Pattern_HW2_student_2026

January 30, 2026

1 Employee Attrition Prediction

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1.1 Read CSV File

[2]:
```python
df = pd.read_csv("hr-employee-attrition-with-null.csv")
```

1.2 Dataset Statistic

[3]:
```python
df.describe()
```

[3]:
```
        Unnamed: 0          Age    DailyRate  DistanceFromHome    Education  \
count  1470.000000  1176.000000  1176.000000        1176.00000  1176.000000
mean    734.500000    37.134354   798.875850           9.37500     2.920918
std     424.496761     9.190317   406.957684           8.23049     1.028796
min       0.000000    18.000000   102.000000           1.00000     1.000000
25%     367.250000    30.000000   457.750000           2.00000     2.000000
50%     734.500000    36.000000   798.500000           7.00000     3.000000
75%    1101.750000    43.000000  1168.250000          15.00000     4.000000
max    1469.000000    60.000000  1499.000000          29.00000     5.000000

       EmployeeCount  EmployeeNumber  EnvironmentSatisfaction   HourlyRate  \
count         1176.0     1176.000000              1176.000000  1176.000000
mean             1.0     1031.399660                 2.733844    65.821429
std              0.0      601.188955                 1.092992    20.317323
min              1.0        1.000000                 1.000000    30.000000
25%              1.0      494.750000                 2.000000    48.000000
50%              1.0     1027.500000                 3.000000    66.000000
75%              1.0     1562.250000                 4.000000    84.000000
max              1.0     2068.000000                 4.000000   100.000000

       JobInvolvement  …  RelationshipSatisfaction  StandardHours  \
count     1176.000000  …               1176.000000         1176.0
mean         2.728741  …                  2.694728           80.0
```

```
std          0.705280  …                  1.093660           0.0
min          1.000000  …                  1.000000          80.0
25%          2.000000  …                  2.000000          80.0
50%          3.000000  …                  3.000000          80.0
75%          3.000000  …                  4.000000          80.0
max          4.000000  …                  4.000000          80.0

       StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
count       1176.000000        1176.000000            1176.000000
mean           0.752551          11.295068               2.787415
std            0.822550           7.783376               1.290507
min            0.000000           0.000000               0.000000
25%            0.000000           6.000000               2.000000
50%            1.000000          10.000000               3.000000
75%            1.000000          15.000000               3.000000
max            3.000000          40.000000               6.000000

       WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
count      1176.000000     1176.000000         1176.000000
mean          2.770408        7.067177            4.290816
std           0.705004        6.127836            3.630901
min           1.000000        0.000000            0.000000
25%           2.000000        3.000000            2.000000
50%           3.000000        5.000000            3.000000
75%           3.000000       10.000000            7.000000
max           4.000000       37.000000           18.000000

       YearsSinceLastPromotion  YearsWithCurrManager
count              1176.000000           1176.000000
mean                  2.159014              4.096939
std                   3.163524              3.537393
min                   0.000000              0.000000
25%                   0.000000              2.000000
50%                   1.000000              3.000000
75%                   2.250000              7.000000
max                  15.000000             17.000000

[8 rows x 27 columns]
```

[4]:
```
df.head()
```

[4]:
```
   Unnamed: 0   Age Attrition     BusinessTravel  DailyRate  \
0           0  41.0       Yes      Travel_Rarely        NaN
1           1   NaN        No                NaN      279.0
2           2  37.0       Yes                NaN     1373.0
3           3   NaN        No  Travel_Frequently     1392.0
4           4  27.0        No      Travel_Rarely      591.0
```

```
               Department  DistanceFromHome  Education EducationField  \
0                     NaN               1.0        NaN  Life Sciences
1  Research & Development               NaN        NaN  Life Sciences
2                     NaN               2.0        2.0            NaN
3  Research & Development               3.0        4.0  Life Sciences
4  Research & Development               2.0        1.0        Medical

   EmployeeCount  …  RelationshipSatisfaction  StandardHours  \
0            1.0  …                       1.0           80.0
1            1.0  …                       4.0            NaN
2            1.0  …                       NaN           80.0
3            NaN  …                       3.0            NaN
4            1.0  …                       4.0           80.0

  StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  WorkLifeBalance  \
0              0.0                8.0                    0.0              NaN
1              1.0               10.0                    NaN              3.0
2              0.0                7.0                    3.0              NaN
3              NaN                8.0                    3.0              NaN
4              1.0                6.0                    NaN              3.0

  YearsAtCompany  YearsInCurrentRole YearsSinceLastPromotion  \
0            6.0                 NaN                     0.0
1           10.0                 NaN                     NaN
2            NaN                 0.0                     NaN
3            8.0                 NaN                     3.0
4            2.0                 2.0                     2.0

   YearsWithCurrManager
0                   NaN
1                   7.0
2                   0.0
3                   0.0
4                   NaN

[5 rows x 36 columns]
```

