** | 1470.0 | NaN | NaN | NaN | 80.0 | 0.0 | 80.0 | 80.0 | 80.0 | 80.0 | 80 |
| **StockOptionLevel** | 1470.0 | NaN | NaN | NaN | 0.793878 | 0.852077 | 0.0 | 0.0 | 1.0 | 1.0 | 3 |
| **TotalWorkingYears** | 1470.0 | NaN | NaN | NaN | 11.279592 | 7.780782 | 0.0 | 6.0 | 10.0 | 15.0 | 40 |
| **TrainingTimesLastYear** | 1470.0 | NaN | NaN | NaN | 2.79932 | 1.289271 | 0.0 | 2.0 | 3.0 | 3.0 | 6 |
| **WorkLifeBalance** | 1470.0 | NaN | NaN | NaN | 2.761224 | 0.706476 | 1.0 | 2.0 | 3.0 | 3.0 | 4 |
| **YearsAtCompany** | 1470.0 | NaN | NaN | NaN | 7.008163 | 6.126525 | 0.0 | 3.0 | 5.0 | 9.0 | 40 |
| **YearsInCurrentRole** | 1470.0 | NaN | NaN | NaN | 4.229252 | 3.623137 | 0.0 | 2.0 | 3.0 | 7.0 | 18 |
| **YearsSinceLastPromotion** | 1470.0 | NaN | NaN | NaN | 2.187755 | 3.22243 | 0.0 | 0.0 | 1.0 | 3.0 | 15 |

**Numerical features like Age, MonthlyIncome, and DistanceFromHome have wide ranges. Categorical features show limited distinct categories suitable for encoding.**

```
# ----------------------------
# Convert object columns to categorical type for efficiency
# ----------------------------
categorical_cols = df.select_dtypes(include='object').columns
for col in categorical_cols:
    df[col] = df[col].astype('category')

print("✅ Converted all object columns to categorical data types.")
print(df.dtypes)
```

```
✅ Converted all object columns to categorical data types.
Age                            int64
Attrition                   category
BusinessTravel              category
DailyRate                      int64
Department                  category
```

```
DistanceFromHome            int64
Education                   int64
EducationField           category
EmployeeCount               int64
EmployeeNumber              int64
EnvironmentSatisfaction     int64
Gender                   category
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobRole                  category
JobSatisfaction             int64
MaritalStatus            category
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
Over18                   category
OverTime                 category
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

**Converting to categorical helps with memory optimization and encoding operations later.**

```
# ----------------------------
# Apply Label Encoding to all categorical features
# ----------------------------
le = LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])

