Elijah Cruz

April 20, 2025

DAT-430

7-2 Project Two Submission

To begin the preprocessing on the HR Attrition dataset, I first imported the libraries I would need. I used pandas and NumPy for data manipulation and numerical calculations. I also imported sklearn for preprocessing, including tools like OrdinalEncoder, train_test_split, classification_report, confusion_matrix, and more. Additionally, I imported matplotlib and seaborn for data visualizations. The data.info() function in pandas provides a concise summary of a DataFrame. This function is especially helpful in the initial stages of data exploration, as it gives a quick overview of the dataset's structure and highlights any missing values that may need to be handled during preprocessing.

```python
# Libray for Data Manipulation.
import numpy as np
import pandas as pd

#Library for Data Visualization.
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as ticker
sns.set(style="white",font_scale=1.5)
sns.set(rc={"axes.facecolor":"#FFFAF0","figure.facecolor":"#FFFAF0"})
sns.set_context("poster",font_scale = .7)

# Library to perform Statistical Analysis.
from scipy import stats
from scipy.stats import chi2
from scipy.stats import chi2_contingency

# Library to Display whole Dataset.
pd.set_option("display.max.columns",None)
pd.set_option("display.max.rows",None)

#Our tools

from sklearn.impute import SimpleImputer #handles missing data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, OrdinalEncoder #encoding categorical data
from sklearn.tree import DecisionTreeClassifier, plot_tree #Decision Tree
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV, permutation_test_score #splitting train
from sklearn.preprocessing import StandardScaler, MinMaxScaler # feature scaling
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, LinearRegression # Logistic Regression
from sklearn.metrics import mean_squared_error, accuracy_score, precision_score, recall_score, f1_score, classification_repor
from scipy.stats import logistic
import statsmodels.api as sm
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
```

```
In [6]: data.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
```

To enhance the interpretability of the dataset, I converted several numerical categorical columns into more meaningful string labels. This involved replacing numerical codes with descriptive category names using the .replace() method. For example, values in the "EnvironmentSatisfaction", "JobInvolvement", "JobSatisfaction", and "RelationshipSatisfaction" columns were mapped from 1–4 to corresponding labels such as "Low", "Medium", "High", and "Very High". Similarly, "PerformanceRating" was updated to reflect ratings like "Low", "Good", "Excellent", and "Outstanding", while "WorkLifeBalance" was translated into "Bad", "Good", "Better", and "Best". Educational levels were updated from numeric values to categories like "Below College", "Bachelor", and "Doctor", and the "JobLevel" column was transformed to represent job seniority from "Entry Level" to "Executive Level". These changes help make the data more readable and intuitive for analysis and visualization.

```
In [19]: data.select_dtypes(include='O').sample(5)
```

Out[19]:

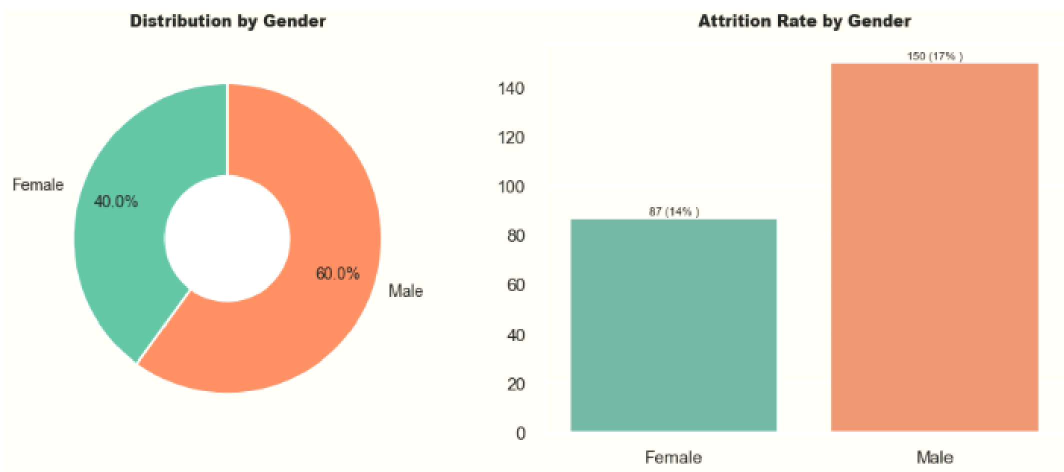
	Attrition	BusinessTravel	Department	Education	EducationField	EnvironmentSatisfaction	Gender	JobInvolvement	JobLevel	JobRole	JobSa
798	Yes	Travel_Frequently	Research & Development	Bachelor	Technical Degree	High	Male	Medium	Entry Level	Laboratory Technician	
787	No	Travel_Frequently	Research & Development	Below College	Life Sciences	Very High	Male	High	Mid Level	Manufacturing Director	
1408	No	Travel_Rarely	Research & Development	College	Other	Very High	Male	High	Entry Level	Laboratory Technician	
427	No	Travel_Frequently	Sales	Bachelor	Marketing	High	Female	Medium	Mid Level	Sales Executive	
776	Yes	Travel_Rarely	Research & Development	Below College	Medical	Very High	Male	High	Entry Level	Laboratory Technician	

