HR Attrition Analysis: Insights and Predictive Modelling

A **Data Analyst** project by

AMIT BHADE

*Tools Used – **Google Colab, Excel***

*Techniques – **Python, SQL, Machine Learning***

OBJECTIVES

Employee attrition is one of the most pressing HR challenges. High turnover can increase recruitment and training costs, disrupt teams, and reduce organizational productivity.

This project analyzes IBM's HR dataset to:

1. **Understand current turnover rates** and demographic breakdowns.
2. **Identify key factors influencing attrition**, focusing on job satisfaction, work-life balance, salary/benefits, and other HR factors.
3. **Build predictive models** to forecast attrition and provide data-driven recommendations.

DATASET

Click here to download the dataset

ABOUT DATASET

The dataset contains employee records covering demographics, job-related information, compensation, satisfaction, and performance metrics.
It can be used to analyze workforce characteristics, understand attrition patterns, and develop predictive models for employee turnover.

Key Columns:

- **Age:** Employee's age in years.
- **Attrition:** Whether the employee left the company (Yes/No).
- **BusinessTravel:** Frequency of business travel (Rarely, Frequently, None).
- **Department:** Department of the employee (e.g., Sales, R&D, HR).
- **DistanceFromHome:** Distance from employee's home to workplace (in km).
- **Education:** Education level (1 = Below College, 2 = College, 3 = Bachelor, 4 = Master, 5 = Doctor).
- **EducationField:** Field of study (e.g., Life Sciences, Medical, Marketing).
- **EnvironmentSatisfaction:** Satisfaction with work environment (1–4).
- **Gender:** Male/Female.
- **JobInvolvement:** Level of job involvement (1–4).
- **JobRole:** Employee's role (e.g., Sales Executive, Laboratory Technician, Manager).
- **JobSatisfaction:** Satisfaction with job (1–4).
- **MaritalStatus:** Marital status (Single, Married, Divorced).
- **MonthlyIncome:** Monthly salary.
- **OverTime:** Whether employee works overtime (Yes/No).
- **PercentSalaryHike:** Percentage increase in salary during last appraisal.

- **PerformanceRating:** Performance rating (1–4).
- **StockOptionLevel:** Stock option level provided (0–3).
- **TotalWorkingYears:** Total years of professional experience.
- **TrainingTimesLastYear:** Number of trainings attended in last year.
- **WorkLifeBalance:** Work-life balance rating (1–4).
- **YearsAtCompany:** Total years spent at the company.
- **YearsInCurrentRole:** Number of years in current role.
- **YearsSinceLastPromotion:** Years since last promotion.
- **YearsWithCurrManager:** Years with current manager.

PROJECT SUMMERY

This project focuses on performing an in-depth analysis of HR employee data to understand patterns of attrition and retention. The main objective is to evaluate workforce demographics, job-related factors, and compensation attributes to identify the key drivers of employee turnover.

Special attention is given to:

- Demographic attributes such as age, gender, education, department, and job role.
- Job satisfaction and work-life balance metrics including JobSatisfaction, JobInvolvement, EnvironmentSatisfaction, and WorkLifeBalance.
- Compensation and benefits factors such as MonthlyIncome, PercentSalaryHike, and StockOptionLevel.
- Work-related variables like OverTime, BusinessTravel, and DistanceFromHome.

By combining exploratory data analysis with predictive modelling techniques (Logistic Regression and Random Forest), the project not only highlights the major factors influencing attrition but also builds models capable of predicting at-risk employees. The ultimate goal is to provide actionable insights and strategies for improving employee retention, reducing costs, and enhancing workforce stability.

PROJECT STEPS:

1. Data Overview

- Examined dataset structure.
- Checked for missing values (none found).
- Reviewed descriptive statistics of demographic, job-related, and compensation features.
- Summarized target variable Attrition (≈16% Yes, 84% No).

2. Exploratory Data Analysis (EDA)

- Visualized distributions of key demographic variables (Age, Gender, Education, Department, JobRole).
- Analyzed attrition rates across different groups (e.g., Age bands, Gender, Department, JobRole).
- Explored relationships between attrition and key drivers such as OverTime, MonthlyIncome, WorkLifeBalance, JobSatisfaction, and DistanceFromHome.
- Used statistical tests (Chi-square, T-tests) to confirm significance of categorical and numerical factors.