1.3 Feature transformation

[5]:
```
df.loc[df["Attrition"] == "no", "Attrition"] = 0.0
df.loc[df["Attrition"] == "yes", "Attrition"] = 1.0

string_categorical_col = [
    "Department", "Attrition", "BusinessTravel",
```

```
    "EducationField", "Gender", "JobRole",
    "MaritalStatus", "Over18", "OverTime"
]

# ENCODE STRING COLUMNS TO CATEGORICAL COLUMNS
for col in string_categorical_col:
    # INSERT CODE HERE
    df[col] = df[col].astype("category").cat.codes

# HANDLE NULL NUMBERS
# INSERT CODE HERE
df = df.fillna(df.median())
df = df.loc[:, ~df.columns.isin(["EmployeeNumber", "Unnamed: 0",␣
 ↪"EmployeeCount", "StandardHours", "Over18"])]
```

1.3.1 Spliting data into train and test

[6]:
```
from sklearn.model_selection import train_test_split
```

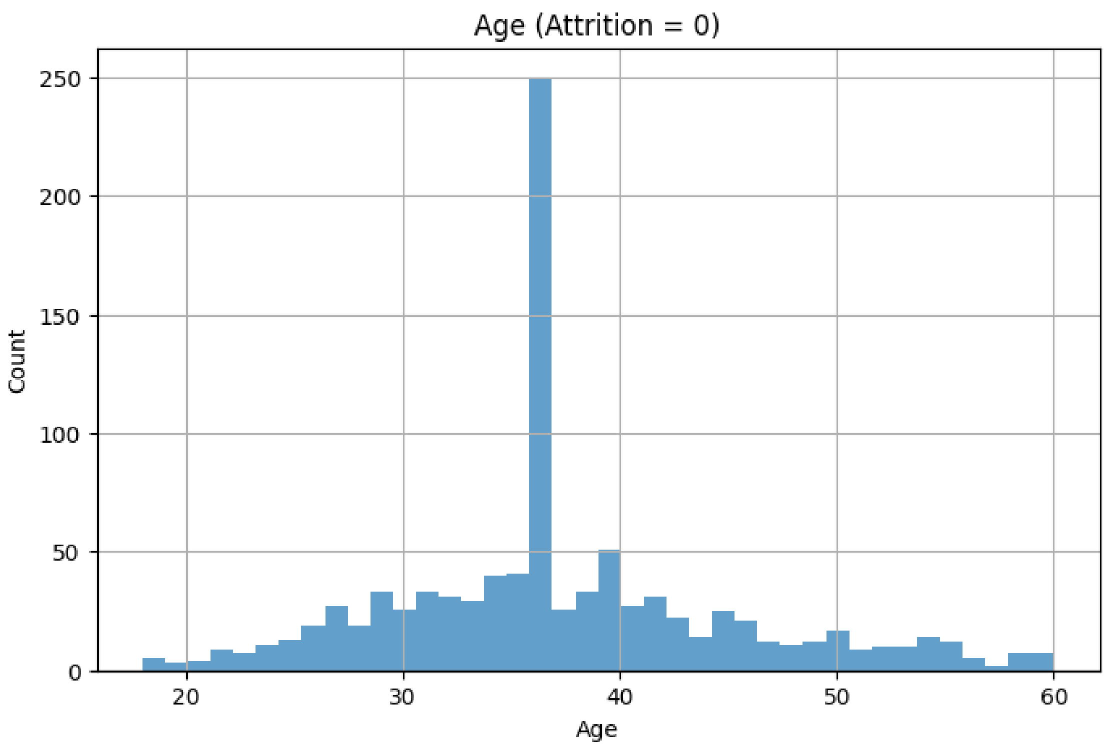
[7]:
```
df_train, df_test = train_test_split(df, test_size=0.2, random_state=42)
```

1.3.2 Display histogram of each feature

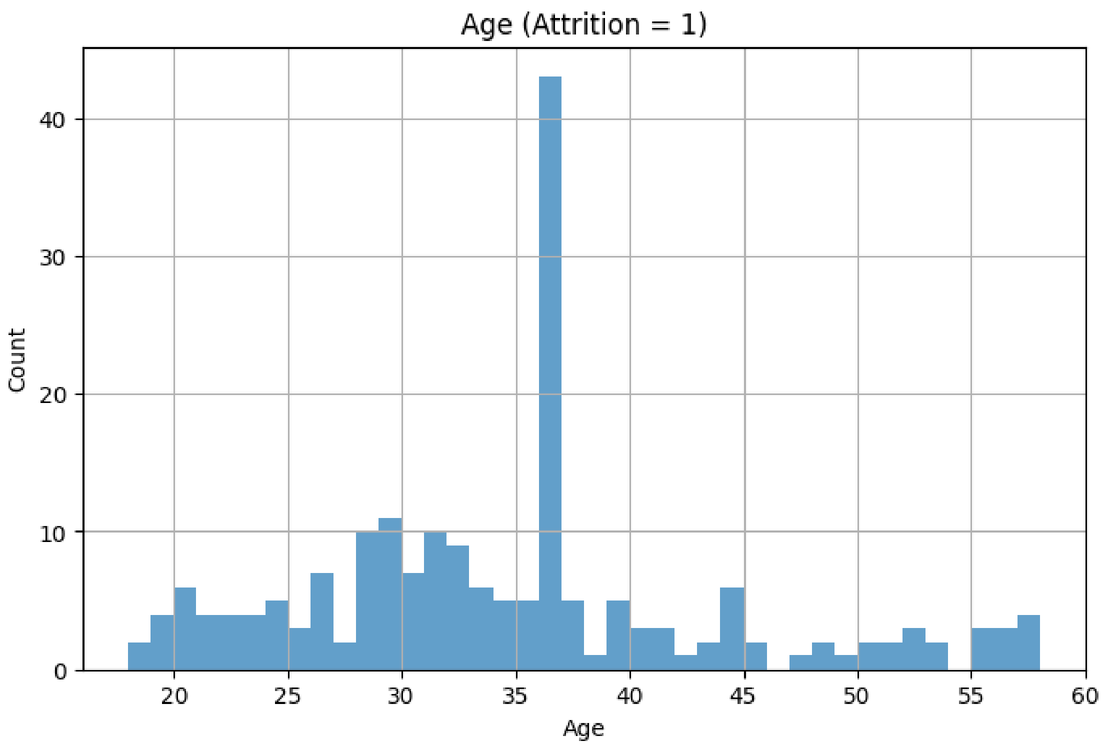
[8]:
```
def display_histogram(df, col_name, cls, n_bin = 40):
    # INSERT CODE HERE
    # Filter data by Attrition == cls
    data = df[df["Attrition"] == cls][col_name]

    plt.figure(figsize=(8, 5))

    counts, bins, _ = plt.hist(
        data,
        bins=n_bin,
        alpha=0.7
    )

    zero_bins = np.sum(counts == 0)

    plt.xlabel(col_name)
    plt.ylabel("Count")
    plt.title(f"{col_name} (Attrition = {cls})")
    plt.grid(True)

    plt.show()

    print(f"Number of bins with zero counts: {zero_bins}")
```

