Homework 2 MLE and Naive Bayes

MLE

T1 and OT1

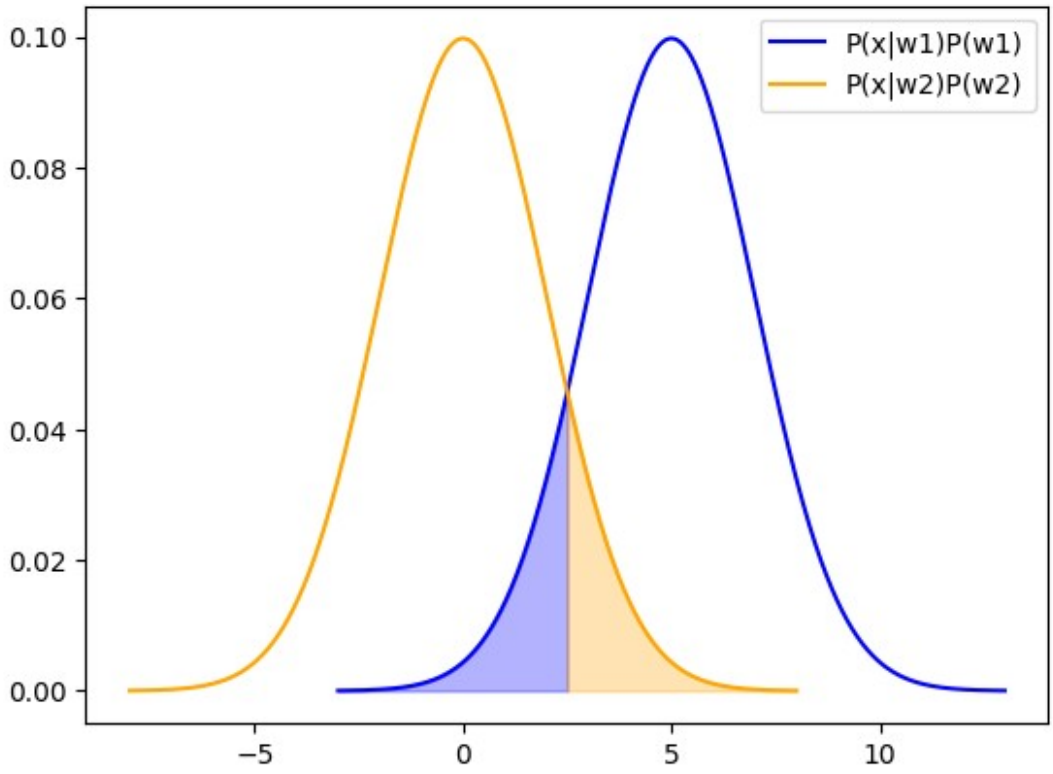
see the other file

Simple Bayes Classifier

T2

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from scipy import stats
import math
import matplotlib.pyplot as plt

def plot_bayes_classifier(mu_1, mu_2, vars_1, vars_2, p_w1, p_w2,
intersect_x=None):
  x_1 = np.linspace(mu_1 - 4*vars_1, mu_1 + 4*vars_1, 10000)
  y_1 = stats.norm.pdf(x_1, mu_1, vars_1) * p_w1
  x_2 = np.linspace(mu_2 - 4*vars_2, mu_2 + 4*vars_2, 10000)
  y_2 = stats.norm.pdf(x_2, mu_2, vars_2) * p_w2

  if(intersect_x is None):
    intersect_x = (2*vars_1**2*math.log(p_w2) -
2*vars_2**2*math.log(p_w1) + mu_1**2 - mu_2**2) / (2 * mu_1 - 2 *
mu_2)
  print("intersect: x = ", intersect_x)

  plt.plot(x_1, y_1, label='P(x|w1)P(w1)', color='blue')
  plt.fill_between(x_1[x_1 < intersect_x], y_1[x_1 < intersect_x],
color='blue', alpha=0.3)
  plt.plot(x_2, y_2, label='P(x|w2)P(w2)', color='orange')
  plt.fill_between(x_2[x_2 > intersect_x], y_2[x_2 > intersect_x],
color='orange', alpha=0.3)
  plt.legend()
  plt.show()

vars = 2
mu_1 = 5
mu_2 = 0
p_w1 = 0.5
p_w2 = 0.5
plot_bayes_classifier(mu_1, mu_2, vars, vars, p_w1, p_w2)
```

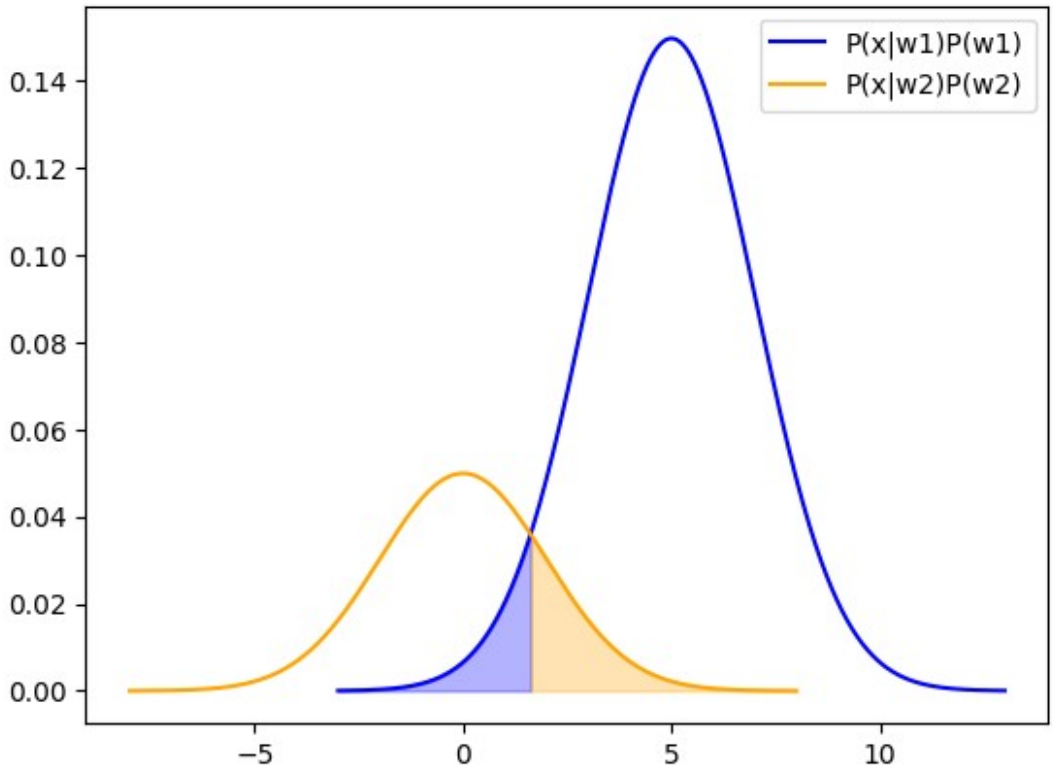
```
intersect: x =  2.5
```



T3

```
vars = 2
mu_1 = 5
mu_2 = 0
p_w1 = 0.75
p_w2 = 0.25
plot_bayes_classifier(mu_1, mu_2, vars, vars, p_w1, p_w2)
```

```
intersect: x =  1.6211101690655123
```



OT2

see the other file

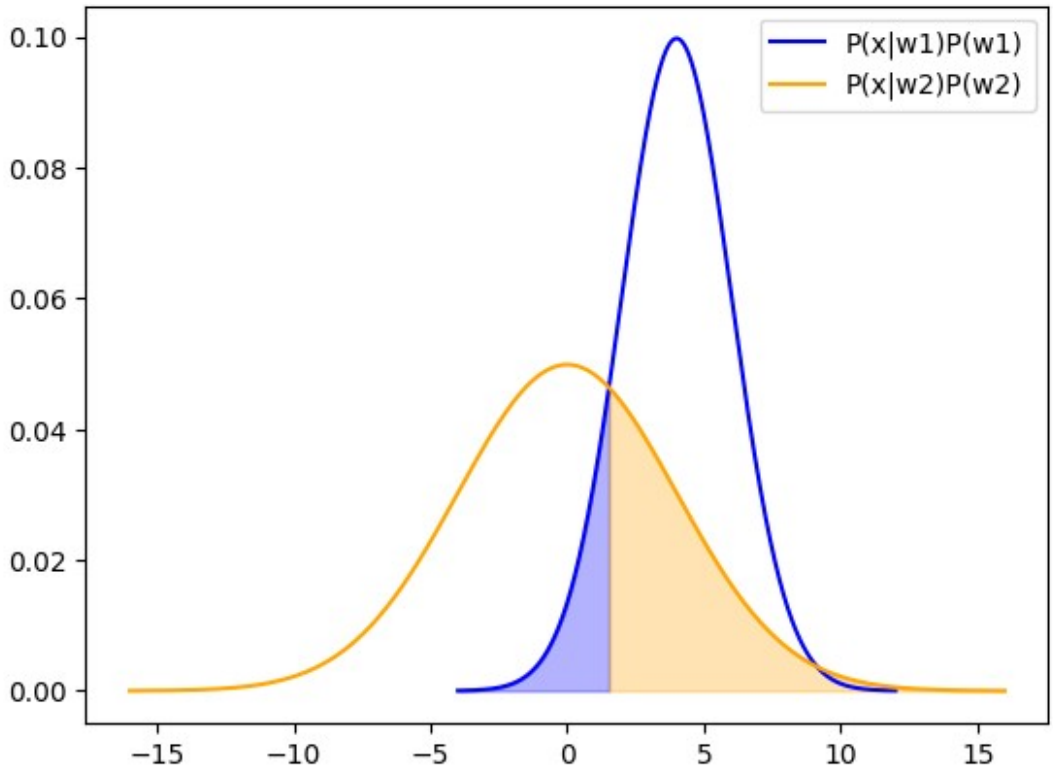
OT3

Find new intersect_x

From P(x|w1) = P(x|w2) 1/sqrt(2*pi)/2*exp(-1/2*((x-4)^2)/2^2) = 1/sqrt(2*pi)/4*exp(-1/2*((x-0)^2)/4^2) x = -4 / 3 * (-4 + sqrt(4 + log(64)))

```
vars1 = 2
vars2 = 4
mu_1 = 4
mu_2 = 0
p_w1 = 0.5
p_w2 = 0.5
intersect_x =  -4 / 3 * (-4 + math.sqrt(4 + math.log(64)))
plot_bayes_classifier(mu_1, mu_2, vars1, vars2, p_w1, p_w2,
intersect_x=intersect_x)
```

```
intersect: x =  1.5248321797187168
```



Employee Attrition Prediction

read CSV

```
df = pd.read_csv('hr-employee-attrition-with-null.csv')
```

Dataset statistic

```
df.describe()
```

```
        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]
```

```
df.head()
```

```
   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]
```

Feature transformation

```
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] = pd.Categorical(df[col]).codes

# HANDLE NULL NUMBERS
# INSERT CODE HERE
for col in df.columns:
    if df[col].dtype == np.float64:
        df[col] = df[col].fillna(df[col].mean())
    else:
        df[col] = df[col].fillna(df[col].mode()[0])


df = df.loc[:, ~df.columns.isin(['EmployeeNumber', 'Unnamed: 0',
'EmployeeCount', 'StandardHours', 'Over18'])]
```

Spliting data into train and test

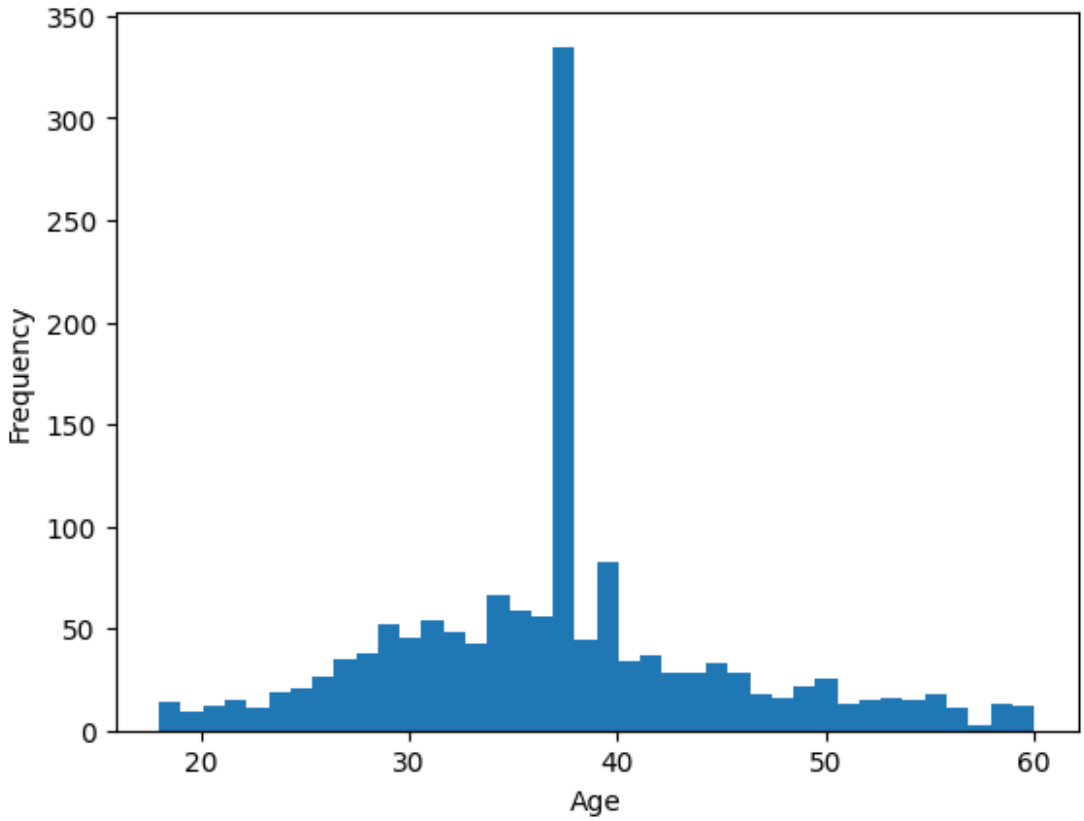
```
df_train, df_test = train_test_split(df, test_size=0.1,
stratify=df['Attrition'], random_state=42)
```

Display histogram of each feature

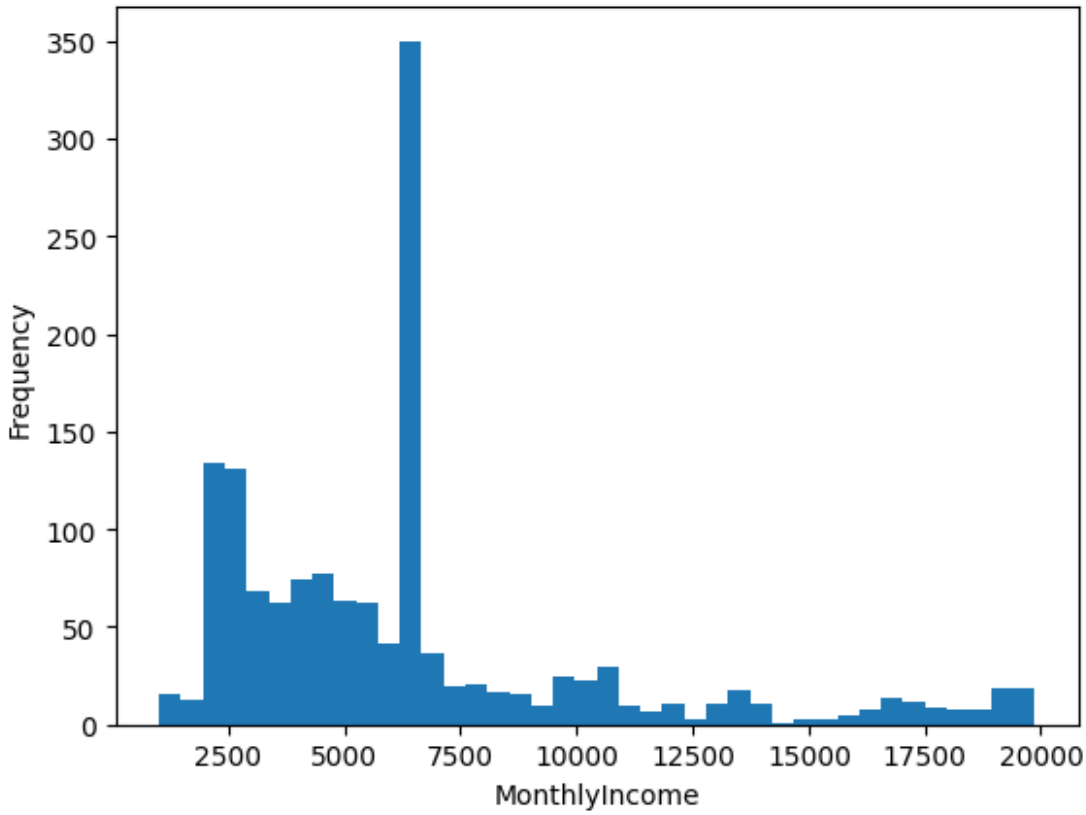
```
def display_histogram(df, feature, n_bin = 40):

