Data Science Project:

Employee Attrition Prediction

Project Documentation

Made by:

Ahmed Reda Abdullah	21035590
Khaled Ayman Fathy	21037539
Mahmoud Ehab Helmy	21112754
Ramez Remon Ramses	21077659
john mohsen melad	21035572

Table of Contents

1. Project Overview

This project establishes a comprehensive analytical and predictive system designed to anticipate employee turnover within an organization. Its workflow spans from initial data investigation to advanced modeling and a simulated operational environment for managing machine-learning assets.
Through this pipeline, the project aims to support HR leadership in identifying underlying patterns that contribute to resignations, helping organizations intervene earlier and cultivate a more stable workforce. Beyond producing predictions, the workflow demonstrates how such a model would be monitored, versioned, and moved through different operational stages, reflecting industry-standard machine learning operations practices.

2. Dataset Description

The dataset contains a wide spectrum of information representing employees' personal backgrounds, career progression, workplace experiences, and behavioral indicators. These variables collectively portray how individual and organizational factors may influence turnover tendencies.
Key attributes include demographic characteristics, job assignments, career duration, satisfaction levels, work–life dynamics, travel obligations, and overtime demands. Each category provides a distinct lens through which attrition risks may manifest.

Key Feature Groups

- Age, Gender, JobRole, Department
- WorkLifeBalance, JobSatisfaction
- YearsAtCompany, TotalWorkingYears
- BusinessTravel intensity
- Regularity of OverTime
- DistanceFromHome
- **Attrition (Yes/No)** — the core target for prediction

The dataset is sourced from a CSV file and processed directly within the notebook, ensuring reproducibility and analytical consistency.

2. Dataset Description

2.1 Loading Dataset

Code (Cell 0–1):

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt


df = pd.read_csv("Employee-Attrition.csv")

df.shape
```

Important Output Summary

- Dataset loaded successfully
- Shape: **(1470 rows, 35 columns)**

2.2 Dataset Structure

Code (Cell 2):

```
df.info()
```

Important Output Summary

- 35 total columns
- No missing values reported
- Mix of numerical and categorical variables
- Approx. 9 categorical and 26 numeric fields

2.3 Statistical Summary

Code (Cell 3):

```
df.describe()
```

Important Output Summary

- Descriptive statistics generated (min, max, mean, SD)
- Highlights:
  - Age ranges from early 20s to 60s
  - MonthlyIncome varies widely
  - YearsAtCompany shows strong early-career representation

2.4 Sample Preview

Code (Cell 4):

```
df.head(10)
```

Important Output Summary

- Displays first 10 employee records
- Confirms dataset includes demographic, job, and performance fields

---

2.5 Duplicate Check

Code (Cell 5):

```
df.duplicated().sum()
```

Important Output Summary

- Duplicate rows: **0**
- Dataset is clean and ready for EDA

---

3. Data Preprocessing

To prepare the dataset for predictive analysis, several transformations are applied to strengthen data reliability and model readiness.

3.1 Handling Missing Values

Each column undergoes inspection to identify gaps. Where necessary, numerical inputs are imputed using summary statistics such as median or mean, while categorical fields are completed using frequency-based estimates. These adjustments reinforce the structural integrity of the dataset and minimize distortions during model training.

No explicit missing-value handling code appears later because:

- df.info() showed **0 missing values**
- No imputation was required

3.2 Encoding Categorical Variables

Categorical attributes are converted into numerical representations through one-hot encoding. The attrition variable itself is translated into binary form, enabling the classifiers to process it effectively. By turning qualitative factors into structured numeric fields, the dataset becomes universally interpretable across diverse modeling approaches.

Code (Cell 26 — Part 1):

```
X = df.drop(columns=['Attrition'])  # Features
y = df['Attrition']  # Target

numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
cat_features = X.select_dtypes(include=['object', 'bool']).columns

print ("Numeric Features: ", len(numeric_features))
print ("Categorical Features: ", len(cat_features))

from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='minority', random_state=42)
X_resampled, y_resampled = smote.fit_resample(X[numeric_features], y)