To visualize both the overall distribution and the attrition breakdown of categorical features, I created a custom function called pie_bar_plot. This function takes in a DataFrame, the column to analyze, and a target column (in this case, "Attrition") to use as a filter. It produces two side-by-side plots: a pie chart and a bar plot. The pie chart displays the percentage distribution of all values within the specified column (e.g., "Gender"), providing a clear overview of category proportions. The bar plot, on the other hand, focuses on employees who left the company (i.e., where "Attrition" is "Yes"), showing the count of attrition instances for each category and annotating them with the absolute values and their percentage relative to the total in that category. This dual-visual approach offers both a general perspective and a targeted look at how different groups contribute to attrition, aiding in more nuanced analysis.

Distribution by Gender: A donut chart showing that your workforce is composed of 60% male and 40% female employees.

Attrition Rate by Gender: A bar chart displaying that there are 100 male employees who have left (representing 17% attrition) compared to 87 female employees (14% attrition)

```python
def pie_bar_plot(df, col, hue):
    plt.figure(figsize=(14, 6))

    # Extract value counts for the specified column
    value_counts = df[col].value_counts().sort_index()

    # First subplot: Pie chart
    plt.subplot(1, 2, 1)
    ax1 = value_counts
    plt.title(f"Distribution by {col}", fontweight="black", size=14, pad=15)
    colors = sns.color_palette('Set2', len(ax1))
    plt.pie(ax1.values, labels=ax1.index, autopct="%.1f%%", pctdistance=0.75, startangle=90,
            colors=colors, textprops={"size":14})
    center_circle = plt.Circle((0, 0), 0.4, fc='white')
    fig = plt.gcf()
    fig.gca().add_artist(center_circle)

 # Second subplot: Bar plot
    plt.subplot(1, 2, 2)
    new_df = df[df[hue] == 'Yes']
    value_1 = value_counts
    value_2 = new_df[col].value_counts().sort_index()  # Sort the values in the same order
    ax2 = np.floor((value_2 / value_1) * 100).values
    sns.barplot(x=value_2.index, y=value_2.values, palette='Set2')
    plt.title(f"Attrition Rate by {col}", fontweight="black", size=14, pad=15)
    for index, value in enumerate(value_2):
        plt.text(index, value, str(value) + " (" + str(int(ax2[index])) + "% )", ha="center", va="bottom", size=10)

    plt.tight_layout()
    plt.show()
pie_bar_plot(data, 'Gender', 'Attrition')
```