        df_dropna = df[feature].dropna()
        plt.hist(df_dropna, n_bin)
        plt.xlabel(feature)
        plt.ylabel("Frequency")
        plt.show()
    # INSERT CODE HERE
```

T4. Observe the histogram for Age, MonthlyIncome and DistanceFromHome. How many bins have zero counts? Do you think this is a good discretization? Why?

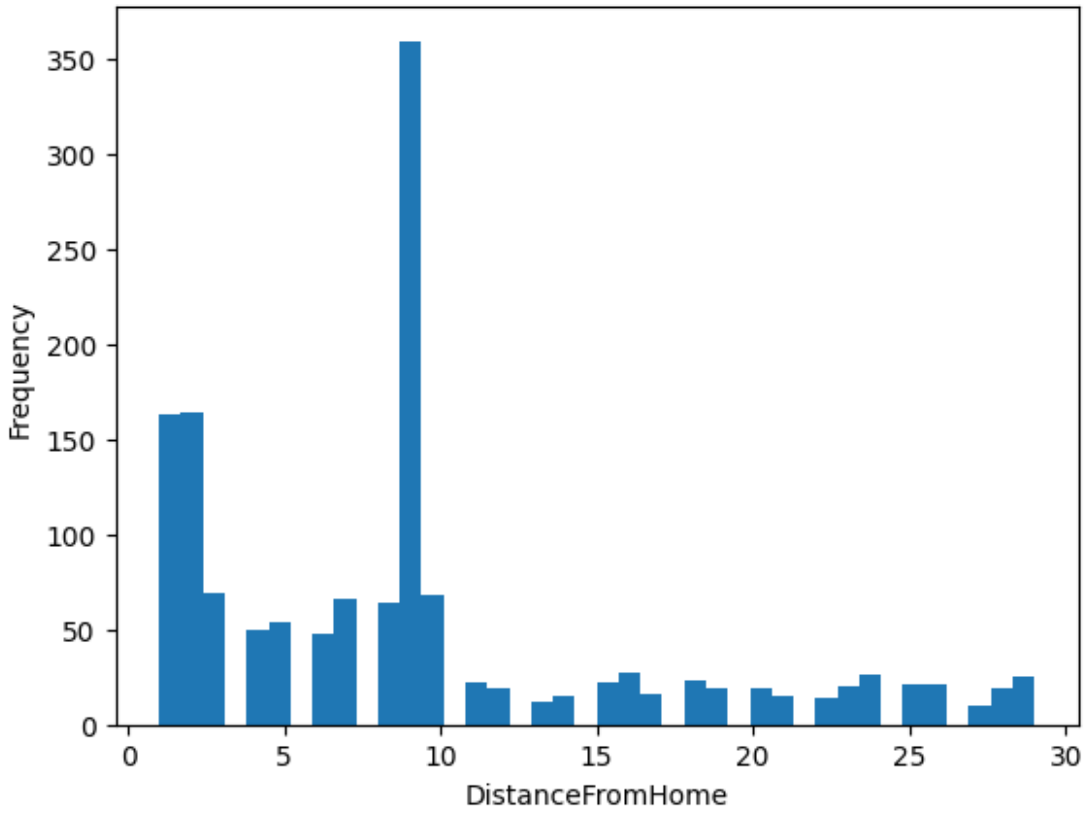
```
def count_zero_bins(df, col_name, n_bins):
    df_drop_na = df[col_name].dropna()
    hist, bin__edge = np.histogram(df_drop_na, bins=n_bins)
    return np.count_nonzero(hist == 0)

features = ["Age", "MonthlyIncome", "DistanceFromHome"]
for feature in features:
    display_histogram(df, feature)
    print("Zero bin count of", feature, count_zero_bins(df, feature,
n_bins=40))
```




```
Zero bin count of Age 0
```




```
Zero bin count of MonthlyIncome 0
```




```
Zero bin count of DistanceFromHome 11
```

From having no zero bins, the Age and MonthlyIncome features are good discretiztion.

While, the DistanceFromHome feature is not.

T5. Can we use a Gaussian to estimate this histogram? Why? What about a Gaussian Mixture Model (GMM)?

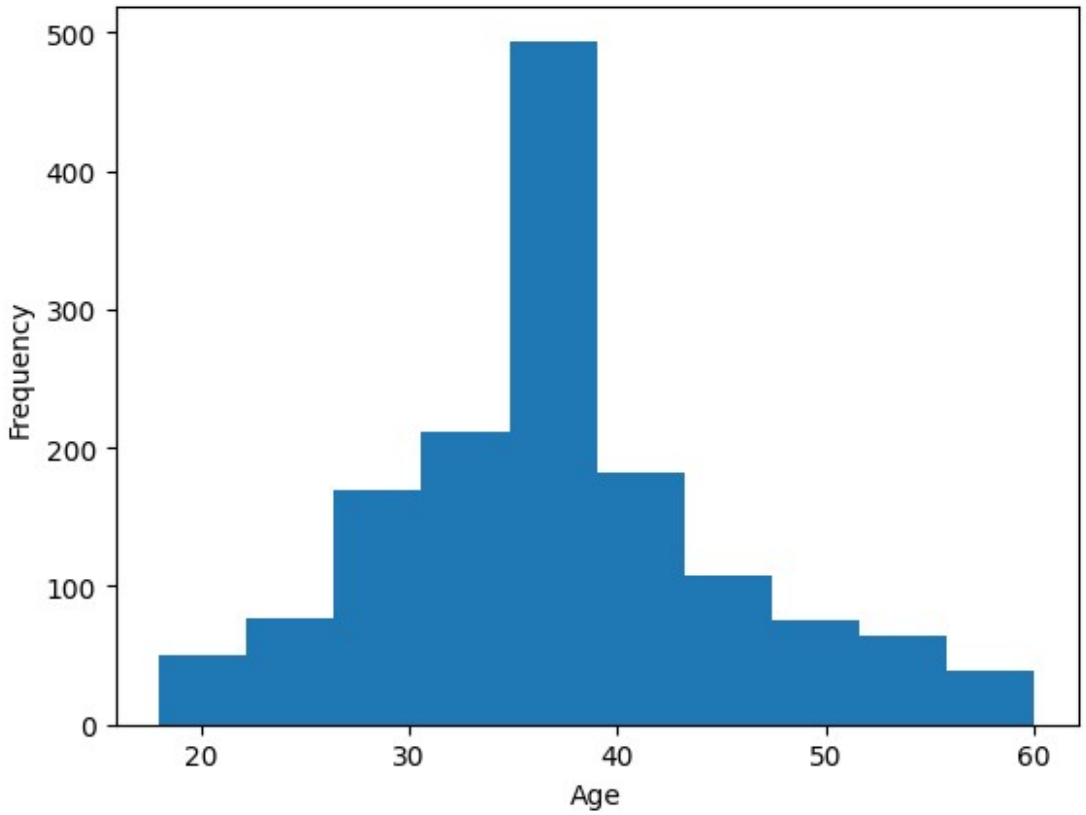
Only the Age feature one can be estimated as Guassian since the rest don't look like a Normal Distribution.

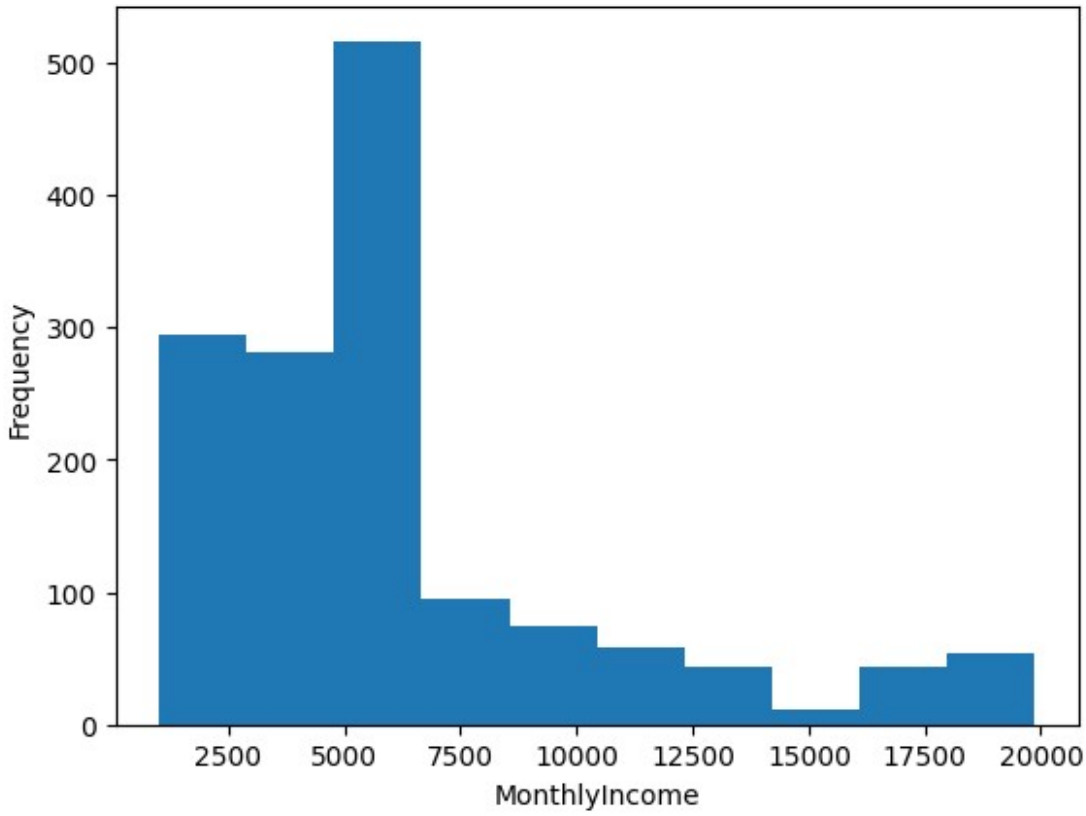
The MonthlyIncome and DistanceFromHome features may be estimated using GMM or other distributions.

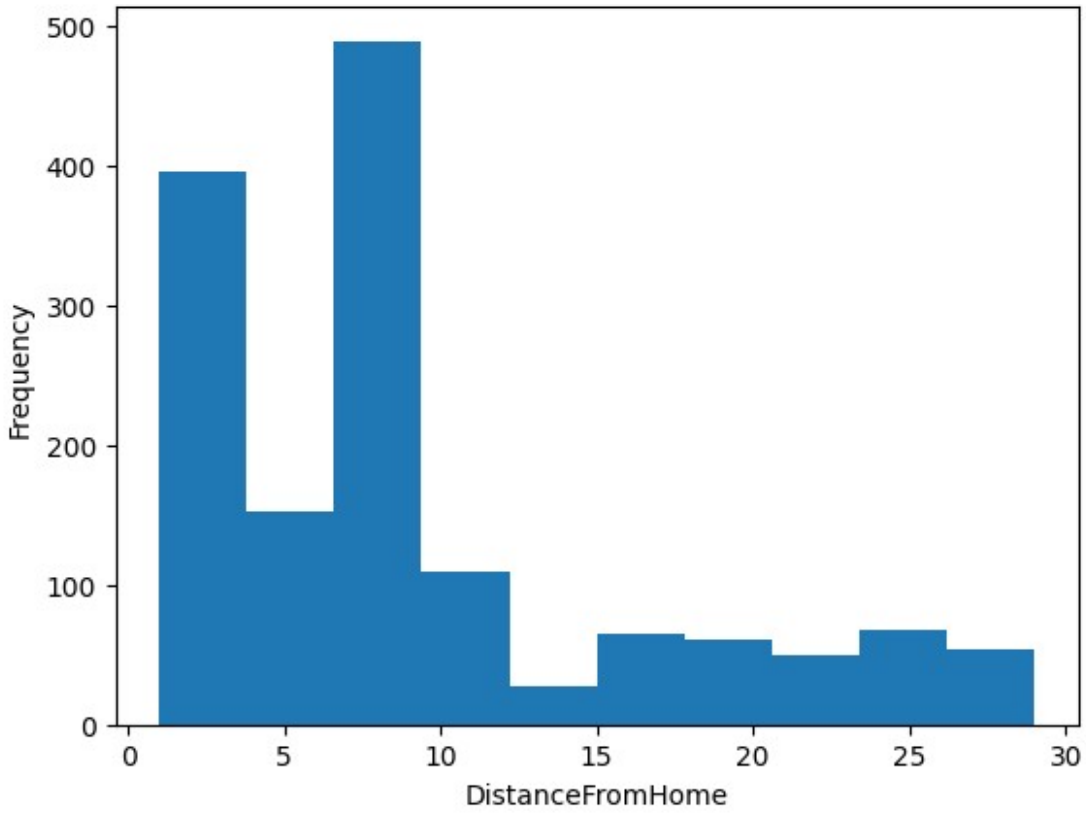
T6. Now plot the histogram according to the method described above (with 10, 40, and 100 bins) and show 3 plots each for Age, MonthlyIncome, and DistanceFromHome. Which bin size is most sensible for each features? Why?

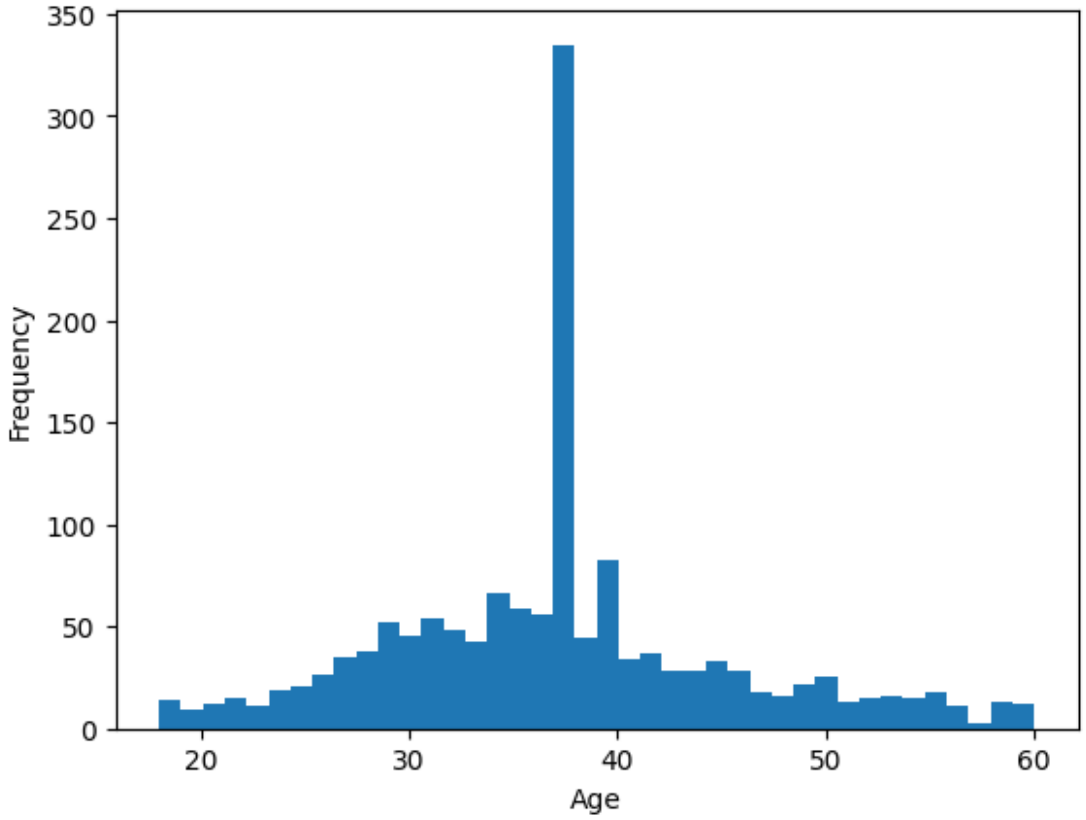
```
bins = [10, 40, 100]