3. Data Preprocessing

- Converted categorical variables into numerical format (One-Hot Encoding).
- Scaled numerical features for model compatibility.
- Addressed class imbalance in the target variable using SMOTE.
- Split dataset into training and testing sets (80/20 with stratification).

4. Predictive Modeling

- Built classification models (Logistic Regression and Random Forest).
- Applied Stratified K-Fold Cross Validation to evaluate performance.
- Used performance metrics: Accuracy, Precision, Recall, F1-Score, ROC-AUC, and PR-AUC.
- Compared models and identified best-performing one (Random Forest).

5. Results & Insights

- Identified top predictors of attrition: OverTime, JobSatisfaction, MonthlyIncome, WorkLifeBalance, BusinessTravel, Age.
- Quantified impact of demographic and compensation factors on turnover.
- Generated visualizations (confusion matrices, ROC/PR curves, feature importance, odds ratios).

6. Recommendations

- Reduce overtime to mitigate burnout.
- Enhance work-life balance policies (flexible hours, remote work options).
- Improve compensation structures, especially salary hikes for lower-income employees.
- Conduct regular satisfaction surveys and address concerns in high-risk groups (e.g., Sales, younger employees).

STEP BY STEP PROJECT IMPLEMENTATION

Step 1: Import Libraries

```
!pip -q install imbalanced-learn

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats

from sklearn.model_selection import train_test_split, StratifiedKFold, cross_validate
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.metrics import (
    classification_report, confusion_matrix, roc_auc_score,
    roc_curve, precision_recall_curve, average_precision_score
)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE

import warnings, textwrap
warnings.filterwarnings("ignore")

# Aesthetics
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (7,4)
RANDOM_STATE = 42
np.random.seed(RANDOM_STATE)

pd.set_option("display.max_columns", 100)
```

Step 2: Load Dataset

```
from google.colab import files
uploaded = files.upload()

df = pd.read_csv('HR Data.csv')
df.head()
```

Step 3: Overview

```
print("Shape:", df.shape)
print("\nColumns:", df.columns.tolist())
print("\nMissing values per column:\n", df.isnull().sum()[df.isnull().sum()>0])

# Constant columns (should be removed for modeling if constant)
const_cols = [c for c in df.columns if df[c].nunique()==1]
print("\nConstant columns:", const_cols)

# Basic stats
display(df.describe(include='all').T.head(20))
```

Step 4: Prepare helper fields (age bands + numeric target)

```
# Copy to keep raw Attrition labels for plotting
df_eda = df.copy()

# Numeric target for stats/ML
df_eda["Attrition_Flag"] = df_eda["Attrition"].map({"Yes":1, "No":0})

# Age bands for demographic breakdown
bins = [17, 29, 39, 49, 60]
labels = ["<30", "30-39", "40-49", "50+"]
df_eda["AgeBand"] = pd.cut(df_eda["Age"], bins=bins, labels=labels,
include_lowest=True)
df_eda["AgeBand"] = df_eda["AgeBand"].astype("object")
df_eda[["Age", "AgeBand", "Attrition", "Attrition_Flag"]].head()
```

Step 5: Turnover rate & class balance

```
rate = df_eda["Attrition"].value_counts(normalize=True).mul(100).round(2)
print("Overall Attrition Rate (%):")
display(rate.to_frame("Percent"))

sns.countplot(x="Attrition", data=df_eda)
plt.title("Overall Attrition Distribution")
plt.show()
```