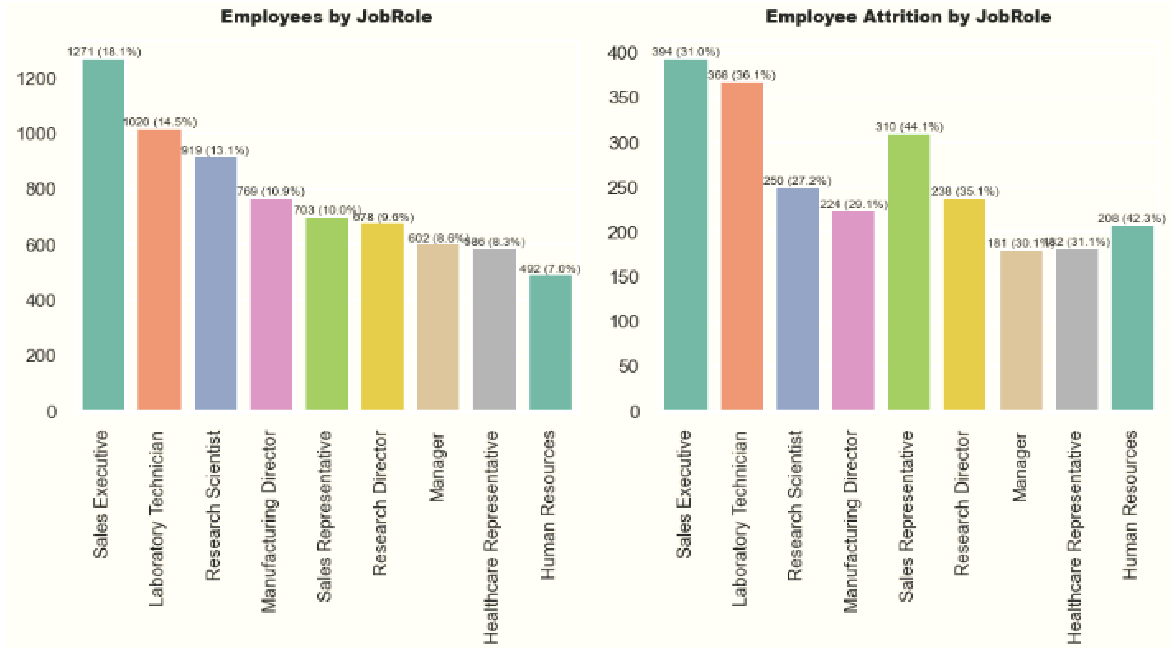
To further analyze how employee attrition varies across different categories, I defined a function called count center plot. This function takes in a DataFrame, a categorical column to analyze (e.g., "JobRole"), and a target column (e.g., "Attrition"). It produces two bar plots side by side. The first plot displays the total number of employees in each category, with each bar labeled to show both the count and the percentage of the total dataset. The second plot focuses on employees who have left the company, showing attrition counts for each category while also indicating the attrition rate as a percentage of the total in that category. By maintaining a consistent order across both plots, this function makes it easy to compare category sizes and attrition trends, offering clear insights into which roles or groups are most affected by attrition.

Employees by JobRole: Shows the distribution of employees across different job roles, with Sales Executive having the highest count (1,271 employees, 18.1%) and Human Resources having the lowest (482 employees, 6.9%).

Employee Attrition by JobRole: Shows the number of employees who have left from each role, with Sales Executive again having the highest attrition (394 employees, 31.0%) and Healthcare Representative having the lowest (206 employees, 42.3%).