1.3.3 Problem T4

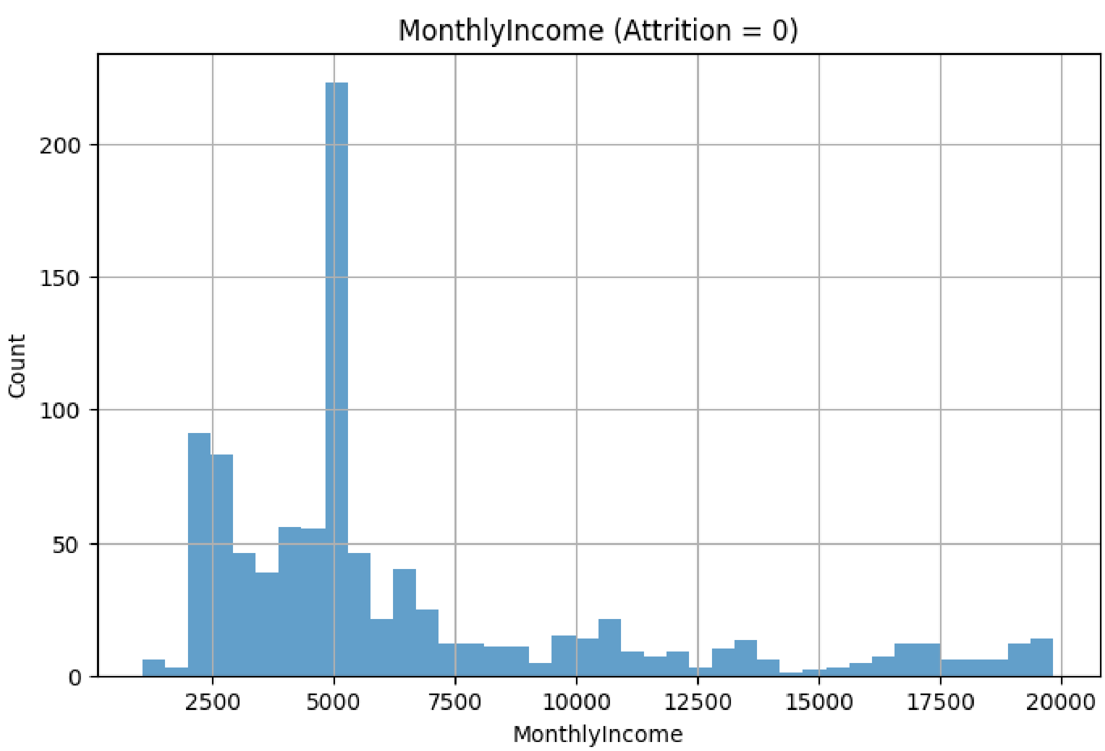
[9]:
```
display_histogram(df_train, "Age", 0)
display_histogram(df_train, "Age", 1)

display_histogram(df_train, "MonthlyIncome", 0)
display_histogram(df_train, "MonthlyIncome", 1)

display_histogram(df_train, "DistanceFromHome", 0)
display_histogram(df_train, "DistanceFromHome", 1)
```