Step 6: Helper: Percentage of Attrition by category

```
def percent_attrition_by(col):
    tmp = (df_eda
          .groupby(col)["Attrition_Flag"]
          .mean()
```

```
         .mul(100)
         .reset_index()
         .rename(columns={"Attrition_Flag":"Attrition_%"}))
    tmp = tmp.sort_values("Attrition_%", ascending=False)
    return tmp

def plot_percent_attrition(col, rotate=0):
    tmp = percent_attrition_by(col)
    ax = sns.barplot(data=tmp, x=col, y="Attrition_%")
    plt.title(f"% Attrition by {col}")
    plt.ylabel("% Attrition")
    plt.xlabel(col)
    if rotate:
        plt.xticks(rotation=rotate, ha="right")
    # Annotate bars
    for p in ax.patches:
        ax.annotate(f"{p.get_height():.1f}%", (p.get_x()+p.get_width()/2,
p.get_height()),
                    ha='center', va='bottom', xytext=(0,3), textcoords='offset points')
    plt.tight_layout()
    plt.show()
```

Step 7: Demographic breakdowns

```
# Age bands
display(percent_attrition_by("AgeBand"))
plot_percent_attrition("AgeBand")

# Gender
display(percent_attrition_by("Gender"))
plot_percent_attrition("Gender")

# Education Level (ordinal coded 1-5 in this dataset but treat as category for reporting)
df_eda["Education_cat"] = df_eda["Education"].astype(str)
display(percent_attrition_by("Education_cat"))
plot_percent_attrition("Education_cat")

# Department
display(percent_attrition_by("Department"))
plot_percent_attrition("Department")

# JobRole
display(percent_attrition_by("JobRole"))
plot_percent_attrition("JobRole", rotate=45)
```

Step 8: Key factor analysis (visual EDA)

```
# Categorical drivers → % attrition bars
for col in ["OverTime", "BusinessTravel", "WorkLifeBalance", "JobInvolvement",
          "JobSatisfaction", "EnvironmentSatisfaction", "StockOptionLevel",
"MaritalStatus"]:
    tmpcol = col
    if df_eda[col].dtype != 'O':
        tmpcol = f"{col}_cat"
        df_eda[tmpcol] = df_eda[col].astype(str)
    display(percent_attrition_by(tmpcol))
    plot_percent_attrition(tmpcol, rotate=0 if df_eda[tmpcol].nunique()<6 else 30)

# Numeric drivers → distributions by attrition
num_cols_to_plot = ["MonthlyIncome", "PercentSalaryHike", "DistanceFromHome",
"Age", "TotalWorkingYears"]
for col in num_cols_to_plot:
    ax = sns.boxplot(data=df_eda, x="Attrition", y=col)
    plt.title(f"{col} by Attrition")
    plt.tight_layout()
    plt.show()
```

Step 9: Statistical tests

```
from itertools import chain

cat_vars = [
    "Gender","Department","JobRole","EducationField","BusinessTravel",
    "OverTime","MaritalStatus","WorkLifeBalance","JobInvolvement",
    "JobSatisfaction","EnvironmentSatisfaction","StockOptionLevel","AgeBand"
]

num_vars = ["MonthlyIncome","PercentSalaryHike","DistanceFromHome","Age",
          "TotalWorkingYears","YearsAtCompany","YearsInCurrentRole"]

# Chi-square for categorical vs Attrition
chi_rows = []
for c in cat_vars:
    ct = pd.crosstab(df_eda[c], df_eda["Attrition"])
    chi2, p, dof, exp = stats.chi2_contingency(ct)
    chi_rows.append({"variable": c, "chi2": chi2, "p_value": p, "dof": dof})

chi_df = pd.DataFrame(chi_rows).sort_values("p_value")
print("Chi-square tests (categorical): lower p = stronger association")
display(chi_df)

# Welch t-test for numeric vs Attrition + Cohen's d
def cohens_d(x, y):
    nx, ny = len(x), len(y)
    vx, vy = x.var(ddof=1), y.var(ddof=1)
    pooled = ((nx-1)*vx + (ny-1)*vy) / (nx+ny-2)
    d = (x.mean() - y.mean()) / np.sqrt(pooled)
    return d

tt_rows = []
a1 = df_eda["Attrition"]=="Yes"
a0 = df_eda["Attrition"]=="No"

for c in num_vars:
    x, y = df_eda.loc[a1, c].values, df_eda.loc[a0, c].values
    t, p = stats.ttest_ind(x, y, equal_var=False)
    d = cohens_d(df_eda.loc[a1, c], df_eda.loc[a0, c])
    tt_rows.append({"variable": c, "t_stat": t, "p_value": p, "cohens_d": d})

tt_df = pd.DataFrame(tt_rows).sort_values("p_value")
print("Welch t-tests (numeric): lower p = stronger difference")
display(tt_df)
```