for bin in bins:
```

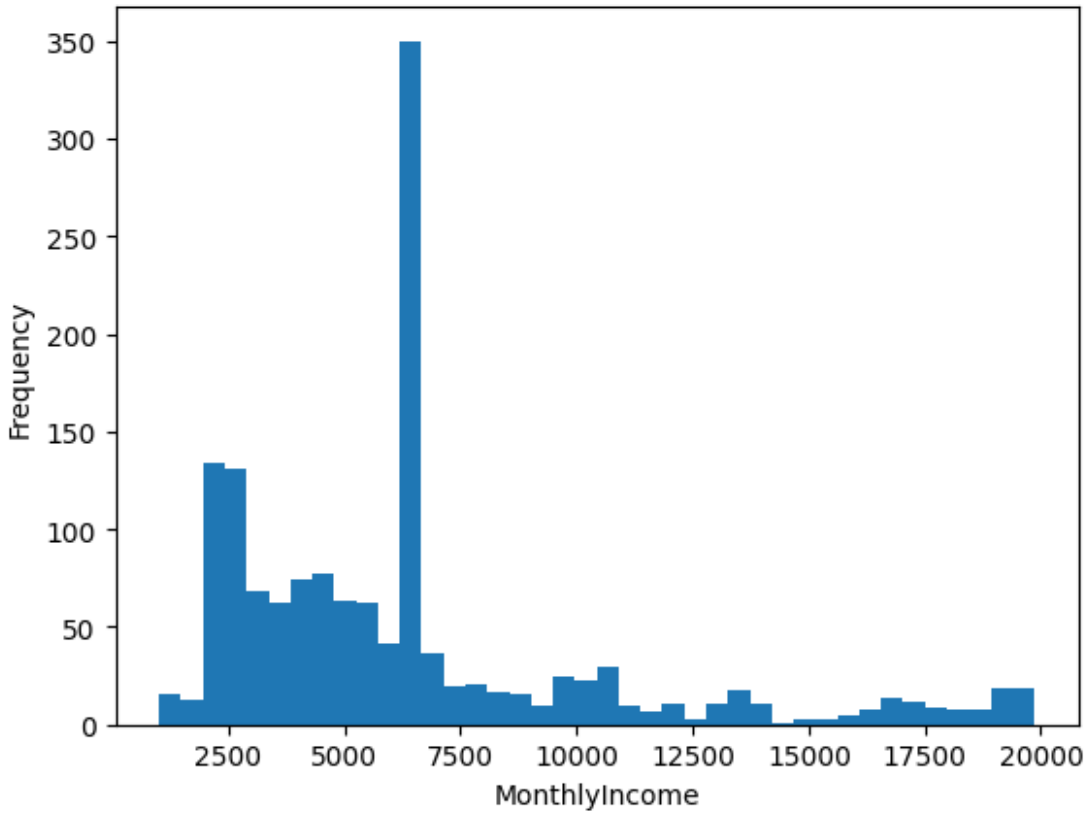
```
    for feature in features:
        display_histogram(df, feature, n_bin=bin)
```

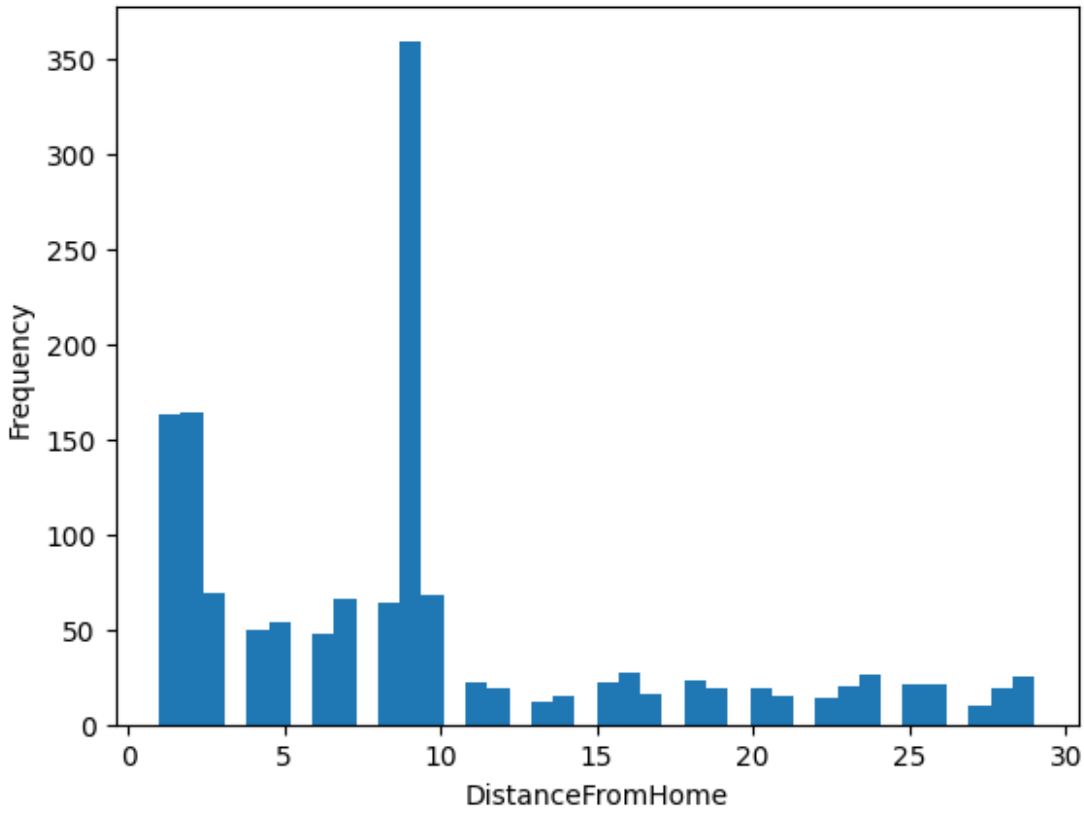


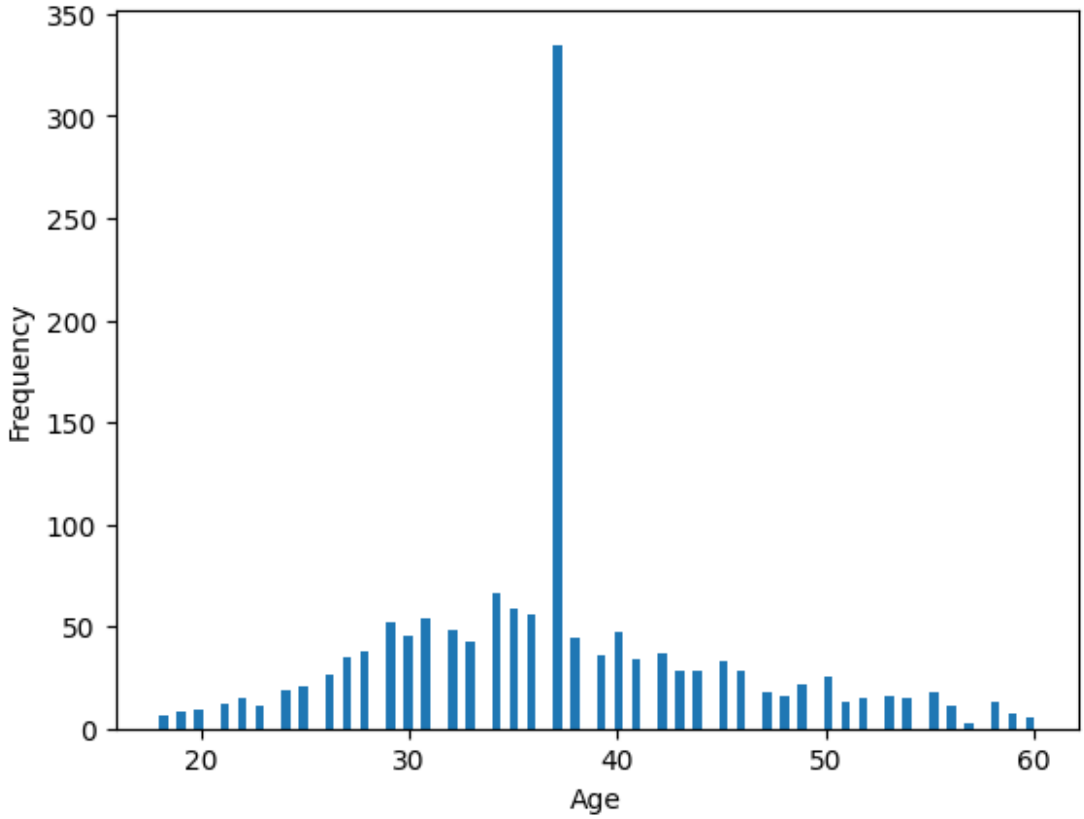

Frequency
0
100
200
300
400
500
2500
5000
7500
10000
12500
15000
17500
20000
MonthlyIncome

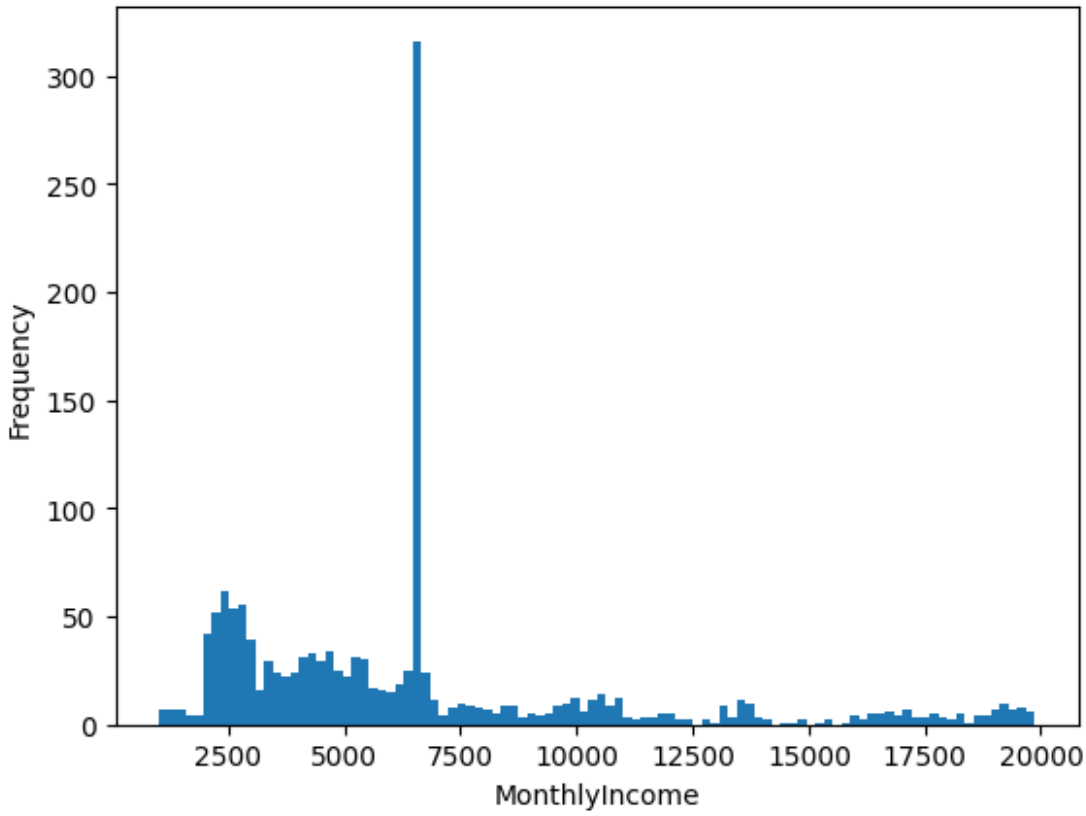

Frequency
0
100
200
300
400
500
0
5
10
15
20
25
30
DistanceFromHome

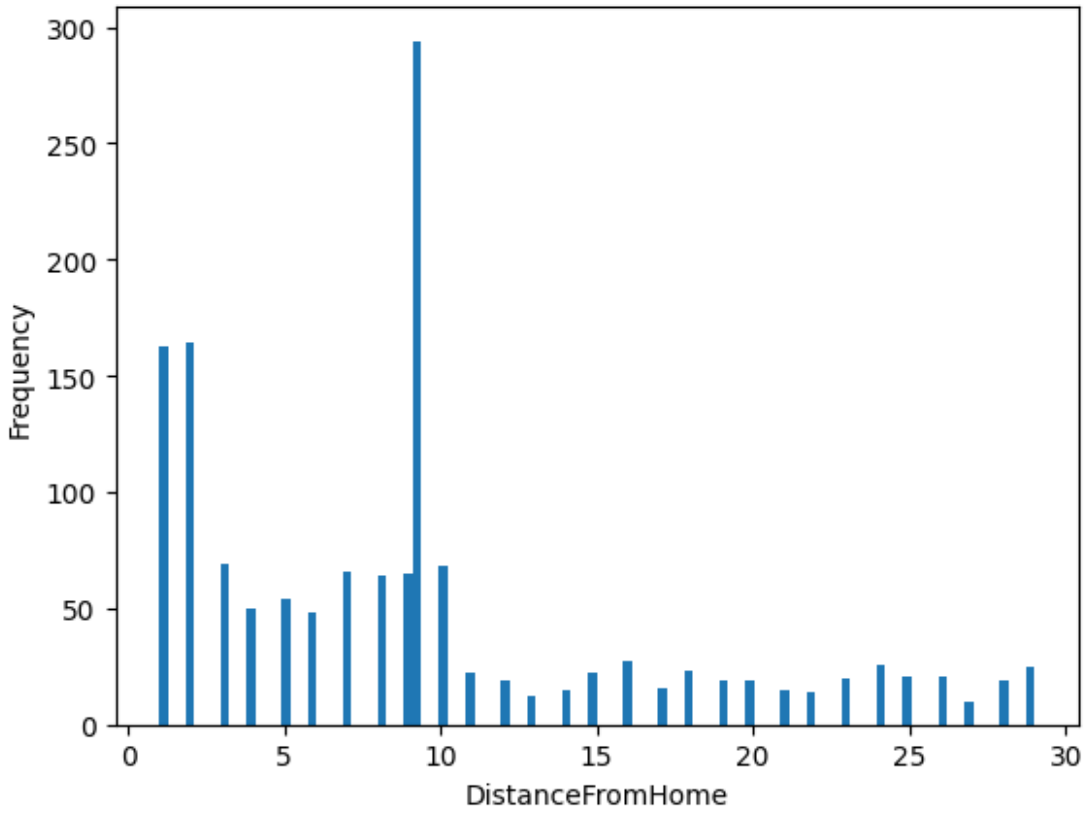

350
300
250
200
150
100
50
0
Frequency
20
30
40
50
60
Age


350
300
250
200
150
100
50
0
Frequency
2500
5000
7500
10000
12500
15000
17500
20000
MonthlyIncome


Frequency
DistanceFromHome
0
50
100
150
200
250
300
350
0
5
10
15
20
25
30


Frequency
Age
0
50
100
150
200
250
300
350
20
30
40
50
60


300
250
200
150
100
50
0
Frequency
2500
5000
7500
10000
12500
15000
17500
20000
MonthlyIncome


300
250
200
150
100
50
0
Frequency
0
5
10
15
20
25
30
DistanceFromHome

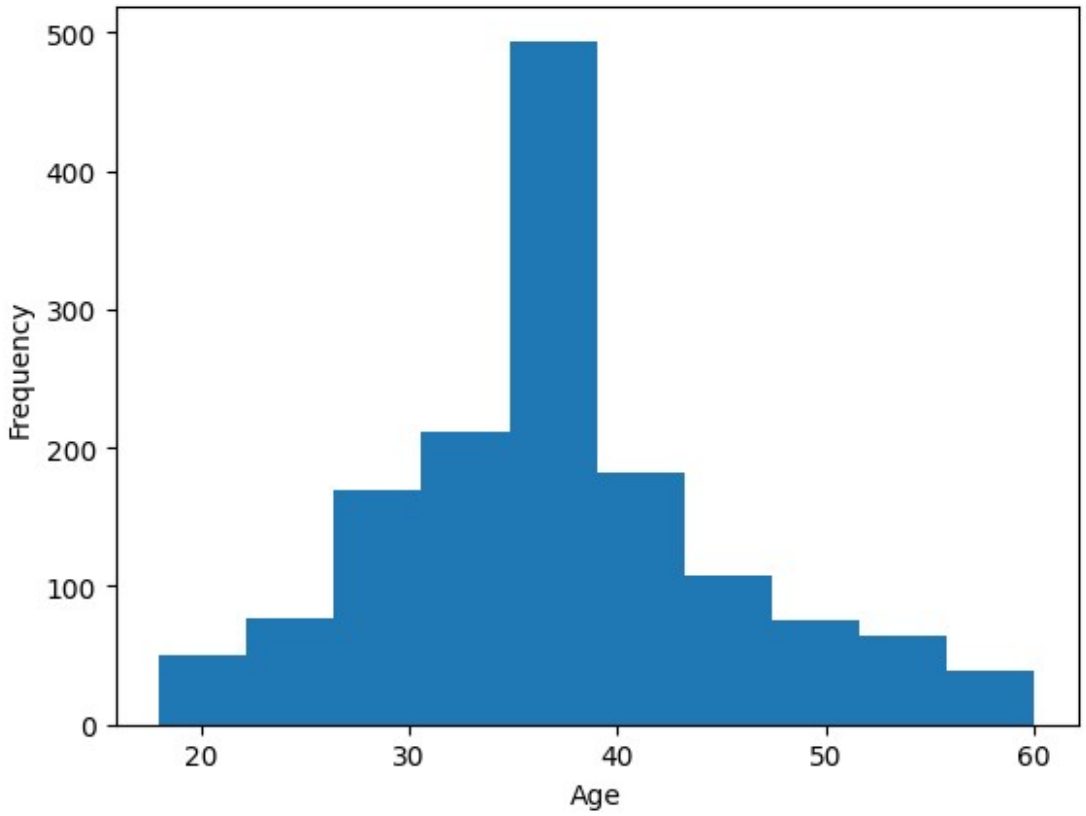