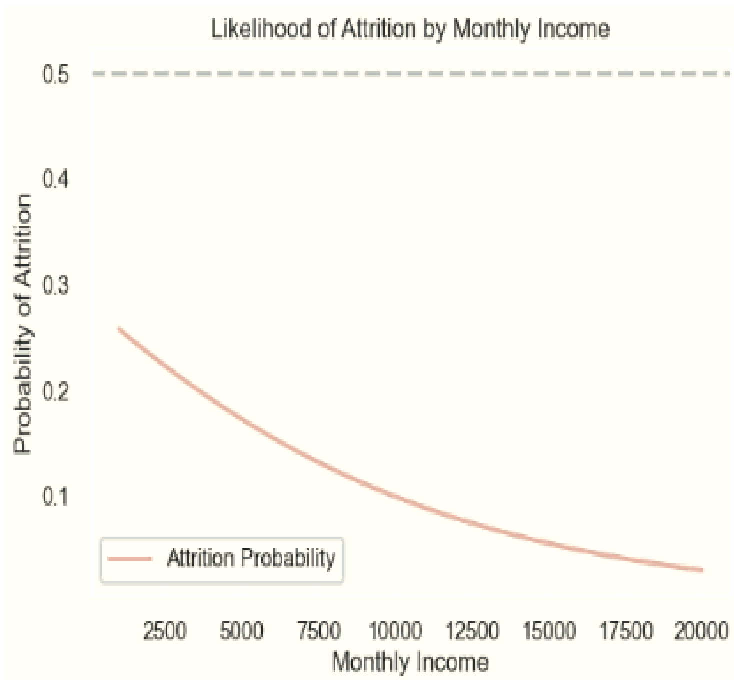
```
[32]: def count_percent_plot(df, col, hue):

    plt.figure(figsize=(13.5, 8))
    plt.subplot(1, 2, 1)
    value_1 = df[col].value_counts()
    sns.barplot(x=value_1.index, y=value_1.values, order=value_1.index, palette='Set2')
    plt.title(f"Employees by {col}", fontweight="black", size=14, pad=15)
    for index, value in enumerate(value_1.values):
        count_percentage = "{:.1f}%".format((value / len(df)) * 100)
        plt.text(index, value, f"{value} ({count_percentage})", ha="center", va="bottom", size=10)
    plt.xticks(rotation=90)

    # Sort the values for the second subplot to match the order of the first subplot
    value_2 = df[df[hue] == 'Yes'][col].value_counts().reindex(value_1.index)
    plt.subplot(1, 2, 2)
    attrition_rate = (value_2 / value_1 * 100).values
    sns.barplot(x=value_2.index, y=value_2.values, order=value_1.index, palette='Set2')
    plt.title(f"Employee Attrition by {col}", fontweight="black", size=14, pad=15)
    for index, value in enumerate(value_2.values):
        attrition_percentage = "{:.1f}%".format(np.round(attrition_rate[index], 1))
        plt.text(index, value, f"{value} ({attrition_percentage})", ha="center", va="bottom", size=10)
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```

```
In [34]: count_percent_plot(data, 'JobRole', 'Attrition')
```



To examine the effect of monthly income on employee attrition, I performed a logistic regression using Monthly Income as the independent variable and a binary version of the Attrition column as the dependent variable. After fitting the model, the resulting equation for the log odds of attrition was Log odds = -0.9291 + (-0.0001) × Monthly Income. This indicates a slight negative relationship between monthly income and the likelihood of attrition, suggesting that as income increases, the probability of an employee leaving the company slightly decreases. To visualize this, I generated a smooth range of income values and plotted the predicted probabilities of attrition. The plot clearly shows a downward trend, emphasizing that employees with lower income are more likely to leave, while those with higher income have a reduced attrition risk.

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

X = data[['MonthlyIncome']]
y = data['Attrition_binary']

log_reg = LogisticRegression()
log_reg.fit(X, y)

intercept = log_reg.intercept_[0]
coefficient = log_reg.coef_[0][0]
print(f"Log odds = {intercept:.4f} + ({coefficient:.4f}) × MonthlyIncome")

#income range
income_range = np.linspace(X.min(), X.max(), 500).reshape(-1, 1)

# prediction
attrition_probs = log_reg.predict_proba(income_range)[:, 1]

plt.figure(figsize = (8, 6))
plt.plot(income_range, attrition_probs, color = '#E2B3A3', label = 'Attrition Probability')
plt.xlabel('Monthly Income')
plt.ylabel('Probability of Attrition')
plt.title('Likelihood of Attrition by Monthly Income')
plt.axhline(0.5, linestyle = '--', color = 'gray', alpha = 0.5)
plt.grid(alpha = 0.3)
plt.legend()
plt.tight_layout()
plt.show()
```



This graph shows the relationship between monthly income and probability of attrition. It clearly illustrates that employees with lower monthly incomes (around $2,500) have a much higher probability of leaving the company (approximately 0.25 or 25%) compared to those with higher incomes (around $20,000) who have a much lower attrition probability (about 0.05 or 5%).

RESOURCES:

GeeksforGeeks. (2025, February 11). *Feature selection techniques in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/

GeeksforGeeks. (2024, December 10). *Bar plot in Matplotlib*. GeeksforGeeks. https://www.geeksforgeeks.org/bar-plot-in-matplotlib/

W3Schools.com. (n.d.). https://www.w3schools.com/python/pandas/default.asp

Aymanlbd. (2025, April 7). *Employee Attrition analysis*. https://www.kaggle.com/code/aymanlbd/employee-attrition-analysis