Step 10: Prepare data for modeling

```
# Create working copy
data = df.copy()

# Target to numeric
data["Attrition"] = data["Attrition"].map({"Yes":1, "No":0})

# Drop ID/constant columns if present
drop_cols = [c for c in
["EmployeeNumber","EmployeeCount","Over18","StandardHours"] if c in data.columns]
data = data.drop(columns=drop_cols)

# Identify column types
cat_cols = data.select_dtypes(include=["object"]).columns.tolist()
num_cols =
data.select_dtypes(include=["int64","float64","int32","float32","int16"]).drop("Attrition",
axis=1).columns.tolist()

print("Categorical cols:", cat_cols)
print("Numeric cols:", num_cols)
print("Target balance:\n",
data["Attrition"].value_counts(normalize=True).mul(100).round(2))

# Split
X = data.drop("Attrition", axis=1)
y = data["Attrition"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=RANDOM_STATE
)

# Preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), num_cols),
        ("cat", OneHotEncoder(drop=None, handle_unknown="ignore"), cat_cols),
    ],
    remainder="drop"
)
```

Step 11: Logistic Regression + SMOTE (CV & Test)

```
logit_pipe = Pipeline(steps=[
    ("prep", preprocessor),
    ("smote", SMOTE(random_state=RANDOM_STATE)),
    ("clf", LogisticRegression(max_iter=2000))
])

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=RANDOM_STATE)
scoring = {
    "accuracy":"accuracy",
    "precision":"precision",
    "recall":"recall",
    "f1":"f1",
    "roc_auc":"roc_auc",
    "pr_auc":"average_precision"
}

cv_res = cross_validate(logit_pipe, X_train, y_train, cv=cv, scoring=scoring, n_jobs=-1,
return_train_score=False)
cv_summary = {k: (np.mean(v), np.std(v)) for k,v in cv_res.items() if k.startswith("test_")}
print("5-fold CV (Logistic + SMOTE): mean ± std")
for k,(m,s) in cv_summary.items():
    print(f"{k.replace('test_',''):>9}: {m:.3f} ± {s:.3f}")

# Fit on full train, evaluate on test
logit_pipe.fit(X_train, y_train)
proba = logit_pipe.predict_proba(X_test)[:,1]
pred  = (proba >= 0.5).astype(int)

print("\nTest ROC-AUC:", roc_auc_score(y_test, proba).round(3))
print("\nClassification Report (Logistic):\n", classification_report(y_test, pred, digits=3))

# Confusion matrix (normalized)
cm = confusion_matrix(y_test, pred, normalize='true')
```

```
sns.heatmap(cm, annot=True, fmt=".2f", xticklabels=["No","Yes"],
yticklabels=["No","Yes"])
plt.title("Logistic Regression — Confusion Matrix (Normalized)")
plt.xlabel("Predicted"); plt.ylabel("True")
plt.show()
```

Step 12: Random Forest + SMOTE (CV & Test)

```
rf_pipe = Pipeline(steps=[
    ("prep", preprocessor),
    ("smote", SMOTE(random_state=RANDOM_STATE)),
    ("clf", RandomForestClassifier(
        n_estimators=400, max_depth=None, min_samples_split=4,
        n_jobs=-1, random_state=RANDOM_STATE
    ))
])

cv_res_rf = cross_validate(rf_pipe, X_train, y_train, cv=cv, scoring=scoring, n_jobs=-1,
return_train_score=False)
cv_summary_rf = {k: (np.mean(v), np.std(v)) for k,v in cv_res_rf.items() if
k.startswith("test_")}
print("5-fold CV (RandomForest + SMOTE): mean ± std")
for k,(m,s) in cv_summary_rf.items():
    print(f"{k.replace('test_',''):>9}: {m:.3f} ± {s:.3f}")

rf_pipe.fit(X_train, y_train)
proba_rf = rf_pipe.predict_proba(X_test)[:,1]
pred_rf  = (proba_rf >= 0.5).astype(int)

print("\nTest ROC-AUC (RF):", roc_auc_score(y_test, proba_rf).round(3))
print("\nClassification Report (Random Forest):\n", classification_report(y_test, pred_rf,
digits=3))

cm_rf = confusion_matrix(y_test, pred_rf, normalize='true')
sns.heatmap(cm_rf, annot=True, fmt=".2f", xticklabels=["No","Yes"],
yticklabels=["No","Yes"])
plt.title("Random Forest — Confusion Matrix (Normalized)")
plt.xlabel("Predicted"); plt.ylabel("True")
plt.show()
```