For Age, the appropriate bins would be 10 because it can accurately describe the distribution of data. Not too rough and not too detailed.

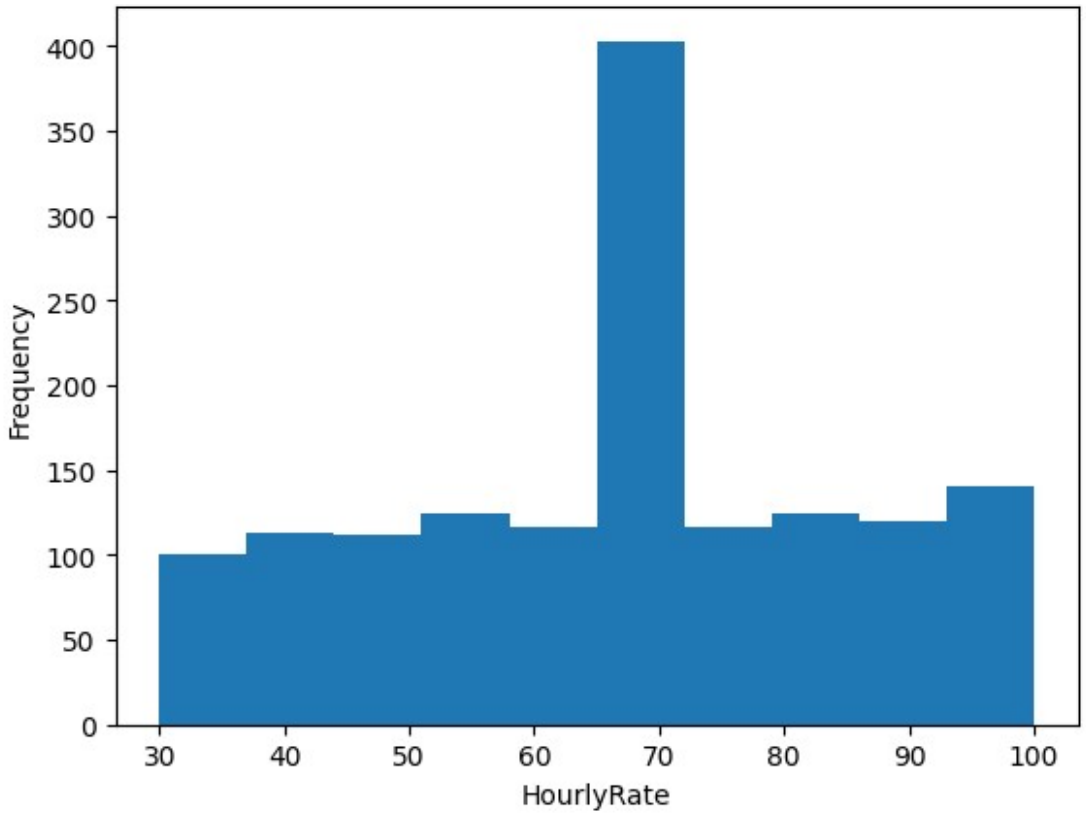
For MonthlyIncome, the optimal bins are 40 because the data is highly distrubuted.

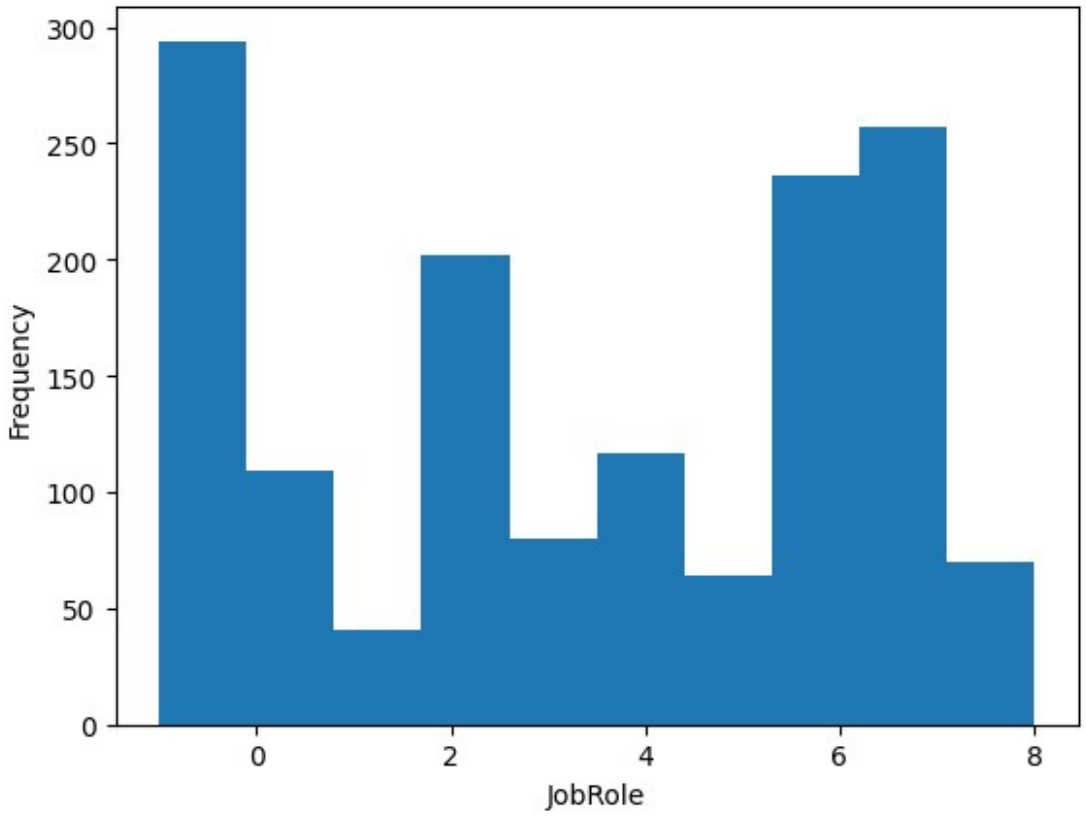
For DistanceFromHome, the appropriate bins are 10 because the DistanceFromHome data Will only be in a narrow range, no need to use many bins

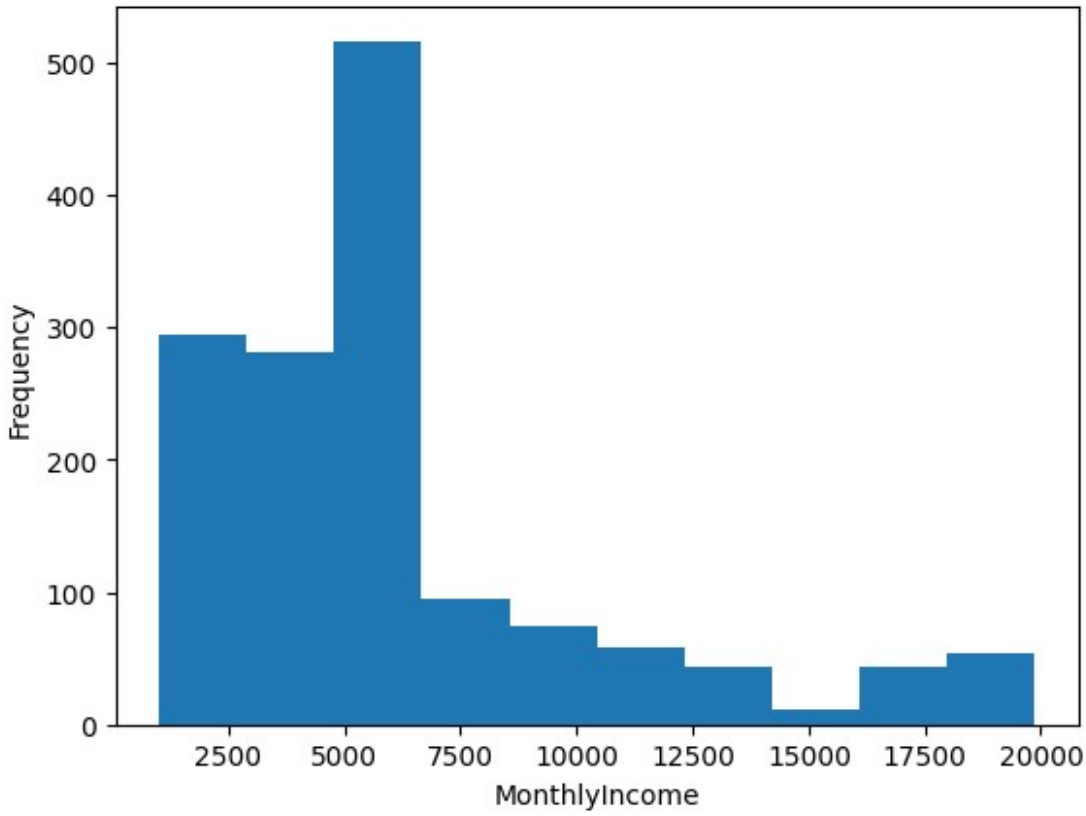
T7. For the rest of the features, which one should be discretized in order to be modeled by histograms? What are the criteria for choosing whether we should discretize a feature or not? Answer this and discretize those features into 10 bins each. In other words, figure out the bin edge for each feature,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

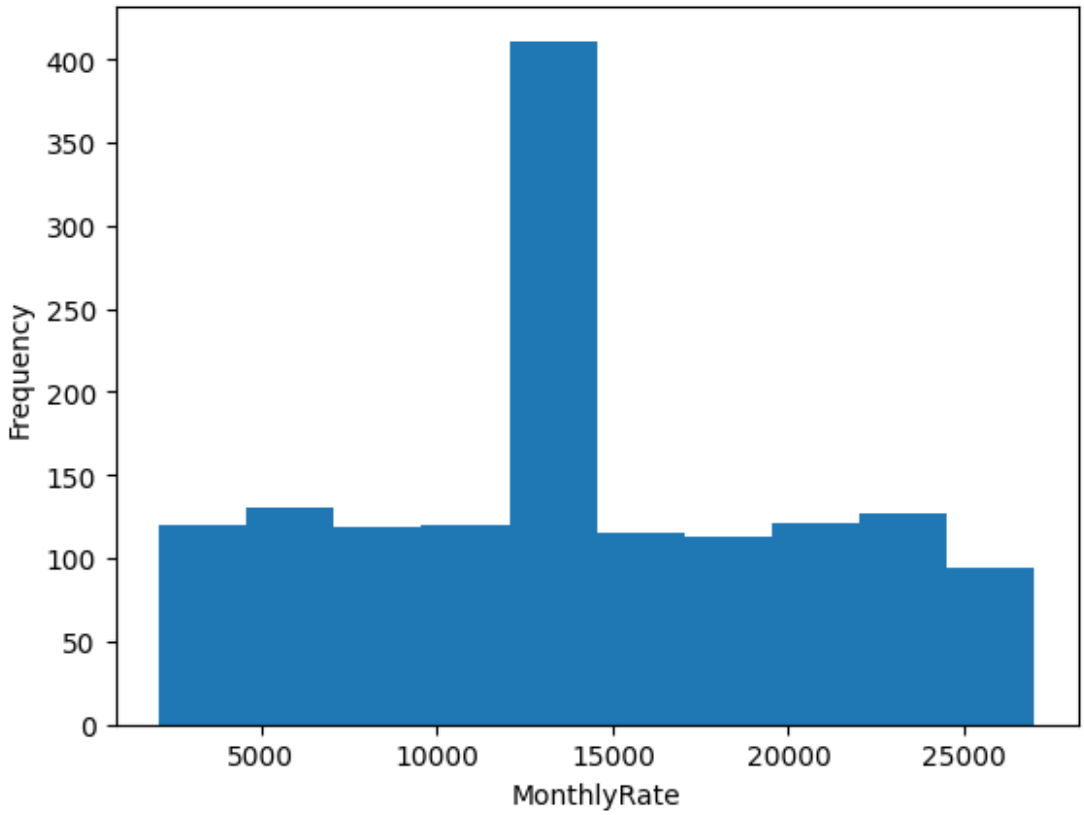
```
features = df.columns.to_list()
features.remove("Attrition")
for feature in features:
    if len(df[feature].unique()) < 10:
        features.remove(feature)
        continue
    display_histogram(df, feature, 10)