print("✅ Label Encoding Applied Successfully.")
df.head()
```

✅ Label Encoding Applied Successfully.

| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHome | Education | EducationField | EmployeeCount | Employ |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 41 | 1 | 2 | 1102 | 2 | 1 | 2 | 1 | 1 | |
| **1** | 49 | 0 | 1 | 279 | 1 | 8 | 1 | 1 | 1 | |
| **2** | 37 | 1 | 2 | 1373 | 1 | 2 | 2 | 4 | 1 | |
| **3** | 33 | 0 | 1 | 1392 | 1 | 3 | 4 | 1 | 1 | |
| **4** | 27 | 0 | 2 | 591 | 1 | 2 | 1 | 3 | 1 | |

5 rows × 35 columns

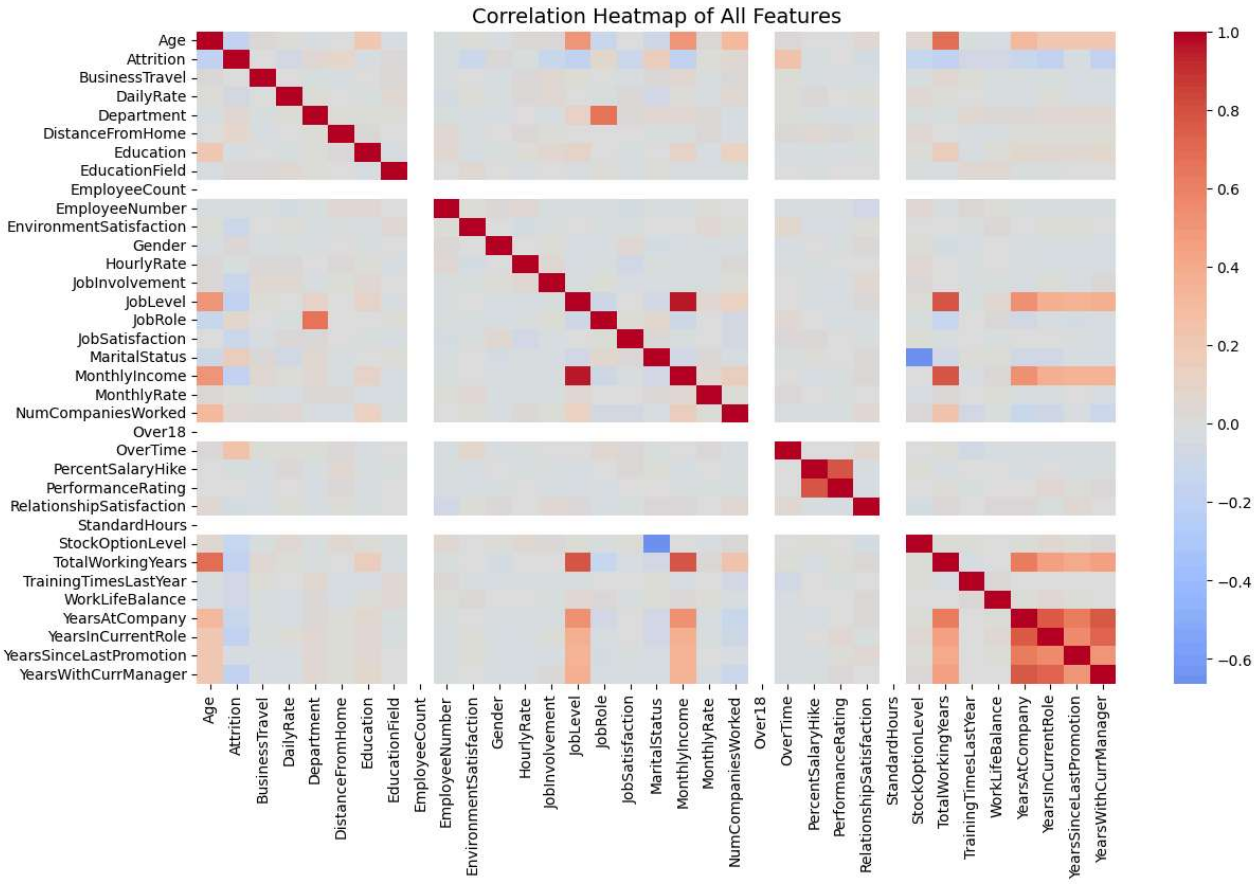
**Label encoding converts categories into integers, making them usable in numeric models. Example: Gender → 0 for Female, 1 for Male.**

```
# ----------------------------
# Standardize numeric columns for equal influence
# ----------------------------
num_cols = ['Age', 'MonthlyIncome', 'DistanceFromHome', 'YearsAtCompany', 'YearsInCurrentRole']
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])