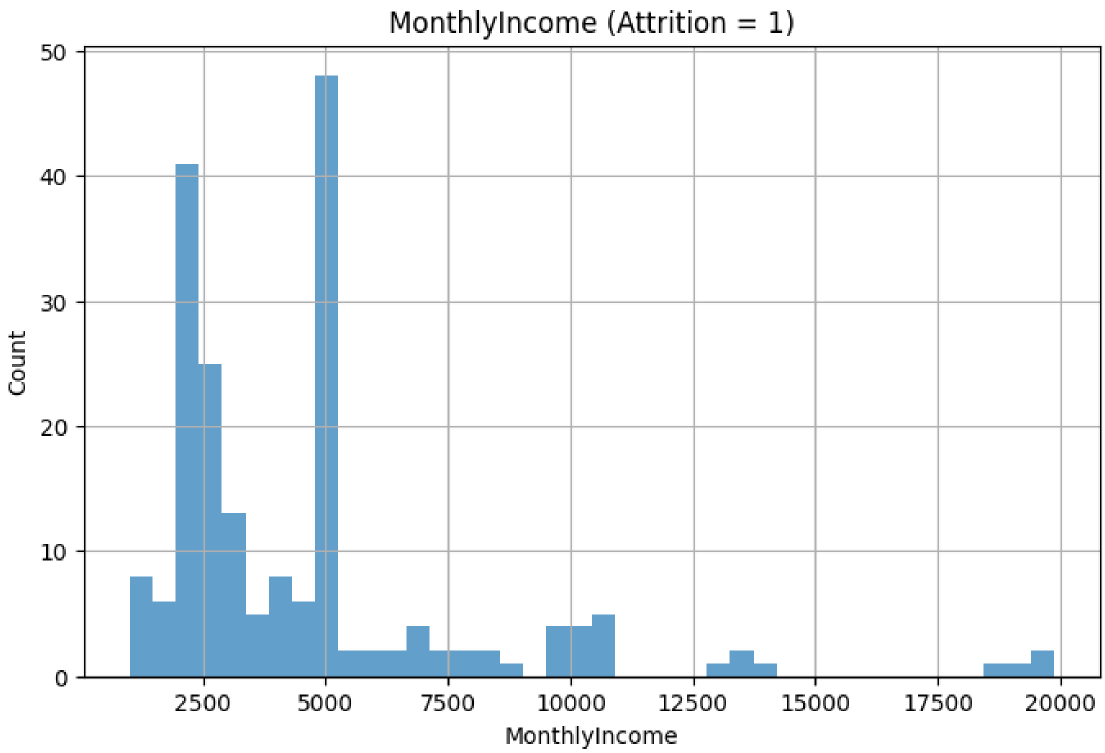

```
Number of bins with zero counts: 0
```



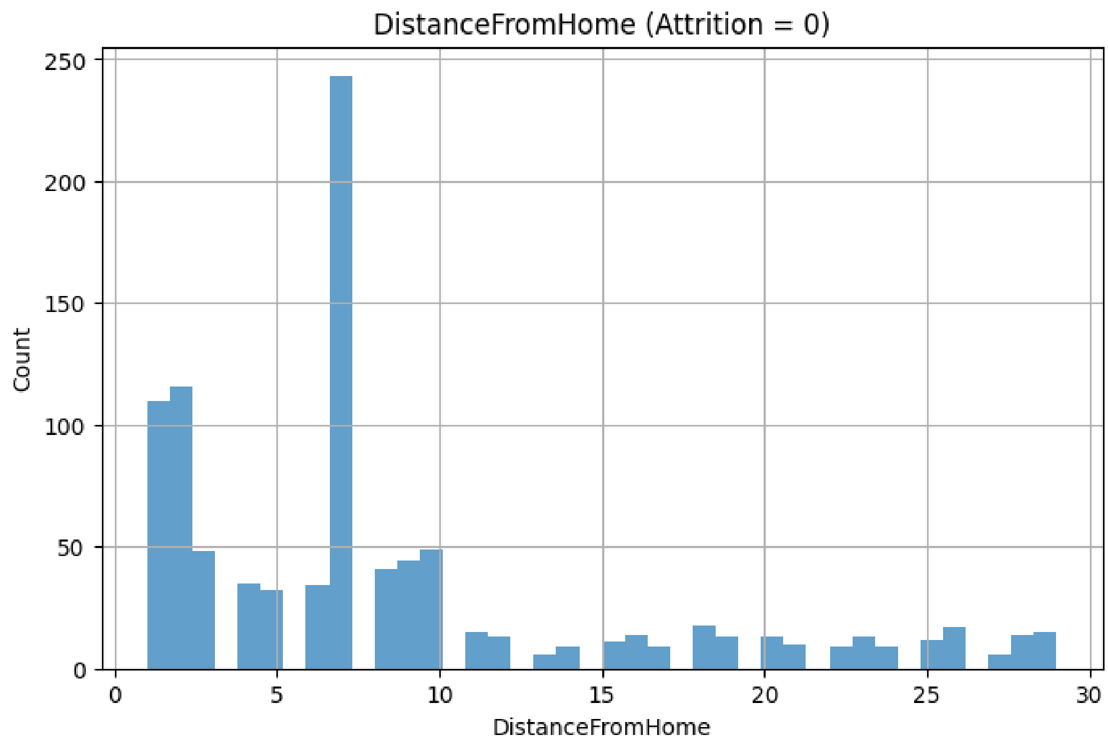

```
Number of bins with zero counts: 2
```