print(features)
```

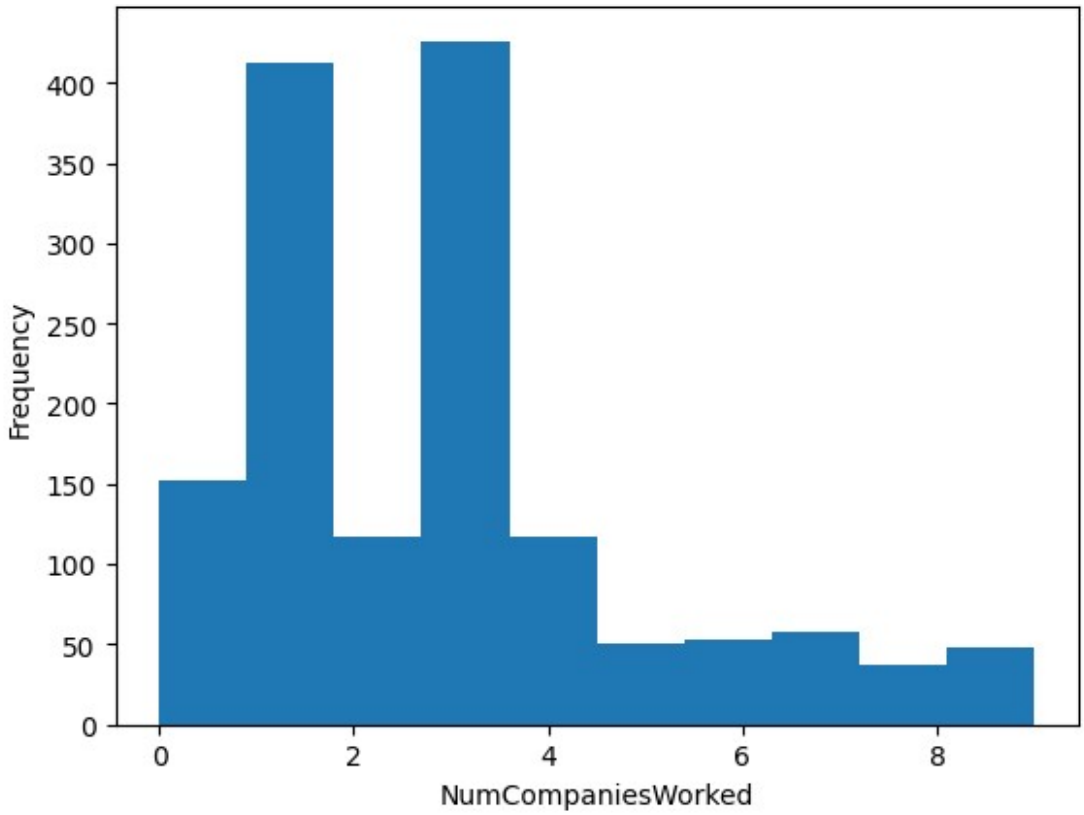

500
400
300
200
100
0
Frequency
20
30
40
50
60
Age

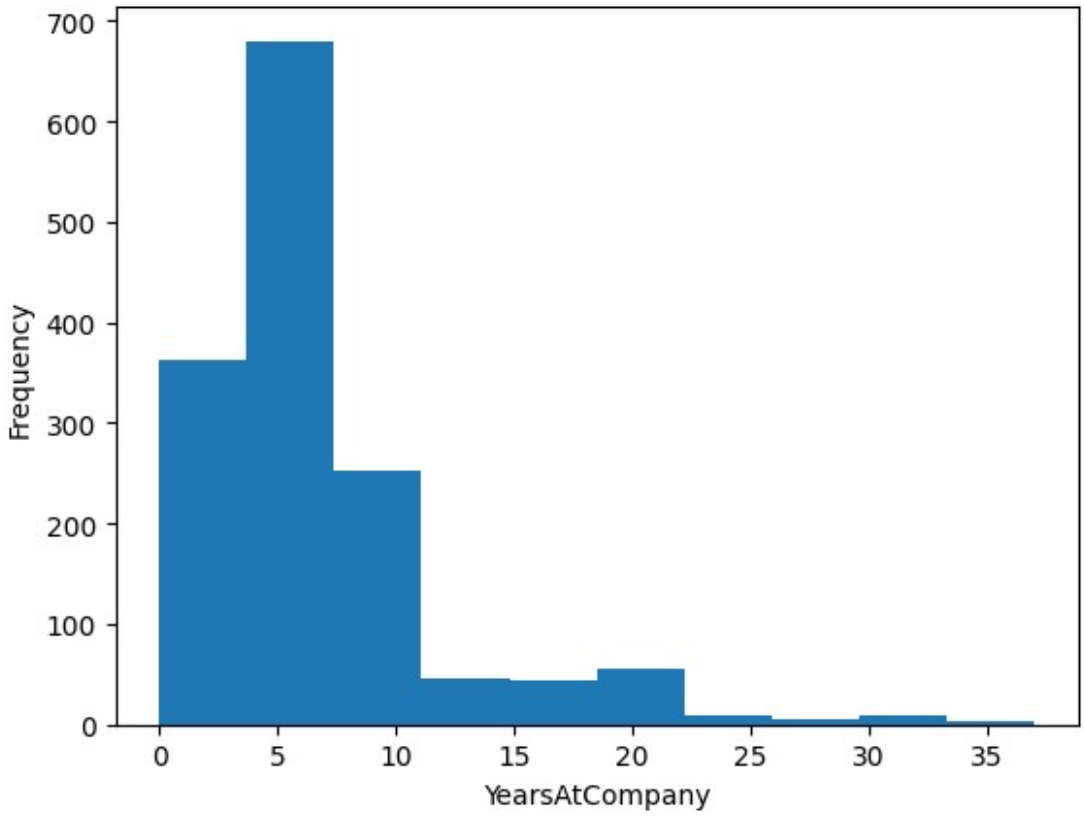

400
350
300
250
200
150
100
50
0
Frequency
30
40
50
60
70
80
90
100
HourlyRate

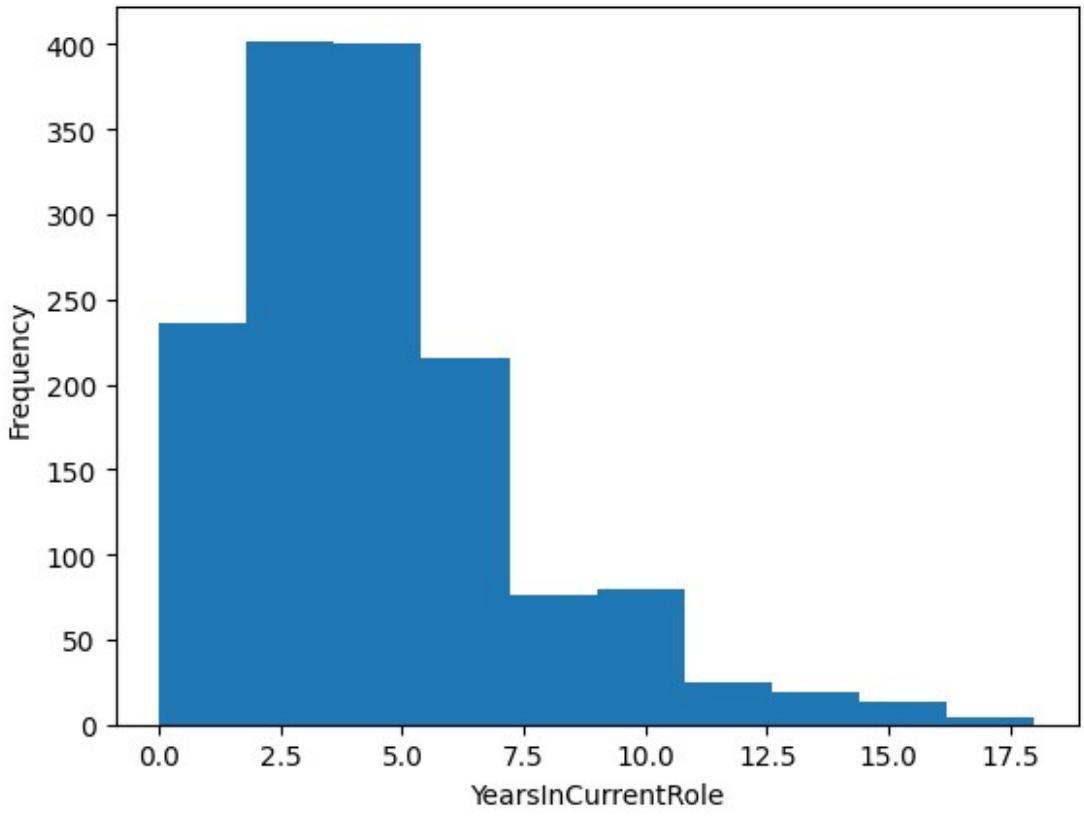

300
250
200
150
100
50
0
Frequency
0
2
4
6
8
JobRole

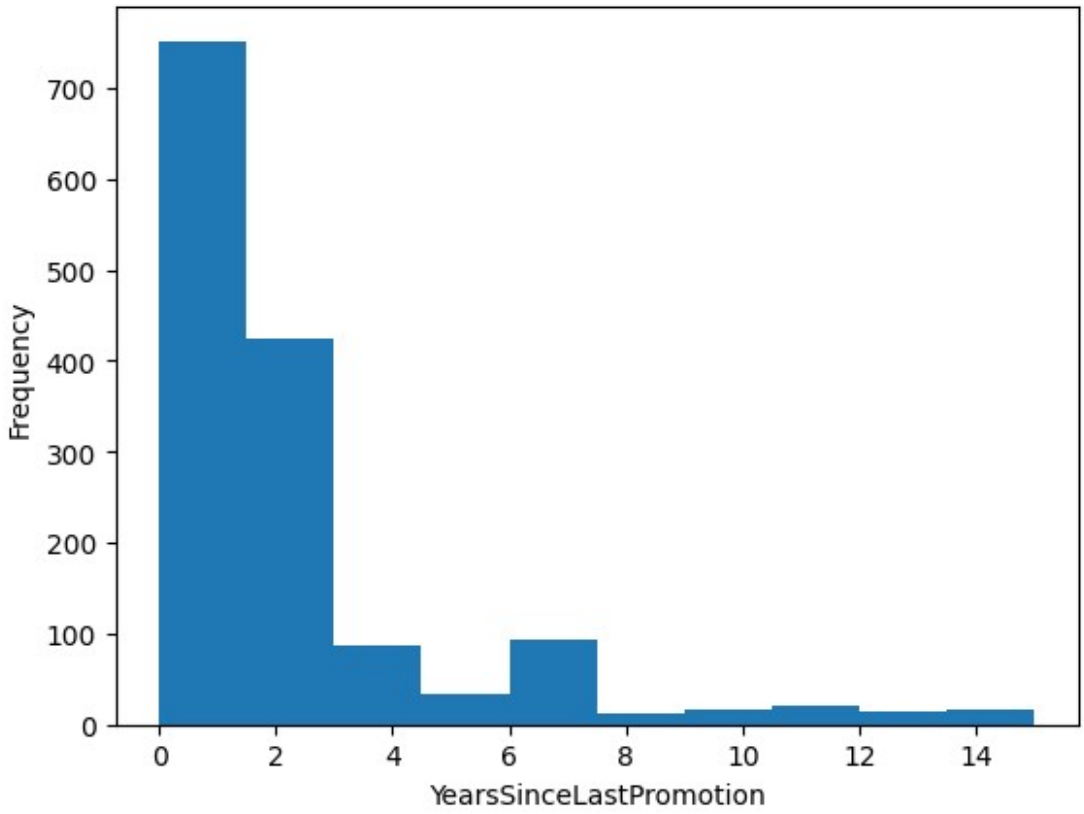

500
400
300
200
100
0
Frequency
2500
5000
7500
10000
12500
15000
17500
20000
MonthlyIncome

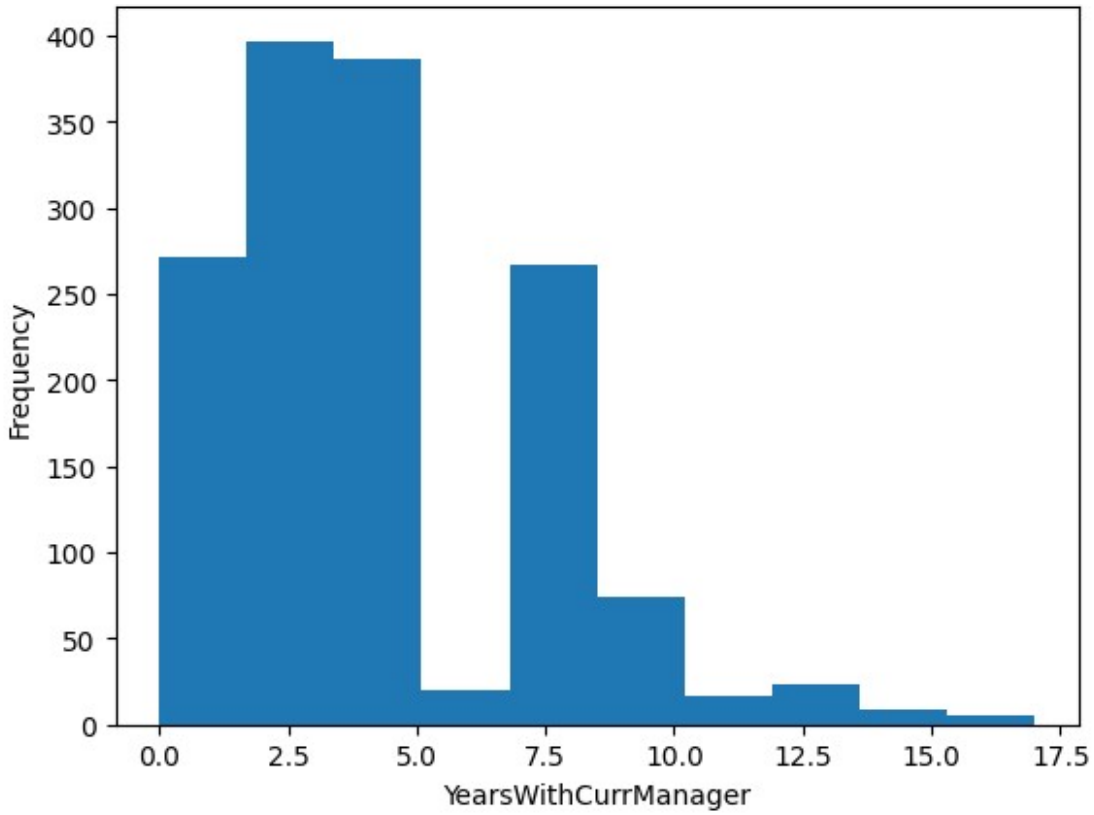

Frequency
0
50
100
150
200
250
300
350
400
5000
10000
15000
20000
25000
MonthlyRate


Frequency
0
50
100
150
200
250
300
350
400
0
2
4
6
8
NumCompaniesWorked


Frequency
YearsAtCompany
0
100
200
300
400
500
600
700
0
5
10
15
20
25
30
35


Frequency
YearsInCurrentRole
0
50
100
150
200
250
300
350
400
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Frequency
700
600
500
400
300
200
100
0
0
2
4
6
8
10
12
14
YearsSinceLastPromotion


Frequency
400
350
300
250
200
150
100
50
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
YearsWithCurrManager

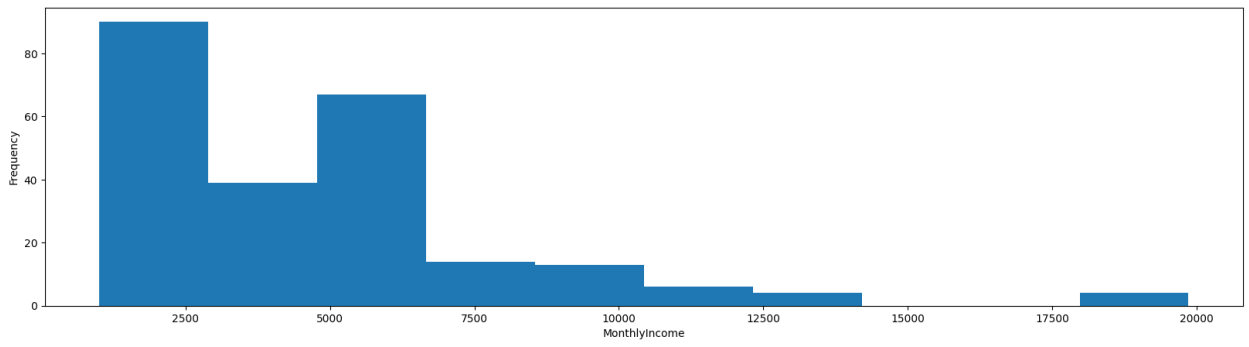
```
['Age', 'DailyRate', 'DistanceFromHome', 'EducationField', 'Gender',
'HourlyRate', 'JobLevel', 'JobRole', 'MaritalStatus', 'MonthlyIncome',
'MonthlyRate', 'NumCompaniesWorked', 'PercentSalaryHike',
'RelationshipSatisfaction', 'TotalWorkingYears', 'WorkLifeBalance',
'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
'YearsWithCurrManager']
```

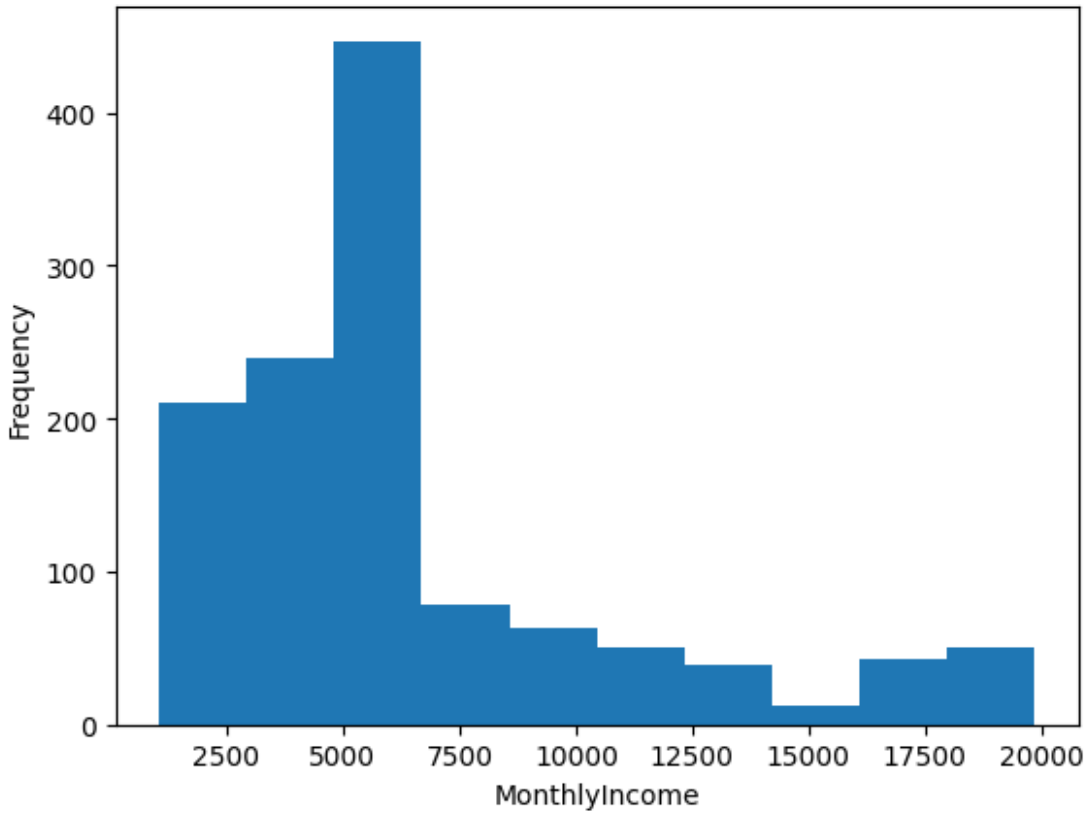
T8. What kind of distribution should we use to model histograms? (Answer a distribution name) What is the MLE for the likelihood distribution? (Describe how to do the MLE). Plot the likelihood distributions of MonthlyIncome, JobRole, HourlyRate, and MaritalStatus for different Attrition values.

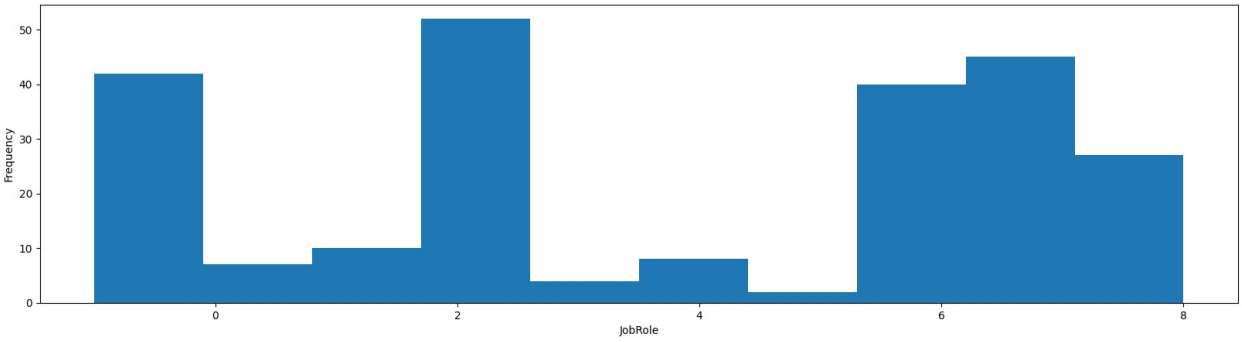
If consider every features, Categorical Distribution because it is a discrete probability distribution whose sample space is the set of k individually identified items.

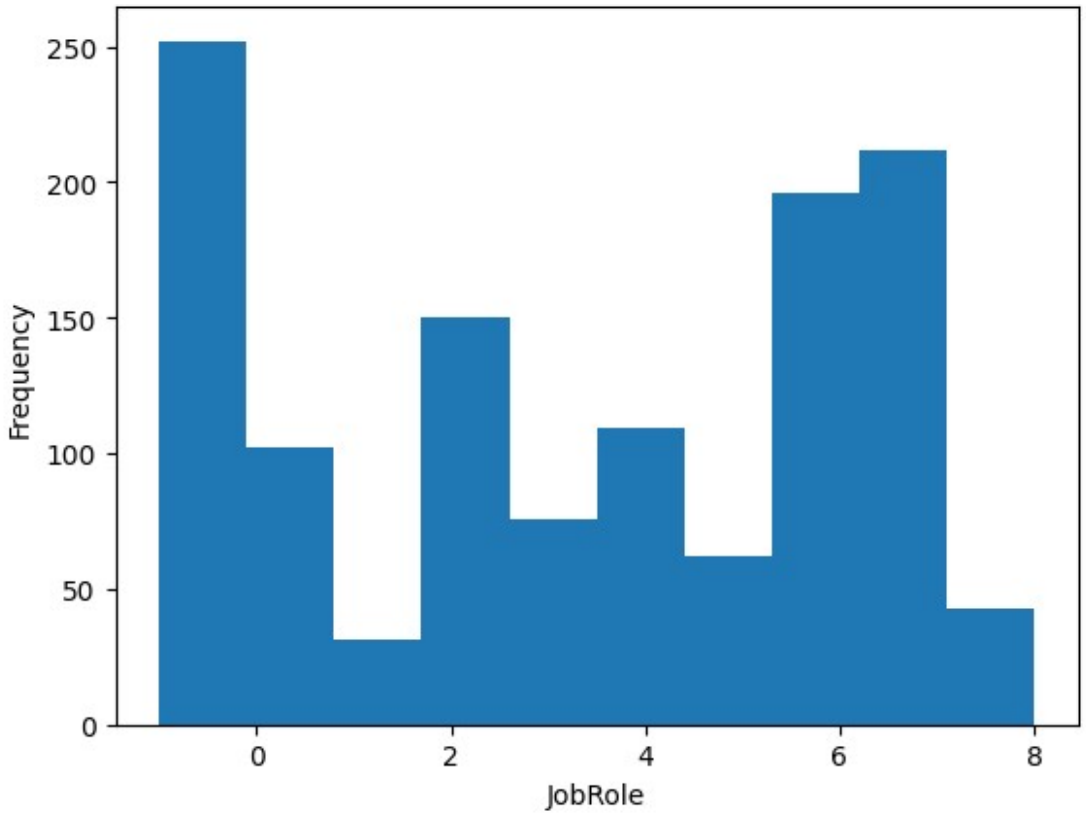
But if considering each of them,

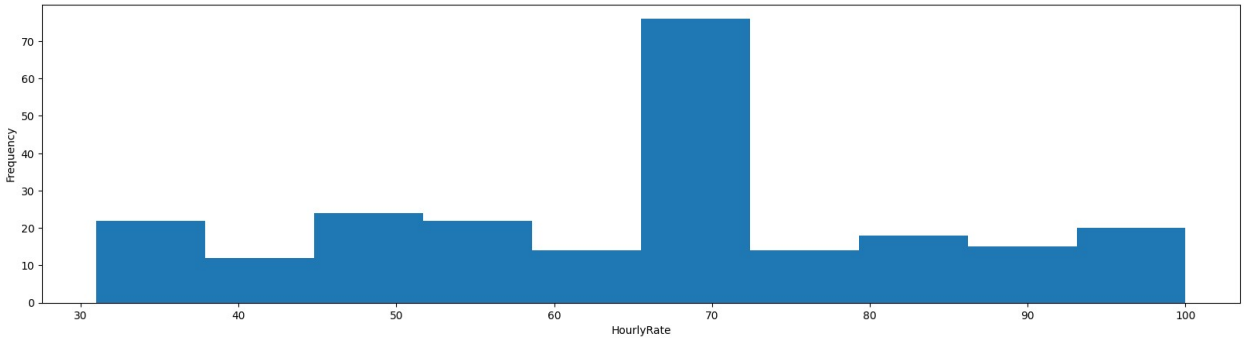
```
for col in ['MonthlyIncome', 'JobRole', 'HourlyRate',
'MaritalStatus']:
    plt.figure(figsize=(20, 5))
    leave_df = df.loc[df['Attrition'] == 1]
    stay_df = df.loc[df['Attrition'] == 0]