Step 13: ROC & PR curves (both models)

```
# Logistic
fpr_l, tpr_l, _ = roc_curve(y_test, proba)
prec_l, rec_l, _ = precision_recall_curve(y_test, proba)
aupr_l = average_precision_score(y_test, proba)

# RF
fpr_r, tpr_r, _ = roc_curve(y_test, proba_rf)
prec_r, rec_r, _ = precision_recall_curve(y_test, proba_rf)
aupr_r = average_precision_score(y_test, proba_rf)

# ROC
plt.plot(fpr_l, tpr_l, label=f"Logit (AUC={roc_auc_score(y_test, proba):.3f})")
plt.plot(fpr_r, tpr_r, label=f"RF (AUC={roc_auc_score(y_test, proba_rf):.3f})")
plt.plot([0,1],[0,1],'--')
plt.xlabel("False Positive Rate"); plt.ylabel("True Positive Rate")
plt.title("ROC Curves")
plt.legend(); plt.show()

# PR
plt.plot(rec_l, prec_l, label=f"Logit (AUPR={aupr_l:.3f})")
plt.plot(rec_r, prec_r, label=f"RF (AUPR={aupr_r:.3f})")
plt.xlabel("Recall"); plt.ylabel("Precision")
plt.title("Precision-Recall Curves")
plt.legend(); plt.show()
```

Step 14: Feature importance (RF)

```
# Get feature names after preprocessing
ohe = rf_pipe.named_steps["prep"].named_transformers_["cat"]
num_names = rf_pipe.named_steps["prep"].transformers_[0][2]
cat_names =
ohe.get_feature_names_out(input_features=rf_pipe.named_steps["prep"].transformers_[1][
2])
feat_names = np.r_[num_names, cat_names]

rf = rf_pipe.named_steps["clf"]
importances = pd.Series(rf.feature_importances_,
index=feat_names).sort_values(ascending=False)
topN = 20
display(importances.head(topN).to_frame("importance"))

ax = importances.head(topN).iloc[::-1].plot(kind="barh")
plt.title("Top Feature Importances — Random Forest")
plt.xlabel("Gini Importance")
plt.tight_layout()
plt.show()
```

Step 15: Logistic coefficients → odds ratios

```
# Retrieve trained logistic model on test-fit pipeline
log_model = logit_pipe.named_steps["clf"]
ohe = logit_pipe.named_steps["prep"].named_transformers_["cat"]
num_names = logit_pipe.named_steps["prep"].transformers_[0][2]
cat_names =
ohe.get_feature_names_out(input_features=logit_pipe.named_steps["prep"].transformers_[
1][2])
feat_names = np.r_[num_names, cat_names]

coefs = pd.Series(log_model.coef_.ravel(), index=feat_names)
odds = np.exp(coefs)  # >1 increases attrition odds; <1 decreases
coef_df = pd.DataFrame({"coef": coefs, "odds_ratio": odds, "abs_coef":
coefs.abs()}).sort_values("abs_coef", ascending=False)

print("Top 15 (by absolute coefficient):")
display(coef_df.head(15))

# Visualize sign & magnitude
top = coef_df.head(20).copy()
colors = top["coef"].apply(lambda x: "red" if x>0 else "blue")
top[::-1]["coef"].plot(kind="barh", color=colors[::-1])
plt.title("Logistic Coefficients (Top |Attrition impact|)")
plt.xlabel("Coefficient (log-odds)")
plt.tight_layout()
plt.show()
```