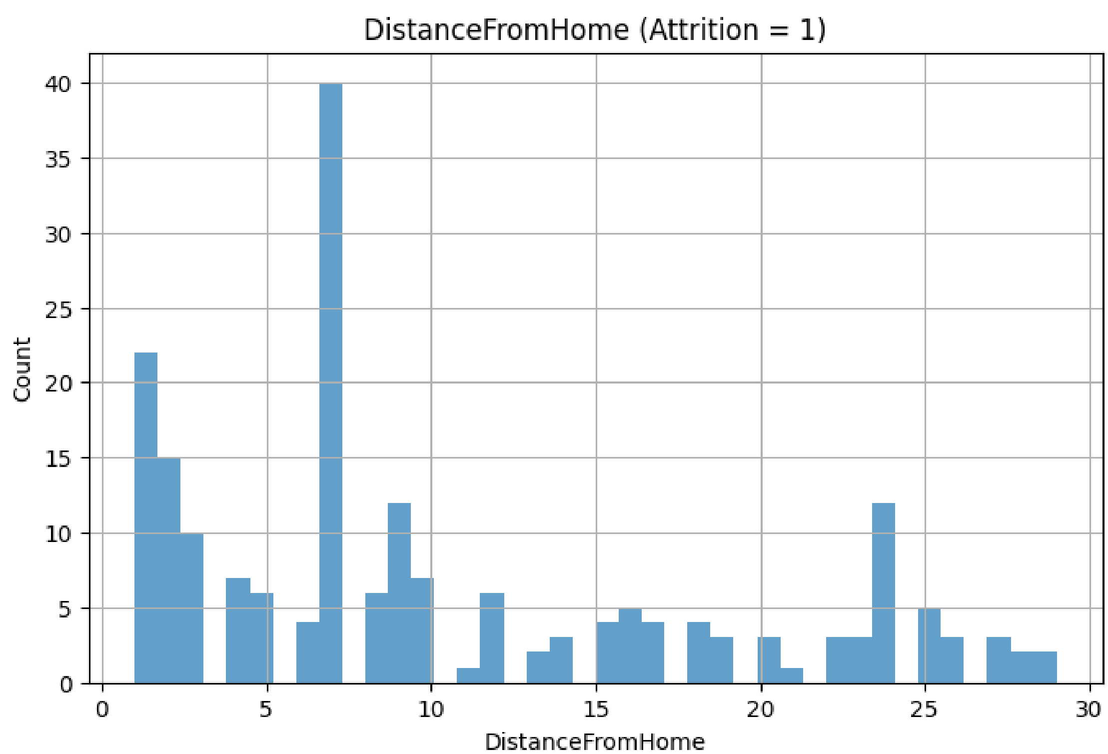
```
Number of bins with zero counts: 0
```




```
Number of bins with zero counts: 14
```




```
Number of bins with zero counts: 11
```

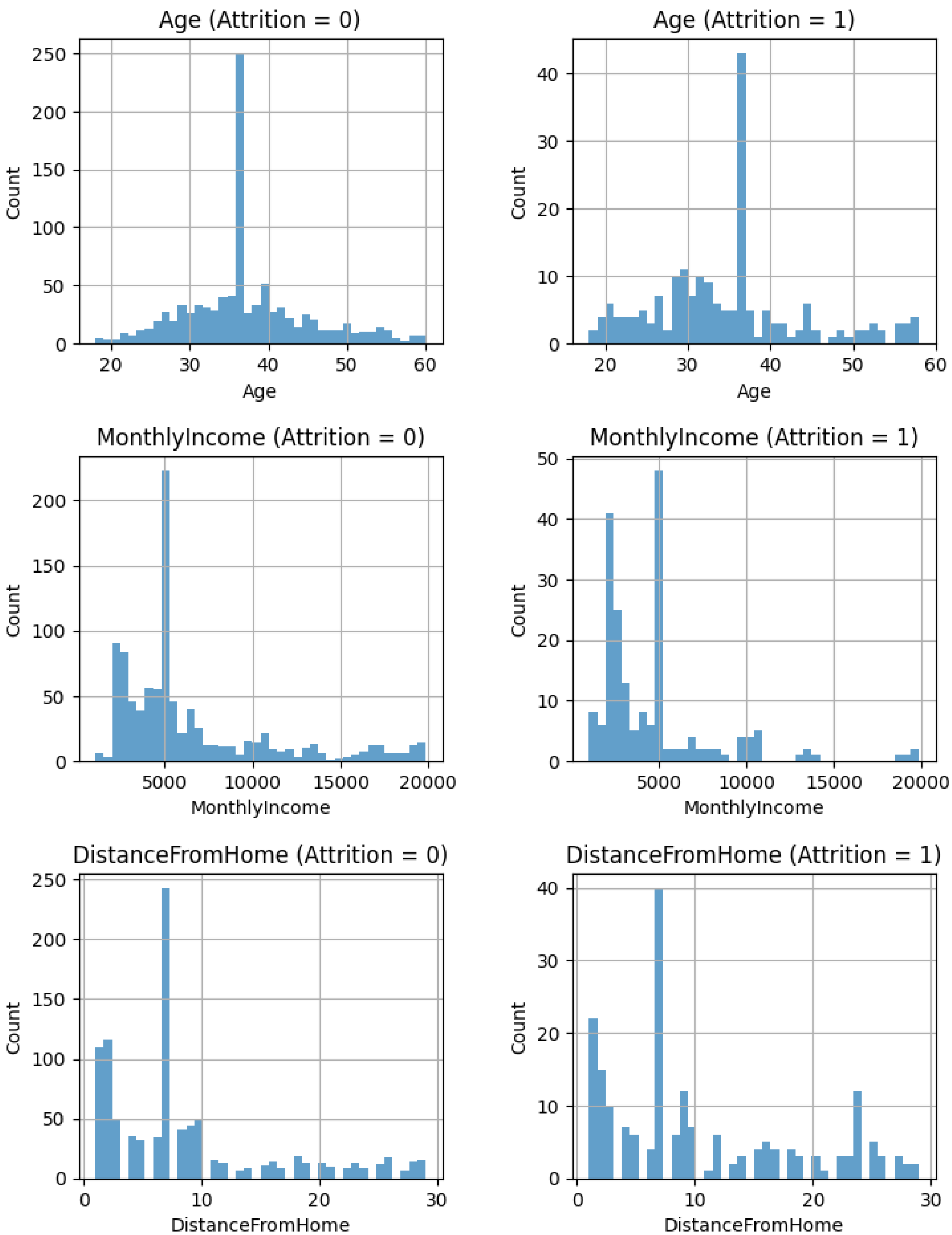


```
Number of bins with zero counts: 11
```