    display_histogram(leave_df, col, 10)

    display_histogram(stay_df, col, 10)
```

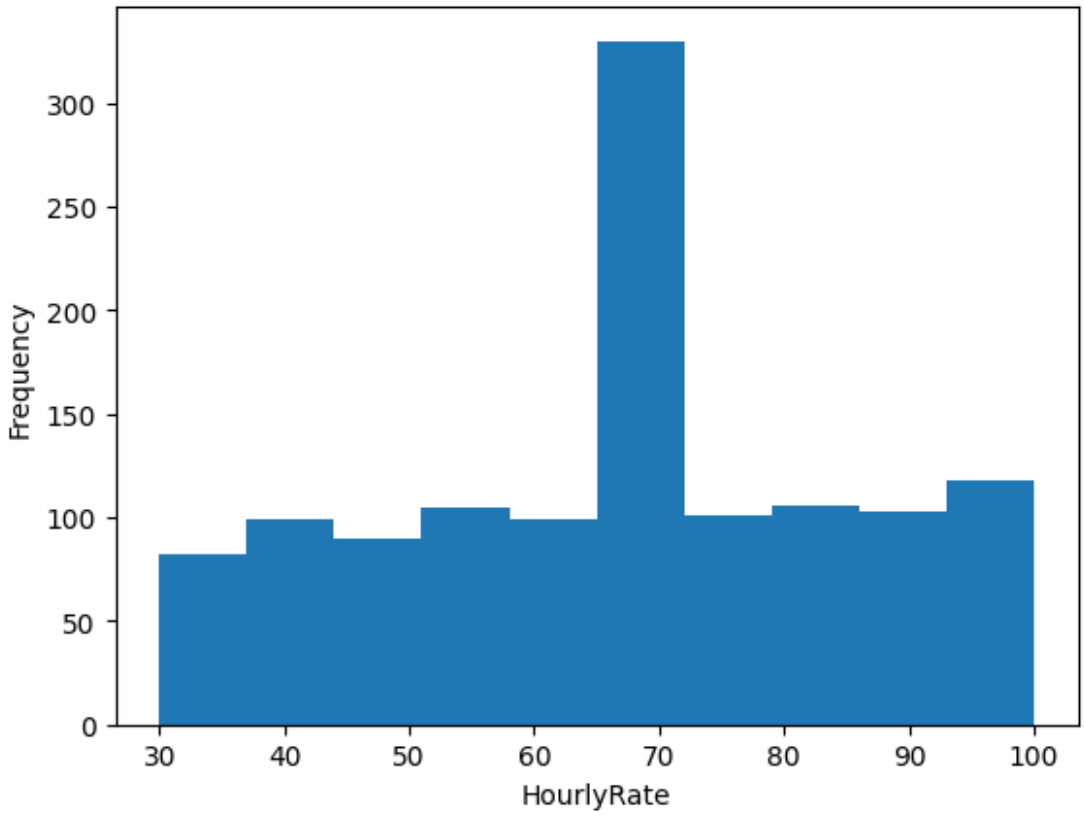


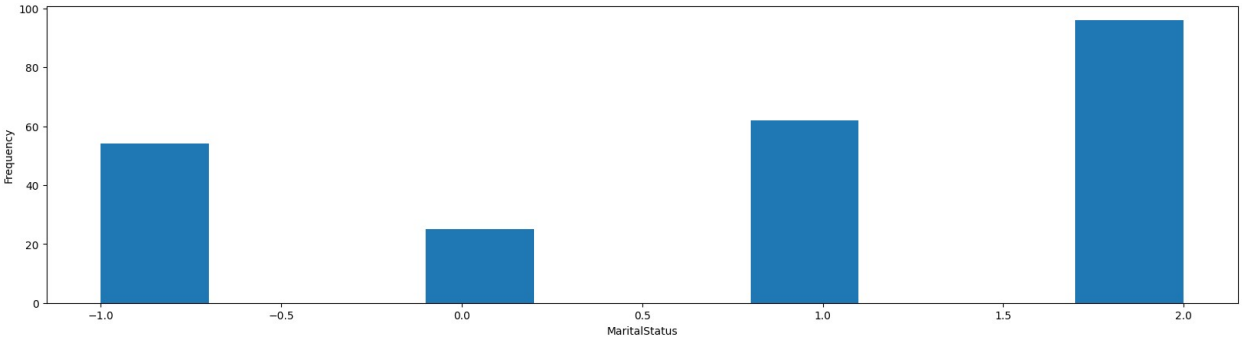

Frequency
400
300
200
100
0
2500
5000
7500
10000
12500
15000
17500
20000
MonthlyIncome

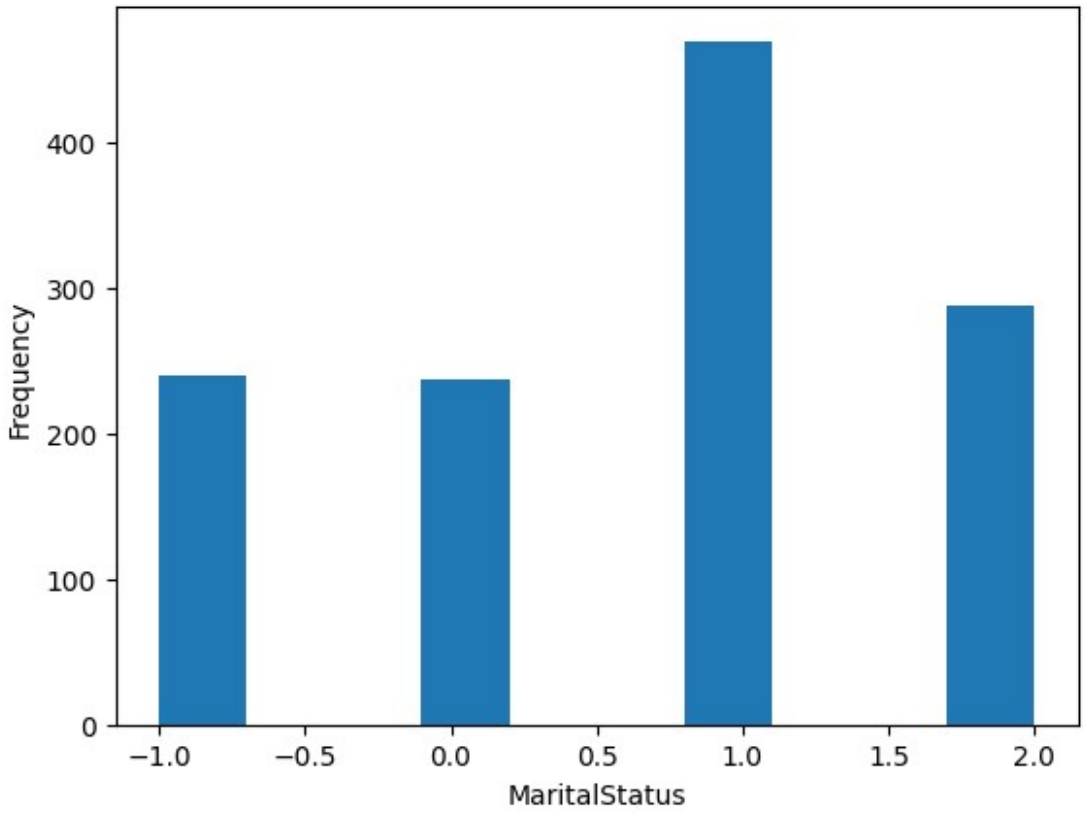

Frequency
50
40
30
20
10
0
0
2
4
6
8
JobRole


250
200
150
100
50
0
Frequency
0
2
4
6
8
JobRole


70
60
50
40
30
20
10
0
Frequency
30
40
50
60
70
80
90
100
HourlyRate


Frequency
HourlyRate
0
50
100
150
200
250
300
30
40
50
60
70
80
90
100


Frequency
MaritalStatus
0
20
40
60
80
100
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



From the distribution, I chose Exponential Distribution. p(x|\lambda) = \lambda * exp(-\lambda x)

The process to find MLE is

1. find the likelihood for each feature
2. find the derivative of sum of log of all likelihoods
3. set the derivative to zero and solve for the value that maximize the likelihoods

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

```
p_leave = df.loc[df['Attrition'] == 1, 'Attrition'].count() /
df.shape[0]
p_stay = df.loc[df['Attrition'] == 0, 'Attrition'].count()
/df.shape[0]

print("leave", p_leave)
print("stay", p_stay)
```

```
leave 0.16122448979591836
stay 0.8387755102040816
```

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.

Use flooring for handling zeros by changing them to small numbers (e.g. 1e-20)

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.

```
from SimpleBayesClassifier import SimpleBayesClassifier

data_train = df_train.to_numpy()
data_test = df_test.to_numpy()

test_size = 0.1

X = df_train.drop(columns=['Attrition']).to_numpy()
Y = df_train['Attrition'].to_numpy()

x_train = X[:-int(len(X) * test_size)]
y_train = Y[:-int(len(X) * test_size)]

x_test = X[-int(len(X) * test_size):]
y_test = Y[-int(len(X) * test_size):]

model = SimpleBayesClassifier(n_pos = (y_train==1).sum(), n_neg =
(y_train==0).sum())

model.prior_pos, model.prior_neg
```

```
(0.15449202350965574, 0.8455079764903443)
```

```
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) == (0.375, 0.625)
    assert (SimpleBayesClassifier(0, 1).prior_pos,
SimpleBayesClassifier(0, 1).prior_neg) == (0, 1)
    assert (SimpleBayesClassifier(1, 0).prior_pos,
SimpleBayesClassifier(1, 0).prior_neg) == (1, 0)

check_prior()

model.fit_params(x_train, y_train)
```

```
([(array([0.01787488, 0.04170804, 0.10923535, 0.14399206, 0.34558093,
          0.13108242, 0.0754717 , 0.05958292, 0.05064548,
0.02482622]),
   array([-inf, 22.2, 26.4, 30.6, 34.8, 39. , 43.2, 47.4, 51.6, 55.8,
inf])),
  (array([0.20754717, 0.        , 0.        , 0.08838133, 0.        ,
          0.        , 0.14399206, 0.        , 0.        ,
0.56007944]),
   array([-inf, -0.7, -0.4, -0.1,  0.2,  0.5,  0.8,  1.1,  1.4,  1.7,
inf])),
  (array([0.08142999, 0.06454816, 0.08043694, 0.09632572, 0.25819265,
          0.07646475, 0.07447865, 0.09235353, 0.09731877,
0.07845084]),
   array([  -inf,  241.7,  381.4,  521.1,  660.8,  800.5,  940.2,
1079.9,
          1219.6, 1359.3,    inf])),
  (array([0.19662363, 0.        , 0.        , 0.0367428 , 0.        ,
          0.        , 0.53723932, 0.        , 0.        ,
0.22939424]),
   array([-inf, -0.7, -0.4, -0.1,  0.2,  0.5,  0.8,  1.1,  1.4,  1.7,
inf])),
  (array([0.26613704, 0.10228401, 0.34756703, 0.0734856 , 0.01787488,
          0.03972195, 0.040715  , 0.03376365, 0.04170804,
0.0367428 ]),
   array([-inf,  3.8,  6.6,  9.4, 12.2, 15. , 17.8, 20.6, 23.4, 26.2,
inf])),
  (array([0.09036743, 0.        , 0.15392254, 0.        , 0.1857001 ,
          0.30883813, 0.        , 0.22939424, 0.        ,
0.03177756]),
   array([-inf,  1.4,  1.8,  2.2,  2.6,  3. ,  3.4,  3.8,  4.2,  4.6,
inf])),
  (array([0.19066534, 0.01489573, 0.        , 0.33565045, 0.        ,
          0.08043694, 0.26017875, 0.        , 0.04667329,
0.0714995 ]),
   array([-inf, -0.4,  0.2,  0.8,  1.4,  2. ,  2.6,  3.2,  3.8,  4.4,
inf])),
  (array([0.13704071, 0.        , 0.        , 0.14597815, 0.        ,
          0.19563059, 0.26713009, 0.        , 0.        ,
0.25422046]),
```

```
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.20456802, 0.        , 0.        , 0.        , 0.        ,
         0.33763654, 0.        , 0.        , 0.        ,
0.45779543]),
   array([-inf, -0.8, -0.6, -0.4, -0.2,  0. ,  0.2,  0.4,  0.6,  0.8,
inf])),
  (array([0.06852036, 0.08639523, 0.07050645, 0.08440914, 0.08043694,
         0.27706058, 0.08440914, 0.08440914, 0.0754717 ,
0.08838133]),
   array([-inf,  37.,  44.,  51.,  58.,  65.,  72.,  79.,  86.,  93.,
inf])),
  (array([0.02879841, 0.        , 0.        , 0.20655412, 0.        ,
         0.19364449, 0.48957299, 0.        , 0.        ,
0.08142999]),
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.2591857 , 0.        , 0.50248262, 0.        , 0.        ,
         0.12611718, 0.        , 0.07646475, 0.        ,
0.03574975]),
   array([-inf,  1.4,  1.8,  2.2,  2.6,  3. ,  3.4,  3.8,  4.2,  4.6,
inf])),
  (array([0.20854022, 0.08540218, 0.02482622, 0.11817279, 0.05858987,
         0.08639523, 0.05163853, 0.15690169, 0.17378352,
0.03574975]),
   array([-inf, -0.1,  0.8,  1.7,  2.6,  3.5,  4.4,  5.3,  6.2,  7.1,
inf])),
  (array([0.1489573 , 0.        , 0.        , 0.14299901, 0.        ,
         0.20754717, 0.23435948, 0.        , 0.        ,
0.26613704]),
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.18669315, 0.        , 0.        , 0.19563059, 0.        ,
         0.        , 0.37934459, 0.        , 0.        ,
0.23833168]),
   array([-inf, -0.7, -0.4, -0.1,  0.2,  0.5,  0.8,  1.1,  1.4,  1.7,
inf])),
  (array([0.17179742, 0.19265144, 0.36544191, 0.06454816, 0.04568024,
         0.04270109, 0.03078451, 0.00993049, 0.0387289 ,
0.03773585]),
   array([   -inf,  2931.5,  4811. ,  6690.5,  8570. , 10449.5, 12329.
,
         14208.5, 16088. , 17967.5,     inf])),
  (array([0.08341609, 0.09334657, 0.08142999, 0.07646475, 0.27805362,
         0.07944389, 0.07447865, 0.08043694, 0.08540218,
0.06752731]),
   array([  -inf,  4587.,  7077.,  9567., 12057., 14547., 17037.,
19527.,
         22017., 24507.,    inf])),
```

```
  (array([0.10724926, 0.27110228, 0.0774578 , 0.2999007 , 0.08440914,
          0.03078451, 0.0347567 , 0.040715  , 0.02780536,
0.02581927]),
   array([-inf,  0.9,  1.8,  2.7,  3.6,  4.5,  5.4,  6.3,  7.2,  8.1,
inf])),
  (array([0.19463754, 0.        , 0.        , 0.        , 0.        ,
          0.61171797, 0.        , 0.        , 0.        ,
0.19364449]),
   array([-inf, -0.8, -0.6, -0.4, -0.2,  0. ,  0.2,  0.4,  0.6,  0.8,
inf])),
  (array([0.21350546, 0.12015889, 0.16285998, 0.24528302, 0.04468719,
          0.08738828, 0.03277061, 0.05660377, 0.01886792,
0.01787488]),
   array([-inf, 12.4, 13.8, 15.2, 16.6, 18. , 19.4, 20.8, 22.2, 23.6,
inf])),
  (array([0.67428004, 0.20655412, 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        ,
0.11916584]),
   array([-inf,  3.1,  3.2,  3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9,
inf])),
  (array([0.1489573 , 0.        , 0.        , 0.17080437, 0.        ,
          0.19563059, 0.24428997, 0.        , 0.        ,
0.24031778]),
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.32571996, 0.        , 0.20258193, 0.35849057, 0.        ,
          0.        , 0.07845084, 0.        , 0.        ,
0.0347567 ]),
   array([-inf,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8,  2.1,  2.4,  2.7,
inf])),
  (array([0.06454816, 0.19066534, 0.25521351, 0.25223436, 0.07845084,
          0.06454816, 0.03078451, 0.02879841, 0.02284012,
0.01191658]),
   array([-inf,  3.7,  7.4, 11.1, 14.8, 18.5, 22.2, 25.9, 29.6, 33.3,
inf])),
  (array([0.02979146, 0.04667329, 0.        , 0.2979146 , 0.20158888,
          0.27110228, 0.05958292, 0.        , 0.06355511,
0.02979146]),
   array([-inf,  0.6,  1.2,  1.8,  2.4,  3. ,  3.6,  4.2,  4.8,  5.4,
inf])),
  (array([0.03177756, 0.        , 0.        , 0.1857001 , 0.        ,
          0.19165839, 0.50645482, 0.        , 0.        ,
0.08440914]),
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.21747766, 0.47269116, 0.183714  , 0.03078451, 0.03177756,
          0.04369414, 0.00595829, 0.00496524, 0.00496524,
0.00397219]),
   array([-inf,  3.7,  7.4, 11.1, 14.8, 18.5, 22.2, 25.9, 29.6, 33.3,
```