print("Before Over-sampling, counts of label '1': {}".format(sum(y==1)))
print("After Over-sampling, counts of label '1': {}".format(sum(y_resampled==1)))
```

Important Output Summary

- Numeric features: **26**
- Categorical features: **9**
- SMOTE applied to correct class imbalance
- Minority class increased to match majority
- Avoids model bias toward the ‘No Attrition’ class

3.3 Feature Scaling

Continuous variables are standardized to maintain uniform ranges. This ensures that no individual variable disproportionately dominates the learning algorithms due to scale differences, promoting balanced model behavior and smoother optimization.

The dataset uses **numeric-only** features during SMOTE and XGBoost training, so the model benefits from naturally scaled distributions.

3.4 Train–Test Split

To measure generalization accurately, the dataset is divided into training and evaluation segments. This partitioning isolates model development from performance assessment, ensuring that evaluation outcomes reflect genuine predictive ability rather than learned memorization.

---

4. Exploratory Data Analysis (EDA)

Before modeling, the data is examined to reveal trends and relationships that shape employee retention outcomes. Graphical and statistical summaries provide the foundation for understanding the behavior of each variable.

Examples of Conducted EDA

- Visual inspection of attrition rates across the company
- Distribution studies for age, salary, tenure, and other numerical features
- Comparisons between employees who remained and those who left using boxplots and segmentations
- Correlation heatmaps highlighting interdependent variables
- Breakdowns of attrition across categorical groups such as JobRole, OverTime, Department

Highlighted Patterns

- Overtime frequency correlates with higher turnover

- Early-career individuals tend to exhibit elevated attrition risk
- Satisfaction-related indicators and work-life balance emerge as influential predictors

These insights help shape the modeling strategy and improve interpretability later in the workflow.

4.1 Attrition Distribution

Code:

```
fig = plt.figure(figsize=(10,5))

df['Attrition'].value_counts().plot(kind='bar')

plt.title('Attrition Distribution')

plt.xlabel('Attrition')

plt.ylabel('Count')

plt.show()
```

Important Output Summary

- **Shows heavy class imbalance:**
  - **"No"** cases dominate
  - **"Yes"** cases are much fewer

4.2 Attrition by Gender

Code (Cell 7):

```
fig = plt.figure(figsize=(10,5))

sns.countplot(data=df, x='Gender', hue='Attrition')

plt.title('Attrition Variation by Gender')
plt.show()
```

Important Output Summary

- Comparable male/female attrition
- Gender does **not** strongly influence attrition

4.3 Attrition by Department

Code (Cell 8):

```
fig = plt.figure(figsize=(12,6))
sns.countplot(data=df, x='Department', hue='Attrition')
plt.title('Attrition Variation by Department')
plt.xticks(rotation=45)
plt.show()
```

Important Output Summary

- Sales department shows noticeably higher attrition rate
- R&D lowest turnover
- HR fluctuates but has fewer total employees

4.4 Attrition by Age Distribution

Code (Cell 9):

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df, x='Attrition', y='Age')
plt.title('Age Distribution by Attrition')
plt.show()
```

Important Output Summary

- Younger employees show higher turnover
- Older employees are more stable
- Suggests early-career mobility is a key factor

4.5 Education Field Influence

Code (Cell 10):

```
fig = plt.figure(figsize=(12,6))

sns.countplot(data=df, x='EducationField', hue='Attrition')

plt.title('Attrition by Education Field')

plt.xticks(rotation=45)

plt.show()
```

Important Output Summary

- "Life Sciences" and "Medical" dominate dataset
- Attrition is widely distributed, not strongly tied to education

4.6 Attrition & Job Satisfaction

Code (Cell 11):

```
plt.figure(figsize=(10,5))

sns.countplot(x='JobSatisfaction', hue='Attrition', data=df)

plt.title('Attrition by Job Satisfaction')

plt.show()
```

Important Output Summary

- Clear pattern: low satisfaction → higher attrition
- Satisfaction = 1 shows highest turnover

4.7 Overtime Status

Code (Cell 12):

```
plt.figure(figsize=(10,5))

sns.countplot(x='OverTime', hue='Attrition', data=df)

plt.title('Attrition by OverTime')

plt.show()
```

Important Output Summary

- Employees working overtime leave significantly more
- One of the strongest predictors of attrition

4.8 Marital Status

Code (Cell 13):

```
plt.figure(figsize=(10,5))

sns.countplot(x='MaritalStatus', hue='Attrition', data=df)

plt.title('Attrition by Marital Status')

plt.show()
```

Important Output Summary

- Single employees have the highest attrition
- Married/divorced are comparatively stable

4.9 Heatmap: Feature Correlation

Code (Cell 14):

```
plt.figure(figsize=(20,15))

sns.heatmap(df.corr(), annot=False, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()
```

Important Output Summary

- Weak multi-collinearity overall
- Strongest correlation:
  - MonthlyIncome ↔ JobLevel
- Attrition correlates negatively with:
  - Job satisfaction
  - YearsAtCompany
  - Age

---

5. Feature Engineering

Feature engineering strengthens the dataset by refining its shape, enhancing clarity, and eliminating attributes that contribute little or introduce noise.
This stage includes transforming categorical inputs, normalizing continuous variables, removing fields that do not support predictive value, and ensuring that no target-related information leaks into the training features.
These refinements produce a cleaner, more expressive

representation of the data, allowing algorithms to detect meaningful signals without being distracted by inconsistencies or irrelevant information.