[10]:

```
# Make better plots
fig, axes = plt.subplots(3, 2, figsize=(8, 10))
fig.tight_layout(pad=4.0)

col_names = ["Age", "MonthlyIncome", "DistanceFromHome"]
attrition_classes = [0, 1]

for i, col_name in enumerate(col_names):
    for j, cls in enumerate(attrition_classes):
        ax = axes[i, j]
        data = df_train[df_train["Attrition"] == cls][col_name]

        counts, bins, _ = ax.hist(
            data,
            bins=40,
            alpha=0.7
        )

        zero_bins = np.sum(counts == 0)

        ax.set_xlabel(col_name)
        ax.set_ylabel("Count")
        ax.set_title(f"{col_name} (Attrition = {cls})")
        ax.grid(True)

        print(f"Number of bins with zero counts in {col_name} (Attrition =␣
↪{cls}): {zero_bins}")

plt.savefig("../images/p4.png", dpi=300)
plt.show()
```

```
Number of bins with zero counts in Age (Attrition = 0): 0
Number of bins with zero counts in Age (Attrition = 1): 2
Number of bins with zero counts in MonthlyIncome (Attrition = 0): 0
Number of bins with zero counts in MonthlyIncome (Attrition = 1): 14
Number of bins with zero counts in DistanceFromHome (Attrition = 0): 11
Number of bins with zero counts in DistanceFromHome (Attrition = 1): 11
```


Age (Attrition = 0)
Count
Age
Age (Attrition = 1)
Count
Age
MonthlyIncome (Attrition = 0)
Count
MonthlyIncome
MonthlyIncome (Attrition = 1)
Count
MonthlyIncome
DistanceFromHome (Attrition = 0)
Count
DistanceFromHome
DistanceFromHome (Attrition = 1)
Count
DistanceFromHome