Step 16: Auto summary bullets

```
summary_points = []

# Overall rate
overall = df_eda["Attrition_Flag"].mean()*100
summary_points.append(f"Overall attrition rate ≈ {overall:.1f}%.")

# Top 5 RF features
top5_rf = importances.head(5).index.tolist()
summary_points.append("Top RF predictors: " + ", ".join(top5_rf) + ".")

# Strongest positive/negative drivers from logistic
pos = coef_df[coef_df.coef>0].head(3).index.tolist()
neg = coef_df[coef_df.coef<0].head(3).index.tolist()
summary_points.append("Logistic (↑ risk): " + ", ".join(pos) + ".")
summary_points.append("Logistic (↓ risk): " + ", ".join(neg) + ".")
```

```
# Key categorical signals (use chi-square ordering)
summary_points.append("Strong categorical associations (chi-square): " + ", ".join(chi_df.head(5)["variable"]) + ".")

print("\n".join(f"- {s}" for s in summary_points))
```

KEY INSIGHTS

- **Overall Attrition Rate:** About 16% of employees left the company.
- **Demographics:** Younger employees (<30) and singles are more likely to leave; Sales department shows highest attrition.
- **Overtime:** Employees working overtime are 2–3 times more likely to quit.
- **Work-Life Balance:** Poor balance strongly correlates with higher attrition.
- **Job Satisfaction:** Low job and environment satisfaction are key drivers of turnover.
- **Compensation:** Lower monthly income and smaller salary hikes increase attrition risk.
- **Benefits:** Employees with fewer stock options show slightly higher attrition.
- **Distance & Travel:** Longer commute distances and frequent business travel increase the likelihood of leaving.
- **Models:** Random Forest achieved best predictive performance (ROC-AUC ≈ 0.82), with top predictors being Overtime, Job Satisfaction, Monthly Income, Age, and Business Travel.

CONCLUSION

What we did

- Explored the IBM HR dataset with demographics, job, and compensation details.
- Performed descriptive analysis and visualized turnover across age, gender, department, job role, and other factors.
- Identified key drivers of attrition such as Overtime, Work-Life Balance, Job Satisfaction, and Monthly Income.
- Built and compared predictive models (Logistic Regression and Random Forest) using Stratified Cross-Validation and SMOTE for imbalance handling.
- Evaluated models on accuracy, precision, recall, F1, ROC-AUC, and PR-AUC; produced confusion matrices, ROC/PR curves, and feature importance plots.
- Generated actionable recommendations for HR to reduce turnover and improve retention.

Key Insights

- **Overtime** is the strongest predictor: employees working overtime are much more likely to quit.
- **Work-Life Balance** and **Job Satisfaction** significantly affect retention.
- **Compensation factors** (low income, small salary hikes) are linked to higher attrition.
- **Demographics**: Younger and single employees, especially in Sales roles, show higher attrition.
- **Distance & Travel**: Longer commutes and frequent travel increase attrition risk.

Best Model and Performance

- Random Forest was the best-performing model, achieving:
  - ❖ ROC-AUC ≈ 0.82 on the test set.
  - ❖ Higher recall for attrition cases compared to Logistic Regression.
- Logistic Regression provided interpretable coefficients (odds ratios), while Random Forest highlighted non-linear feature interactions.

Limitations

- Dataset represents a synthetic HR sample; results may not generalize to every organization.
- Many satisfaction-related features are self-reported, which can introduce bias.
- The dataset is static; no temporal/longitudinal data on attrition trends.

Future Work

- Apply hyperparameter tuning and explore additional models (e.g., Gradient Boosting, XGBoost).
- Incorporate employee engagement survey data or performance reviews for richer features.
- Deploy model as an HR dashboard for real-time attrition risk monitoring.
- Use cost-sensitive learning to balance the business cost of false positives vs. false negatives.

Business Impact

- The model can help HR teams identify at-risk employees early and design targeted interventions.
- Policies to reduce overtime, improve work-life balance, and adjust compensation can directly reduce turnover.
- A proactive retention strategy based on this analysis can lower recruitment costs, preserve institutional knowledge, and improve workforce stability.

REFERENCE