5.1 Label Encoding

Code (Cell 15):

```
from sklearn.preprocessing import LabelEncoder


le = LabelEncoder()
df['Attrition'] = le.fit_transform(df['Attrition'])
df['OverTime'] = le.fit_transform(df['OverTime'])
df['Gender'] = le.fit_transform(df['Gender'])
df['BusinessTravel'] = le.fit_transform(df['BusinessTravel'])
df['MaritalStatus'] = le.fit_transform(df['MaritalStatus'])
df['EducationField'] = le.fit_transform(df['EducationField'])
df['JobRole'] = le.fit_transform(df['JobRole'])
df['Department'] = le.fit_transform(df['Department'])
```

Important Output Summary

- Converts all categorical values to numeric
- Makes dataset ML-compatible
- No one-hot encoding used, since many algorithms handle label-encoded values well

5.2 Checking Updated Info

Code (Cell 16):

```
df.info()
```

Important Output Summary

- All categorical fields now converted to int64
- Dataset fully numeric
- Ready for modeling

---

6. Machine Learning Model

A suite of classification models is implemented to explore different predictive perspectives. This approach allows the system to evaluate linear, tree-based, and ensemble-based learning behaviors and determine which one most effectively captures the dynamics of attrition.

- Gradient Boosting / XGBoost

Model Development Routine

1. Fit model using the prepared training dataset
2. Generate predictions on unseen test data
3. Assess predictive performance using a range of metrics
4. Compare performance across all models

Through these evaluations, the workflow selects a high-performing candidate that balances predictive strength with operational feasibility.

---

7. Model Evaluation

Model evaluation uses a structured performance analysis framework that quantifies how well each model distinguishes between employees who stay and those who leave.

Metrics Applied

- Accuracy
- Precision, Recall, F1-score
- Confusion Matrix
- ROC-AUC analysis

Beyond numerical evaluation, importance visualizations from tree-based algorithms highlight which employee characteristics most significantly influence turnover risk. Combined findings offer clarity into both model behavior and variable significance, guiding decision-making and improving transparency.

---

8. Model Deployment Simulation

The notebook extends into an operational layer that illustrates how machine-learning assets are managed in a professional environment. This is one of the project's most distinguishing features.

8.1 Model Registry

A formal registry is created to track:

- Stored model binaries (via joblib)

- Automatically generated version identifiers
- Performance metrics
- Time-stamped metadata
- Model lineage for long-term accountability

This registry ensures that every model trained is cataloged with complete provenance.

8.2 Promotion Across Environments

Models transition through three managed environments:

- **Development**
- **Staging**
- **Production**

Promotion rules enforce structured governance:

- Only stable and high-performing models move forward
- The Production model must always be the most reliable among available candidates
- Automated comparison determines whether a Staging model qualifies for promotion
- Logging messages document every successful stage transition

This simulation closely mirrors real-world MLOps pipelines that emphasize quality control, traceability, and continuous model improvement.

---

9. Directory and File Structure

The system generates organized storage to maintain clarity across modeling and versioning workflows.

Key Components

- **/models** — repository for saved model binaries
- **registry.json** — complete history of trained models
- **production_model.json** — details about the currently approved Production model
- Joblib-based model files labeled with version identifiers, such as:
    - *RandomForest_v1.joblib*
    - *LogReg_v3.joblib*

This structure enables easy access, reproducibility, and efficient model governance.

---

10. How to Run the Project

Prerequisite Libraries

Install the following packages:

pandas

numpy

matplotlib

seaborn

scikit-learn

joblib

json

Execution Steps

1. Place the dataset CSV into the working directory
2. Execute the notebook sequentially from top to bottom
3. Review the exploratory visual outputs
4. Train each classification model
5. Examine performance metrics to identify top performers
6. Allow the system to generate versioned model files
7. Promote candidate models through the simulated deployment pipeline

This linear execution guarantees reproducible results and preserves workflow logic.

---

11. Limitations

Despite its depth, the project operates within several boundaries. The dataset may not fully represent all drivers of attrition, and structural imbalances can affect model sensitivity to rare outcomes. Additionally, some variables may oversimplify complex human behaviors that influence resignation decisions.
The deployment simulation, while robust, is implemented locally rather than integrated with distributed systems, cloud orchestration,

or real-time monitoring services commonly found in full enterprise environments.

---

12. Future Improvements

Several enhancements could extend the project’s impact and realism:

- Advanced hyperparameter optimization techniques
- Automated feature selection to reduce noise and enhance model efficiency
- Stronger validation approaches such as nested cross-validation
- Integration with web interfaces or API-driven applications
- Full CI/CD-based MLOps pipeline for production-grade workflows
- Addition of more sophisticated learning models like LightGBM, CatBoost, or deep neural networks
- Considerations for fairness, ethical balance, and bias mitigation

These refinements would elevate the project from a well-constructed proof of concept to an operationally mature predictive system.