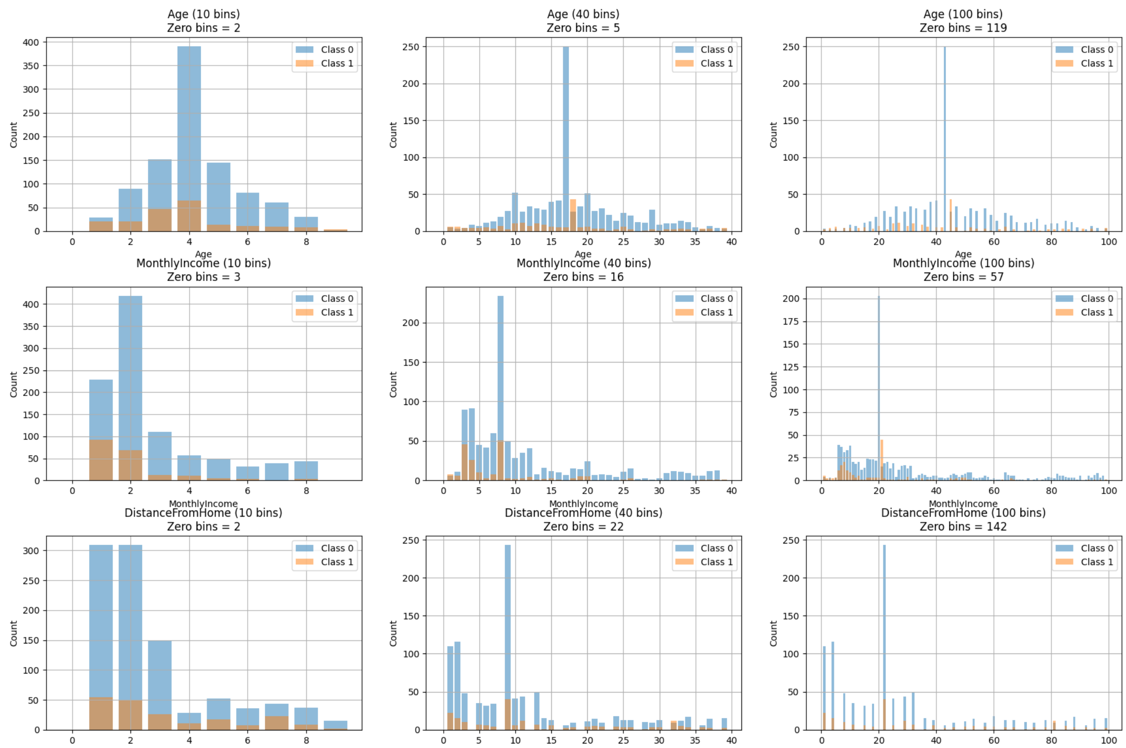
1.3.4 Problem T6

[11]:
```python
def discretize_feature(values, n_bins):
    min_val = np.min(values)
    max_val = np.max(values)

    # Internal bin edges (exclude -inf, +inf)
    bin_edges = np.linspace(min_val, max_val, n_bins - 1)

    # Assign bins
    bin_indices = np.digitize(values, bin_edges)

    # Count samples per bin
    counts = np.bincount(bin_indices, minlength=n_bins)

    return counts, bin_edges
```

[12]:
```python
# Make better plots
fig, axes = plt.subplots(3, 3, figsize=(18, 12))
fig.tight_layout(pad=4.0)

col_names = ["Age", "MonthlyIncome", "DistanceFromHome"]
bin_number_list = [10, 40, 100]
attrition_classes = [0, 1]

for i, col_name in enumerate(col_names):
    for j, n_bin in enumerate(bin_number_list):
        ax = axes[i, j]

        total_zero_bins = 0

        for cls in attrition_classes:
            data = df_train[df_train["Attrition"] == cls][col_name].values

            counts, bin_edges = discretize_feature(data, n_bin)

            # Plot as bar chart (manual histogram)
            ax.bar(
                range(n_bin),
                counts,
                width=0.8,
                alpha=0.5,
                label=f"Class {cls}"
            )

            zero_bins = np.sum(counts == 0)
            total_zero_bins += zero_bins
```

```
        ax.set_xlabel(col_name)
        ax.set_ylabel("Count")
        ax.set_title(f"{col_name} ({n_bin} bins)\nZero bins = {total_zero_bins}")
        ax.legend()
        ax.grid(True)

        print(
            f"Zero-count bins in {col_name} (bins={n_bin}): {total_zero_bins}"
        )

plt.savefig("../images/p6.png", dpi=300)
plt.show()
```

```
Zero-count bins in Age (bins=10): 2
Zero-count bins in Age (bins=40): 5
Zero-count bins in Age (bins=100): 119
Zero-count bins in MonthlyIncome (bins=10): 3
Zero-count bins in MonthlyIncome (bins=40): 16
Zero-count bins in MonthlyIncome (bins=100): 57
Zero-count bins in DistanceFromHome (bins=10): 2
Zero-count bins in DistanceFromHome (bins=40): 22
Zero-count bins in DistanceFromHome (bins=100): 142
```



1.3.5 T7. For the rest of the features, which one should be discretized in order to be modeled by histograms? What are the criteria for choosing whether we should discretize a feature or not? Answer this and discretize those features into 10 bins each. In other words, figure out the bin edge for each feature, then use digitize() to convert the features to discrete values

1.3.6 T8. What kind of distribution should we use to model histograms? (Answer a distribution name) What is the MLE for the likelihood distribution? (Describe how to do the MLE). Plot the likelihood distributions of MonthlyIncome, JobRole, HourlyRate, and MaritalStatus for different Attrition values.

1.3.7 T9. What is the prior distribution of the two classes?

1.3.8 T10. If we use the current Naive Bayes with our current Maximum Likelihood Estimates, we will find that some P (x i |attrition) will be zero and will result in the entire product term to be zero. Propose a method to fix this problem.

1.3.9 T11. Implement your Naive Bayes classifier. Use the learned distributions to classify the test set. Don't forget to allow your classifier to handle missing values in the test set. Report the overall Accuracy. Then, report the Precision, Recall, and F score for detecting attrition. See Lecture 1 for the definitions of each metric.

[13]:
```
from SimpleBayesClassifier import SimpleBayesClassifier
```

```
Traceback (most recent call last):

  File ~/study/cedt-2-2/2110573-pattern/CEDT-2110573-Pattern-Recognition/.venv/
↪lib/python3.10/site-packages/IPython/core/interactiveshell.py:3579 in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  Cell In[13], line 1
    from SimpleBayesClassifier import SimpleBayesClassifier

  File ~/study/cedt-2-2/2110573-pattern/CEDT-2110573-Pattern-Recognition/
↪homework-2/code/SimpleBayesClassifier.py:22
    self.n_pos =
                ^
SyntaxError: invalid syntax
```

[ ]:
```
data_train = df_train.to_numpy()
data_test = df_test.to_numpy()
```

[ ]:
```
x_train =
y_train =

x_test =
y_test =
```

[ ]:
```
model = SimpleBayesClassifier(n_pos = , n_neg = )
```

[ ]:
```
def check_prior():
    """
    This function designed to test the implementation of the prior probability␣
↪calculation in a Naive Bayes classifier.
    Specifically, it checks if the classifier correctly computes the prior␣
↪probabilities for the
    negative and positive classes based on given input counts.
    """

    # prior_neg = 5/(5 + 5) = 0.5 and # prior_pos = 5/(5 + 5) = 0.5
    assert (SimpleBayesClassifier(5, 5).prior_pos, SimpleBayesClassifier(5, 5).
↪prior_neg) == (0.5, 0.5)

    assert (SimpleBayesClassifier(3, 5).prior_pos, SimpleBayesClassifier(3, 5).
↪prior_neg) ==
    assert (SimpleBayesClassifier(0, 1).prior_pos, SimpleBayesClassifier(0, 1).
↪prior_neg) ==
    assert (SimpleBayesClassifier(1, 0).prior_pos, SimpleBayesClassifier(1, 0).
↪prior_neg) ==

check_prior()
```