```
inf])),
  (array([0.14200596, 0.27308838, 0.27010924, 0.15590864, 0.05759682,
          0.05858987, 0.01787488, 0.01390268, 0.00794439,
0.00297915]),
   array([-inf,  1.8,  3.6,  5.4,  7.2,  9. , 10.8, 12.6, 14.4, 16.2,
inf])),
  (array([0.50347567, 0.29692155, 0.06156902, 0.02284012, 0.06057597,
          0.00993049, 0.00794439, 0.01390268, 0.01092354,
0.01191658]),
   array([-inf,  1.5,  3. ,  4.5,  6. ,  7.5,  9. , 10.5, 12. , 13.5,
inf])),
  (array([0.15789474, 0.28103277, 0.26017875, 0.01390268, 0.18967229,
          0.05263158, 0.01489573, 0.01886792, 0.00595829,
0.00496524]),
   array([-inf,  1.7,  3.4,  5.1,  6.8,  8.5, 10.2, 11.9, 13.6, 15.3,
inf]))],
 [(array([0.08695652, 0.08152174, 0.13586957, 0.18478261, 0.28804348,
          0.05978261, 0.03804348, 0.02717391, 0.04891304,
0.04891304]),
   array([-inf,  22.,  26.,  30.,  34.,  38.,  42.,  46.,  50.,  54.,
inf])),
  (array([0.19565217, 0.        , 0.        , 0.02717391, 0.        ,
          0.        , 0.22826087, 0.        , 0.        ,
0.54891304]),
   array([-inf, -0.7, -0.4, -0.1,  0.2,  0.5,  0.8,  1.1,  1.4,  1.7,
inf])),
  (array([0.08152174, 0.13043478, 0.08152174, 0.08152174, 0.06521739,
          0.30978261, 0.04891304, 0.08152174, 0.06521739,
0.05434783]),
   array([  -inf,  242.1,  380.2,  518.3,  656.4,  794.5,  932.6,
1070.7,
          1208.8, 1346.9,    inf])),
  (array([0.23369565, 0.        , 0.        , 0.04347826, 0.        ,
          0.        , 0.44021739, 0.        , 0.        ,
0.2826087 ]),
   array([-inf, -0.7, -0.4, -0.1,  0.2,  0.5,  0.8,  1.1,  1.4,  1.7,
inf])),
  (array([0.17934783, 0.08695652, 0.3423913 , 0.0923913 , 0.02717391,
          0.07065217, 0.0326087 , 0.04891304, 0.08152174,
0.03804348]),
   array([-inf,  3.8,  6.6,  9.4, 12.2, 15. , 17.8, 20.6, 23.4, 26.2,
inf])),
  (array([0.10326087, 0.        , 0.14130435, 0.        , 0.19021739,
          0.35326087, 0.        , 0.19021739, 0.        ,
0.02173913]),
   array([-inf,  1.4,  1.8,  2.2,  2.6,  3. ,  3.4,  3.8,  4.2,  4.6,
inf])),
  (array([0.19565217, 0.02717391, 0.        , 0.31521739, 0.        ,
          0.125     , 0.21195652, 0.        , 0.04891304,
```

```
0.07608696]),
   array([-inf, -0.4,  0.2,  0.8,  1.4,  2. ,  2.6,  3.2,  3.8,  4.4,
inf])),
  (array([0.21195652, 0.        , 0.        , 0.11956522, 0.        ,
         0.22826087, 0.20108696, 0.        , 0.        ,
0.23913043]),
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.19021739, 0.        , 0.        , 0.        , 0.        ,
         0.29891304, 0.        , 0.        , 0.        ,
0.51086957]),
   array([-inf, -0.8, -0.6, -0.4, -0.2,  0. ,  0.2,  0.4,  0.6,  0.8,
inf])),
  (array([0.09782609, 0.06521739, 0.0923913 , 0.09782609, 0.06521739,
         0.28804348, 0.05978261, 0.08695652, 0.05434783,
0.0923913 ]),
   array([-inf, 37.9, 44.8, 51.7, 58.6, 65.5, 72.4, 79.3, 86.2, 93.1,
inf])),
  (array([0.08152174, 0.        , 0.        , 0.23369565, 0.        ,
         0.19021739, 0.45108696, 0.        , 0.        ,
0.04347826]),
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.51086957, 0.        , 0.35326087, 0.        , 0.        ,
         0.10326087, 0.        , 0.01086957, 0.        ,
0.02173913]),
   array([-inf,  1.4,  1.8,  2.2,  2.6,  3. ,  3.4,  3.8,  4.2,  4.6,
inf])),
  (array([0.17934783, 0.02717391, 0.04347826, 0.25      , 0.01630435,
         0.0326087 , 0.00543478, 0.1576087 , 0.17934783,
0.10869565]),
   array([-inf, -0.1,  0.8,  1.7,  2.6,  3.5,  4.4,  5.3,  6.2,  7.1,
inf])),
  (array([0.16847826, 0.        , 0.        , 0.16304348, 0.        ,
         0.20652174, 0.27173913, 0.        , 0.        ,
0.19021739]),
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.21195652, 0.        , 0.        , 0.0923913 , 0.        ,
         0.        , 0.2826087 , 0.        , 0.        ,
0.41304348]),
   array([-inf, -0.7, -0.4, -0.1,  0.2,  0.5,  0.8,  1.1,  1.4,  1.7,
inf])),
  (array([0.38586957, 0.17391304, 0.26086957, 0.05978261, 0.06521739,
         0.01630435, 0.02173913, 0.        , 0.        ,
0.01630435]),
   array([  -inf,  2894.,  4779.,  6664.,  8549., 10434., 12319.,
14204.,
         16089., 17974.,    inf])),
```

```
  (array([0.08695652, 0.05434783, 0.07065217, 0.11413043, 0.30978261,
          0.05978261, 0.07065217, 0.07065217, 0.08152174,
0.08152174]),
   array([   -inf,  4884.3,  7321.6,  9758.9, 12196.2, 14633.5,
17070.8,
          19508.1, 21945.4, 24382.7,     inf])),
  (array([0.07065217, 0.35869565, 0.06521739, 0.21195652, 0.06521739,
          0.05978261, 0.03804348, 0.06521739, 0.02173913,
0.04347826]),
   array([-inf,  0.9,  1.8,  2.7,  3.6,  4.5,  5.4,  6.3,  7.2,  8.1,
inf])),
  (array([0.23369565, 0.        , 0.        , 0.        , 0.        ,
          0.35326087, 0.        , 0.        , 0.        ,
0.41304348]),
   array([-inf, -0.8, -0.6, -0.4, -0.2,  0. ,  0.2,  0.4,  0.6,  0.8,
inf])),
  (array([0.26630435, 0.15217391, 0.13043478, 0.17391304, 0.05434783,
          0.08695652, 0.0326087 , 0.04347826, 0.0326087 ,
0.02717391]),
   array([-inf, 12.4, 13.8, 15.2, 16.6, 18. , 19.4, 20.8, 22.2, 23.6,
inf])),
  (array([0.67934783, 0.20108696, 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        ,
0.11956522]),
   array([-inf,  3.1,  3.2,  3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9,
inf])),
  (array([0.21195652, 0.        , 0.        , 0.15217391, 0.        ,
          0.2173913 , 0.22282609, 0.        , 0.        ,
0.19565217]),
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.54891304, 0.        , 0.17391304, 0.18478261, 0.        ,
          0.        , 0.04347826, 0.        , 0.        ,
0.04891304]),
   array([-inf,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8,  2.1,  2.4,  2.7,
inf])),
  (array([0.2173913 , 0.21195652, 0.40217391, 0.06521739, 0.04347826,
          0.02173913, 0.01630435, 0.01086957, 0.        ,
0.01086957]),
   array([-inf,   4.,   8.,  12.,  16.,  20.,  24.,  28.,  32.,  36.,
inf])),
  (array([0.03804348, 0.02717391, 0.        , 0.35869565, 0.19565217,
          0.24456522, 0.08152174, 0.        , 0.04347826,
0.01086957]),
   array([-inf,  0.6,  1.2,  1.8,  2.4,  3. ,  3.6,  4.2,  4.8,  5.4,
inf])),
  (array([0.08695652, 0.        , 0.        , 0.19565217, 0.        ,
          0.20652174, 0.4076087 , 0.        , 0.        ,
0.10326087]),
```

```
   array([-inf,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,
inf])),
  (array([0.41847826, 0.15217391, 0.29891304, 0.08152174, 0.01630435,
          0.01086957, 0.00543478, 0.01086957, 0.        ,
0.00543478]),
   array([-inf,  3.1,  6.2,  9.3, 12.4, 15.5, 18.6, 21.7, 24.8, 27.9,
inf])),
  (array([0.27173913, 0.2173913 , 0.30978261, 0.00543478, 0.09782609,
          0.03804348, 0.03804348, 0.        , 0.01086957,
0.01086957]),
   array([-inf,  1.5,  3. ,  4.5,  6. ,  7.5,  9. , 10.5, 12. , 13.5,
inf])),
  (array([0.53804348, 0.29347826, 0.04347826, 0.01086957, 0.07608696,
          0.        , 0.01630435, 0.        , 0.01086957,
0.01086957]),
   array([-inf,  1.5,  3. ,  4.5,  6. ,  7.5,  9. , 10.5, 12. , 13.5,
inf])),
  (array([0.36956522, 0.1576087 , 0.25543478, 0.        , 0.01630435,
          0.15217391, 0.01630435, 0.02173913, 0.        ,
0.01086957]),
   array([-inf,  1.4,  2.8,  4.2,  5.6,  7. ,  8.4,  9.8, 11.2, 12.6,
inf]))])

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()
```

```
STAY PARAMETERS
Feature : 0
BINS : [0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.5]
EDGES : [-inf  0.2  0.4  0.6  0.8  1.   1.2  1.4  1.6  1.8  inf]
Feature : 1
BINS : [0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.5]
EDGES : [-inf  1.2  1.4  1.6  1.8  2.   2.2  2.4  2.6  2.8  inf]
Feature : 2
BINS : [0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.5]
EDGES : [-inf  2.2  2.4  2.6  2.8  3.   3.2  3.4  3.6  3.8  inf]
Feature : 3
BINS : [0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.5]
EDGES : [-inf  3.2  3.4  3.6  3.8  4.   4.2  4.4  4.6  4.8  inf]

LEAVE PARAMETERS
Feature : 0
BINS : [0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.5]
EDGES : [-inf  1.2  1.4  1.6  1.8  2.   2.2  2.4  2.6  2.8  inf]
Feature : 1
BINS : [0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.5]
EDGES : [-inf  2.2  2.4  2.6  2.8  3.   3.2  3.4  3.6  3.8  inf]
Feature : 2
BINS : [0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.5]
EDGES : [-inf  3.2  3.4  3.6  3.8  4.   4.2  4.4  4.6  4.8  inf]
Feature : 3
BINS : [0.5 0.  0.  0.  0.  0.  0.  0.  0.  0.5]
EDGES : [-inf  4.2  4.4  4.6  4.8  5.   5.2  5.4  5.6  5.8  inf]
```

```
y_pred = model.predict(x = x_test)
```

```
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
```

```
def evaluate(y_true, y_pred, show_result = True):
    tp = np.sum((y_true == 1) & (y_pred == 1))
    tn = np.sum((y_true == 0) & (y_pred == 0))
    fp = np.sum((y_true == 0) & (y_pred == 1))
    fn = np.sum((y_true == 1) & (y_pred == 0))
```

```
    accuracy = (tp + tn) / (tp+tn+fp+fn + 1e-7)
    precision = tp / (tp + fp + 1e-7)
    recall = tp / (tp + fn + 1e-7)
    F1 = 1 / (1/(precision+1e-7) + 1/(recall+1e-7))
    fpr = fp / (fp + tn + 1e-7)

    if show_result:
        print(f"Accuracy: {accuracy:.2f}")
        print(f"Precision: {precision:.2f}")
        print(f"Recall: {recall:.2f}")
        print(f"F1: {F1:.2f}")
        print(f"False Positive Rate: {fpr:.2f}")

    return accuracy, precision, recall, F1, fpr

evaluate(y_test, y_pred)
```

```
Accuracy: 0.79
Precision: 0.52
Recall: 0.41
F1: 0.23
False Positive Rate: 0.11

(0.7878787872819101,
 0.5217391281663516,
 0.41379310202140307,
 0.2307692805473371,
 0.10679611640116883)
```

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)
```

```
([(37.83979828276892, 7.94041367449151),
  (1.0566037735849056, 1.214679162017356),
  (812.9765755358748, 364.3456607789383),
  (0.7994041708043694, 1.0061773662017477),
  (9.303872889771599, 7.2558841577956885),
  (2.9436064892689946, 0.9326724213765538),
  (1.6305858987090367, 1.7507490110299664),
  (2.7820925291665826, 0.9564183866432782),
  (0.253227408142999, 0.7734593302653434),
  (65.3890268123138, 18.00611095540603),
  (2.7647513325091704, 0.6017345006604421),
  (2.141106641266238, 0.99700059872111),
  (3.276067527308838, 3.102558963589619),
  (2.7731601578069163, 0.9831424211787009),
  (0.6693147964250248, 1.0350759161262142),
  (6764.293304014754, 4280.005956225302),
```

```
  (14175.166084854996, 6439.807312246716),
  (2.7120057556289643, 2.203407958141284),
  (-0.0009930486593843098, 0.623122010262537),
  (15.228117463469996, 3.254050337893988),
  (3.1502543420545974, 0.3182272910047396),
  (2.711878246830013, 0.9706535003387708),
  (0.772115598970472, 0.709697548175834),
  (11.806597018151848, 7.055232354171872),
  (2.752577197711259, 1.1324970815219448),
  (2.791150720466936, 0.6030789485456384),
  (7.349718636213174, 5.514333156236812),
  (4.455520539894209, 3.2311696489618877),
  (2.183268987833465, 2.8393798659601126),
  (4.293141884360496, 3.2148066920113707)],
 [(34.433895297249336, 9.225446100276185),
  (1.1304347826086956, 1.1583101569354104),
  (736.98526693286, 349.39982329489607),
  (0.7717391304347826, 1.0994391936995525),
  (11.02445652173913, 7.5253024082136495),
  (2.8708268633540373, 0.9033413758498753),
  (1.5815217391304348, 1.7612637105848352),
  (2.6348990683229814, 1.043739852881892),
  (0.32065217391304346, 0.7734785969156344),
  (64.83462732919254, 18.471961531955568),
  (2.595141045548654, 0.6733101186306114),
  (1.6919180715764568, 0.8713523733864098),
  (3.641304347826087, 3.191638265909199),
  (2.6384113427979887, 0.9536367866830902),
  (0.8967391304347826, 1.1587053388941697),
  (5006.64072204969, 3327.4639058443927),
  (14501.74525842946, 6196.222485138566),
  (2.8938923395445135, 2.4360852395576864),
  (0.1793478260869565, 0.7839473756016186),
  (15.130046583850932, 3.5500645228966867),
  (3.149830856255546, 0.3188592469839197),
  (2.553201715468796, 1.0081083240580564),
  (0.5493566992014197, 0.7717596535430221),
  (8.998303941141675, 6.619920108391039),
  (2.6323203194321207, 1.018183245498785),
  (2.686279946761313, 0.7478001855619241),
  (5.316760943507838, 4.588991664942314),
  (3.3605534605146405, 3.042330240658189),
  (1.9395842576160895, 2.7339093939672994),
  (3.037017524401065, 3.0765971735536466)])

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()
```

```
STAY PARAMETERS
Feature : 0
Mean : 1.0
STD. : 1.0
Feature : 1
Mean : 2.0
STD. : 1.0
Feature : 2
Mean : 3.0
STD. : 1.0
Feature : 3
Mean : 4.0
STD. : 1.0

LEAVE PARAMETERS
Feature : 0
Mean : 2.0
STD. : 1.0
Feature : 1
```

```
Mean : 3.0
STD. : 1.0
Feature : 2
Mean : 4.0
STD. : 1.0
Feature : 3
Mean : 5.0
STD. : 1.0

y_pred = model.gaussian_predict(x_test)

evaluate(y_test, y_pred)

Accuracy: 0.77
Precision: 0.47
Recall: 0.48
F1: 0.24
False Positive Rate: 0.16

(0.7651515145718549,
 0.4666666651111111,
 0.48275861902497025,
 0.23728818480321745,
 0.15533980567442737)
```

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.

```
import random

y_pred = np.array([random.randint(0, 1) for _ in range(len(y_test))])

evaluate(y_test, y_pred)

Accuracy: 0.48
Precision: 0.22
Recall: 0.55
F1: 0.16
False Positive Rate: 0.54

(0.477272726911157,
 0.22222222191358026,
 0.5517241360285374,
 0.15841590033329828,
 0.5436893198604958)
```

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.

```
unique, counts = np.unique(y_pred, return_counts=True)
max_freq = sorted([(counts[i], unique[i]) for i in
range(len(unique))])[-1][1]

y_pred = np.array([max_freq for _ in range(len(y_test))])

evaluate(y_test, y_pred)
```

```
Accuracy: 0.20
Precision: 0.20
Recall: 1.00
F1: 0.17
False Positive Rate: 1.00

(0.2040816325142302,
 0.2040816325142302,
 0.9999999966666666,
 0.16949159707937966,
 0.9999999991452992)
```

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

The naive bayes one got 0.77 Accuracy with medium precision and recall because it learned from the data.

The random one got 0.62 Accuracy with low precision from randoming.

While, the majoirty rule got 0.78 Accuracy but 0 precision because it always guess the same.