print("✅ Scaling completed for numerical features.")
df[num_cols].head()
```

✅ Scaling completed for numerical features.

| | Age | MonthlyIncome | DistanceFromHome | YearsAtCompany | YearsInCurrentRole |
|---|---|---|---|---|---|
| 0 | 0.446350 | -0.108350 | -1.010909 | -0.164613 | -0.063296 |
| 1 | 1.322365 | -0.291719 | -0.147150 | 0.488508 | 0.764998 |
| 2 | 0.008343 | -0.937654 | -0.887515 | -1.144294 | -1.167687 |
| 3 | -0.429664 | -0.763634 | -0.764121 | 0.161947 | 0.764998 |
| 4 | -1.086676 | -0.644858 | -0.887515 | -0.817734 | -0.615492 |

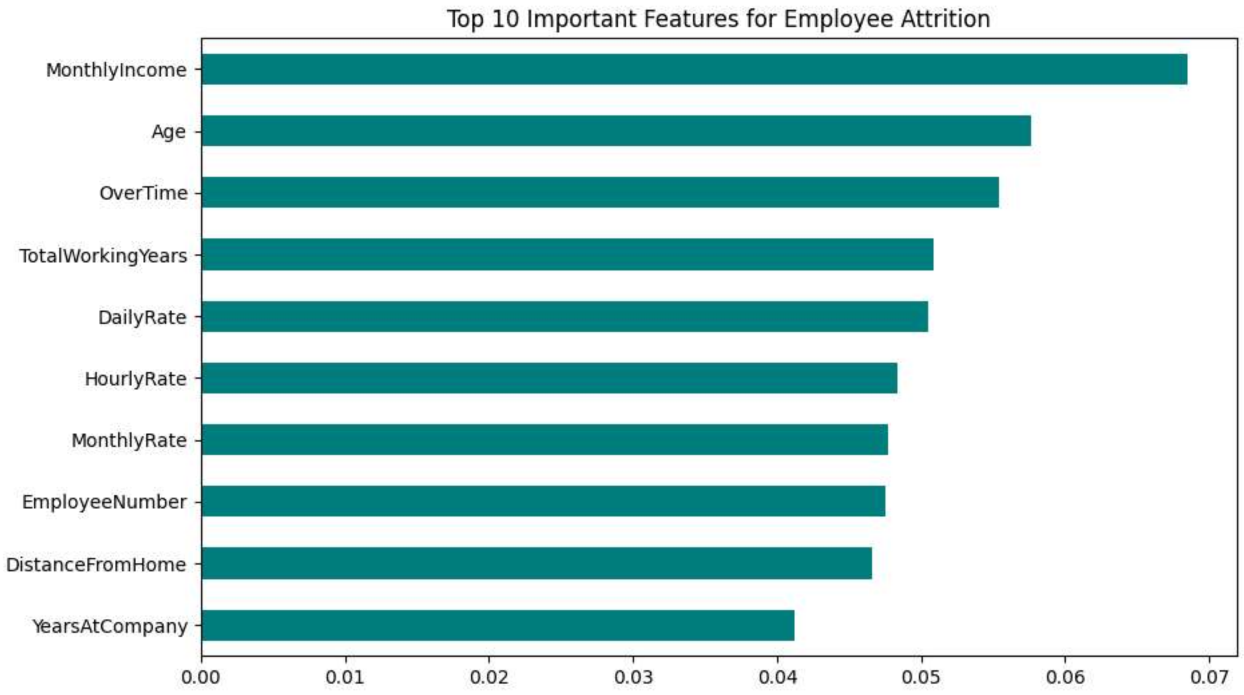
```
# ----------------------------
# Plot correlation heatmap
# ----------------------------
plt.figure(figsize=(14,8))
sns.heatmap(df.corr(), cmap='coolwarm', center=0)
plt.title('Correlation Heatmap of All Features', fontsize=14)
plt.show()
```




**From the heatmap, JobLevel and MonthlyIncome show high correlation. One of them can be dropped to reduce multicollinearity.**

```
# ----------------------------
# Feature importance using Random Forest
# ----------------------------
X = df.drop('Attrition', axis=1)
y = df['Attrition']

model = RandomForestClassifier(random_state=42)
model.fit(X, y)

# Get top 10 important features
```

```
feat_imp = pd.Series(model.feature_importances_, index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(10,6))
feat_imp.head(10).plot(kind='barh', color='teal')
plt.title('Top 10 Important Features for Employee Attrition')
plt.gca().invert_yaxis()
plt.show()

feat_imp.head(10)
```




| | 0 |
|---|---|
| **MonthlyIncome** | 0.068485 |
| **Age** | 0.057602 |
| **OverTime** | 0.055375 |
| **TotalWorkingYears** | 0.050839 |
| **DailyRate** | 0.050452 |
| **HourlyRate** | 0.048364 |
| **MonthlyRate** | 0.047666 |
| **EmployeeNumber** | 0.047490 |
| **DistanceFromHome** | 0.046551 |
| **YearsAtCompany** | 0.041176 |

**dtype:** float64

**The most important predictors include OverTime, MonthlyIncome, JobRole, and Age. *italicized text* These can be prioritized during model building or selection.**

```
# ----------------------------
# Final summary of cleaned and transformed dataset
# ----------------------------
print("✅ Final Dataset Shape:", df.shape)
print("✅ No Missing Values:", df.isnull().sum().sum() == 0)
print("✅ Dataset Ready for Model Building")

df.head()
```

```
✅ Final Dataset Shape: (1470, 35)
✅ No Missing Values: True
✅ Dataset Ready for Model Building
```

| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHome | Education | EducationField | EmployeeCount | E |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.446350 | 1 | 2 | 1102 | 2 | -1.010909 | 2 | 1 | 1 | |
| **1** | 1.322365 | 0 | 1 | 279 | 1 | -0.147150 | 1 | 1 | 1 | |
| **2** | 0.008343 | 1 | 2 | 1373 | 1 | -0.887515 | 2 | 4 | 1 | |
| **3** | -0.429664 | 0 | 1 | 1392 | 1 | -0.764121 | 4 | 1 | 1 | |
| **4** | -1.086676 | 0 | 2 | 591 | 1 | -0.887515 | 1 | 3 | 1 | |

5 rows × 35 columns

`**` **Summary of Feature Engineering** `**`

Dataset contained 1470 records and 35 columns.

No missing or duplicate values found.

Converted categorical data types for efficient memory use.

Applied Label Encoding and Standard Scaling to prepare features for modeling.