[ ]:
```
model.fit_params(x_train, y_train)
```

[ ]:
```
def check_fit_params():

    """
    This function is designed to test the fit_params method of a␣
↪SimpleBayesClassifier.
    This method is presumably responsible for computing parameters for a Naive␣
↪Bayes classifier
    based on the provided training data. The parameters in this context is bins␣
↪and edges from each histogram.
    """

    T = SimpleBayesClassifier(2, 2)
    X_TRAIN_CASE_1 = np.array([
        [0, 1, 2, 3],
```

```
        [1, 2, 3, 4],
        [2, 3, 4, 5],
        [3, 4, 5, 6]
    ])
    Y_TRAIN_CASE_1 = np.array([0, 1, 0, 1])
    STAY_PARAMS_1, LEAVE_PARAMS_1 = T.fit_params(X_TRAIN_CASE_1, Y_TRAIN_CASE_1)

    print("STAY PARAMETERS")
    for f_idx in range(len(STAY_PARAMS_1)):
        print(f"Feature : {f_idx}")
        print(f"BINS : {STAY_PARAMS_1[f_idx][0]}")
        print(f"EDGES : {STAY_PARAMS_1[f_idx][1]}")
    print("")
    print("LEAVE PARAMETERS")
    for f_idx in range(len(STAY_PARAMS_1)):
        print(f"Feature : {f_idx}")
        print(f"BINS : {LEAVE_PARAMS_1[f_idx][0]}")
        print(f"EDGES : {LEAVE_PARAMS_1[f_idx][1]}")

check_fit_params()
```

[ ]:
```
y_pred = model.predict(x = x_test)
```

[ ]:
```
def evaluate(y_true, y_pred, show_result = True):

  return accuracy, precision, recall, F1, fpr
```

[ ]:
```
evaluate(y_test, y_pred)
```

1.3.10 T12. Use the learned distributions to classify the test set. Report the results using the same metric as the previous question.

[ ]:
```
model.fit_gaussian_params(x_train, y_train)
```

[ ]:
```
def check_fit_gaussian_params():

    """
    This function is designed to test the fit_gaussian_params method of a␣
 ↪SimpleBayesClassifier.
    This method is presumably responsible for computing parameters for a Naive␣
 ↪Bayes classifier
    based on the provided training data. The parameters in this context is mean␣
 ↪and STD.
    """

    T = SimpleBayesClassifier(2, 2)
```

```
    X_TRAIN_CASE_1 = np.array([
        [0, 1, 2, 3],
        [1, 2, 3, 4],
        [2, 3, 4, 5],
        [3, 4, 5, 6]
    ])
    Y_TRAIN_CASE_1 = np.array([0, 1, 0, 1])
    STAY_PARAMS_1, LEAVE_PARAMS_1 = T.fit_gaussian_params(X_TRAIN_CASE_1,␣
 ↪Y_TRAIN_CASE_1)

    print("STAY PARAMETERS")
    for f_idx in range(len(STAY_PARAMS_1)):
        print(f"Feature : {f_idx}")
        print(f"Mean : {STAY_PARAMS_1[f_idx][0]}")
        print(f"STD. : {STAY_PARAMS_1[f_idx][1]}")
    print("")
    print("LEAVE PARAMETERS")
    for f_idx in range(len(STAY_PARAMS_1)):
        print(f"Feature : {f_idx}")
        print(f"Mean : {LEAVE_PARAMS_1[f_idx][0]}")
        print(f"STD. : {LEAVE_PARAMS_1[f_idx][1]}")

check_fit_gaussian_params()
```

[ ]:
```
y_pred = model.gaussian_predict(x_test)
```

[ ]:
```
evaluate(y_test, y_pred)
```

1.3.11 T13 : The random choice baseline is the accuracy if you make a random guess for each test sample. Give random guess (50% leaving, and 50% staying) to the test samples. Report the overall Accuracy. Then, report the Precision, Recall, and F score for attrition prediction using the random choice baseline.

1.3.12 T14. The majority rule is the accuracy if you use the most frequent class from the training set as the classification decision. Report the overall Accuracy. Then, report the Precision, Recall, and F score for attrition prediction using the majority rule baseline.

1.3.13 T15. Compare the two baselines with your Naive Bayes classifier.

1.3.14 T16. Use the following threshold values

$ t = np.arange(-5,5,0.05) $ ### find the best accuracy, and F score (and the corresponding thresholds)

1.3.15 T17. Plot the RoC of your classifier.

1.3.16 T18. Change the number of discretization bins to 5. What happens to the RoC curve? Which discretization is better? The number of discretization bins can be considered as a hyperparameter, and must be chosen by comparing the final performance.