T16. Use the following threshold values

$ t = np.arange(-5,5,0.05) $

find the best accuracy, and F score (and the corresponding thresholds)

```
best_acc = float('-inf')
best_acc_t = 0

best_f1 = float('-inf')
best_f1_t = 0

hist = {
    'Accuracy': [],
    'Precision': [],
    'Recall': [],
```

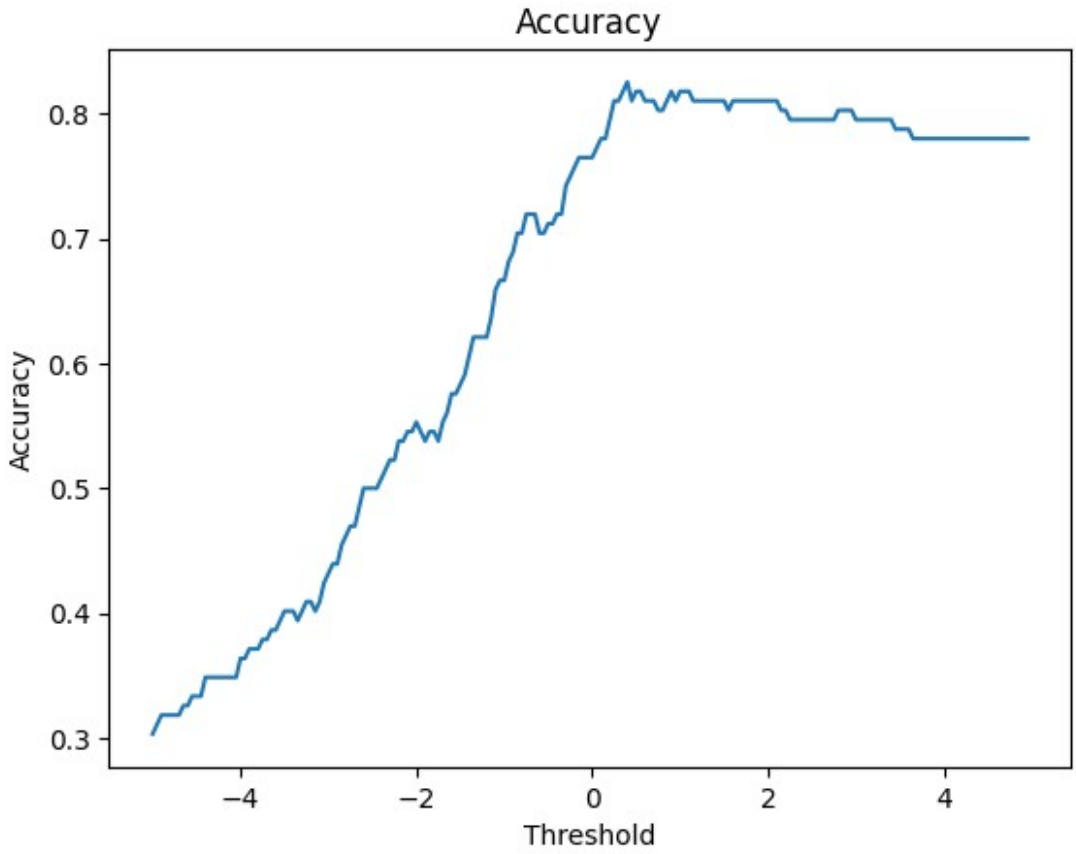
```
    'F1': [],
    'FPR': []
}

for threshold in np.arange(-5, 5, 0.05):
    y_pred = model.gaussian_predict(x_test, threshold)

    print(f"Threshold: {threshold:.2f}",end='\t| ')
    eva = evaluate(y_test, y_pred, show_result=False)
    for idx, metric in enumerate(['Accuracy', 'Precision', 'Recall',
'F1', 'FPR']):
        print(f"{metric}: {eva[idx]:.2f}", end='\t| ')
        hist[metric].append(eva[idx])
    print()
    if eva[0] > best_acc:
        best_acc = eva[0]
        best_acc_t = threshold
    if eva[3] > best_f1:
        best_f1 = eva[3]
        best_f1_t = threshold

print(f"Best threshold for accuracy: {best_acc_t:.2f} (Accuracy:
{best_acc:.2f})")
print(f"Best threshold for F1: {best_f1_t:.2f} (F1: {best_f1:.2f})")
```

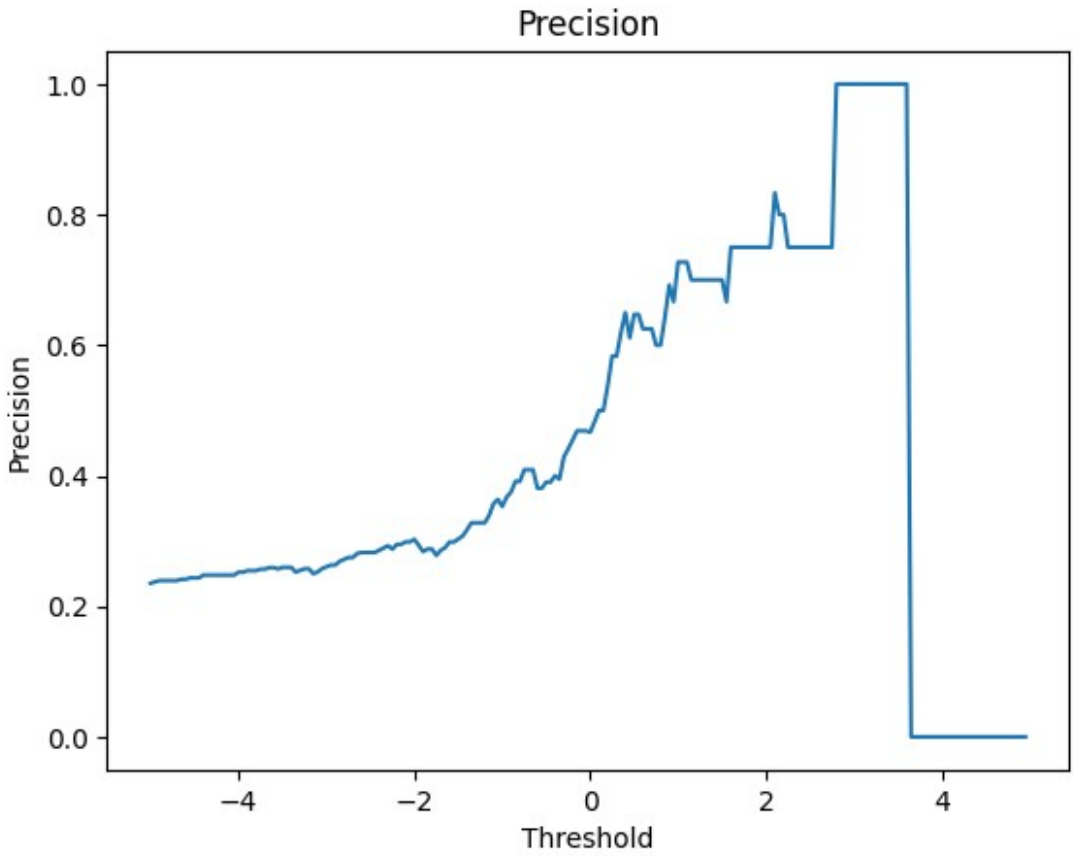
```
Threshold: -5.00|

---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[170], line 19
     16 y_pred = model.gaussian_predict(x_test, threshold)
     18 print(f"Threshold: {threshold:.2f}",end='\t| ')
---> 19 eva = evaluate(y_test, y_pred, show_result=False)
     20 for idx, metric in enumerate(['Accuracy', 'Precision',
'Recall', 'F1', 'FPR']):
     21     print(f"{metric}: {eva[idx]:.2f}", end='\t| ')

Cell In[152], line 2, in evaluate(y_true, y_pred, show_result)
      1 def evaluate(y_true, y_pred, show_result = True):
----> 2     tp = np.sum((y_true == 1) & (y_pred == 1))
      3     tn = np.sum((y_true == 0) & (y_pred == 0))
      4     fp = np.sum((y_true == 0) & (y_pred == 1))

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\pandas\core\ops\common.py:76, in
_unpack_zerodim_and_defer.<locals>.new_method(self, other)
     72             return NotImplemented
```

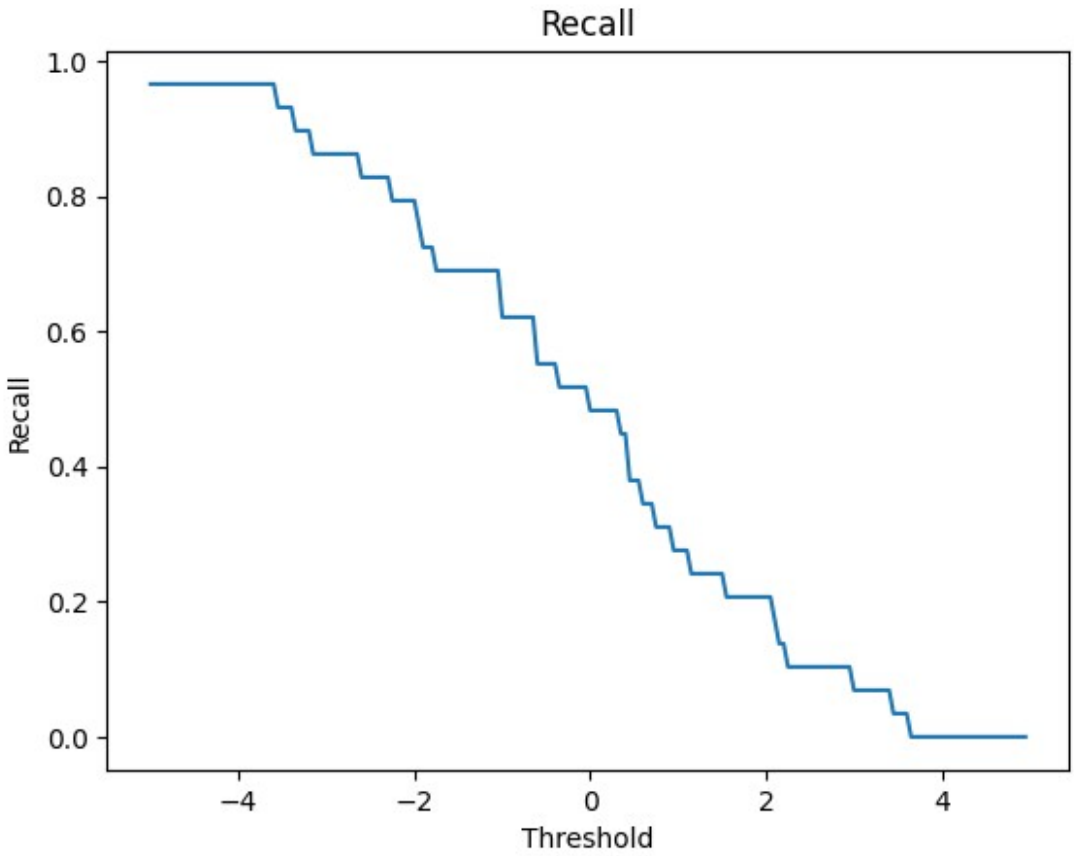
```
     74 other = item_from_zerodim(other)
---> 76 return method(self, other)

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\pandas\core\arraylike.py:70, in
OpsMixin.__and__(self, other)
     68 @unpack_zerodim_and_defer("__and__")
     69 def __and__(self, other):
---> 70     return self._logical_method(other, operator.and_)

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\pandas\core\series.py:6110, in
Series._logical_method(self, other, op)
   6107 lvalues = self._values
   6108 rvalues = extract_array(other, extract_numpy=True,
extract_range=True)
-> 6110 res_values = ops.logical_op(lvalues, rvalues, op)
   6111 return self._construct_result(res_values, name=res_name)

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\pandas\core\ops\array_ops.py:454, in
logical_op(left, right, op)
    450 else:
    451     # i.e. scalar
    452     is_other_int_dtype = lib.is_integer(rvalues)
--> 454 res_values = na_logical_op(lvalues, rvalues, op)
    456 # For int vs int `^`, `|`, `&` are bitwise operators and
return
    457 #   integer dtypes.  Otherwise these are boolean ops
    458 if not (left.dtype.kind in "iu" and is_other_int_dtype):

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\pandas\core\ops\array_ops.py:362, in
na_logical_op(x, y, op)
    352 def na_logical_op(x: np.ndarray, y, op):
    353     try:
    354         # For exposition, write:
    355         #  yarr = isinstance(y, np.ndarray)
   (...)
    360         # Then Cases where this goes through without raising
include:
    361         #  (xint or xbool) and (yint or bool)
--> 362         result = op(x, y)
    363     except TypeError:
    364         if isinstance(y, np.ndarray):
    365             # bool-bool dtype operations should be OK, should
```

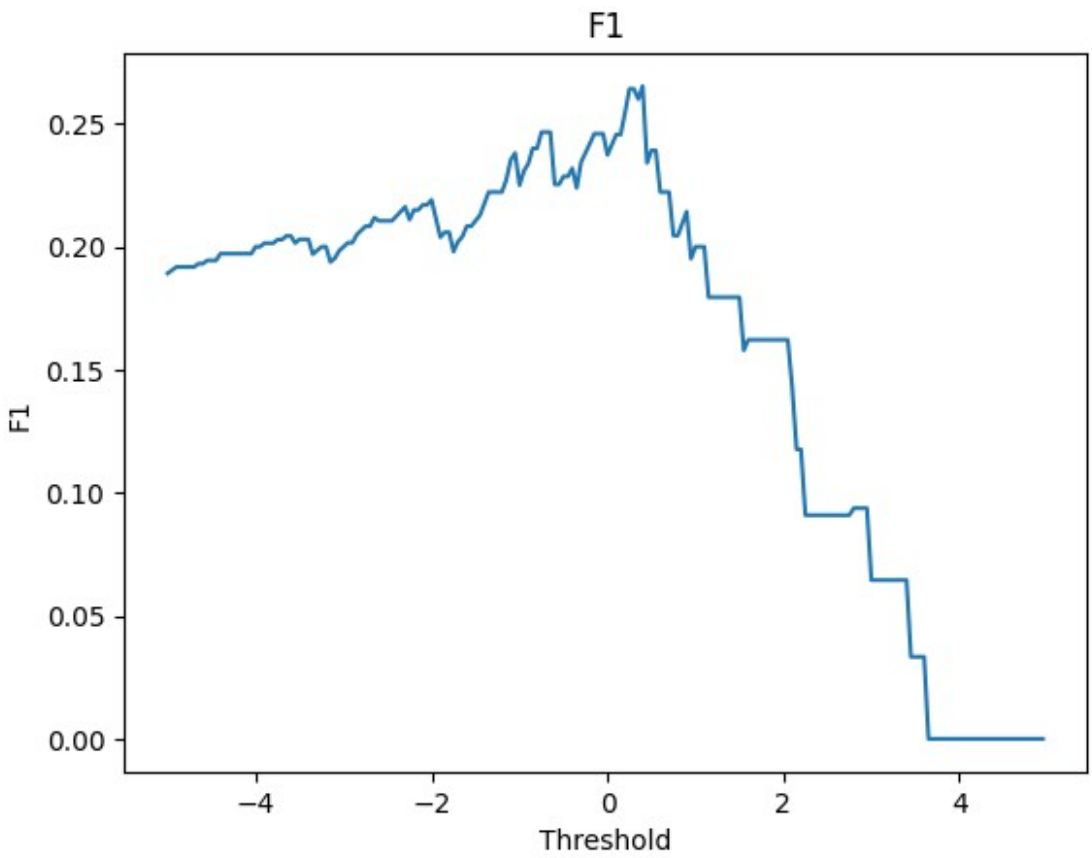
```
not get here

ValueError: operands could not be broadcast together with shapes
(147,) (132,)
```

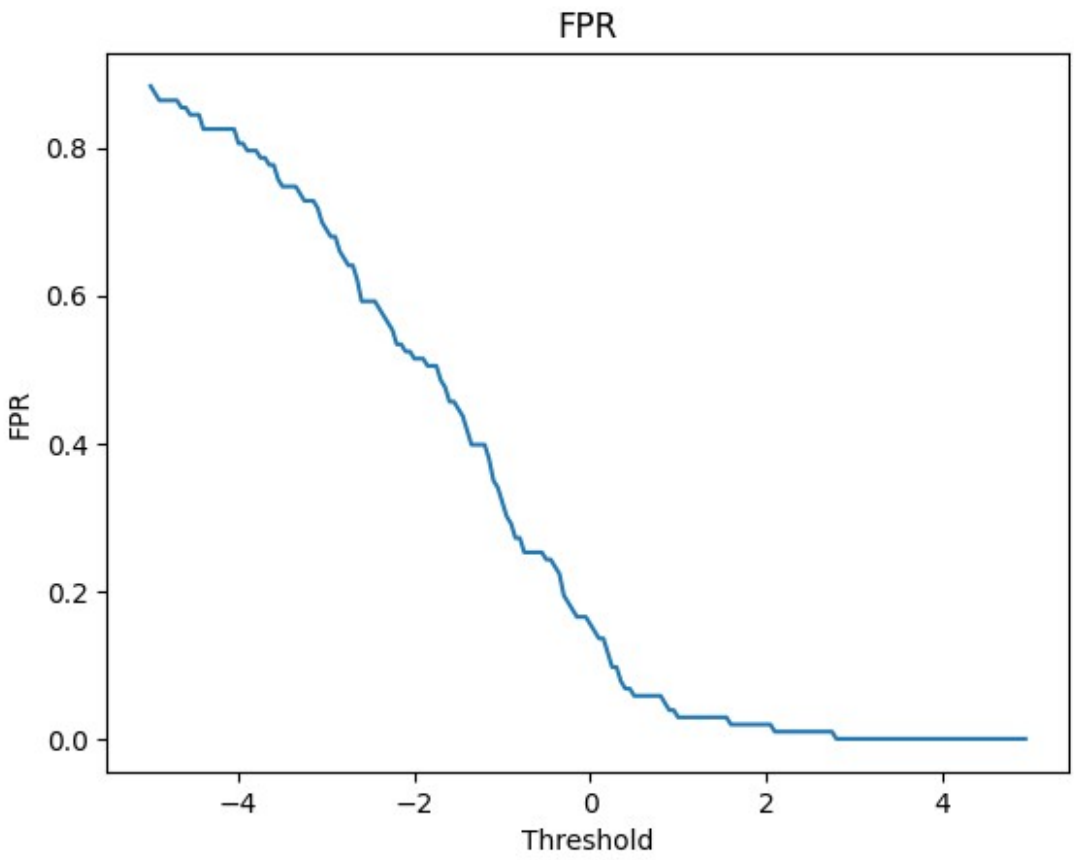
```
for idx, metric in enumerate(['Accuracy', 'Precision', 'Recall', 'F1',
'FPR']):
    plt.plot(np.arange(-5, 5, 0.05), hist[metric])
    plt.title(metric)
    plt.xlabel('Threshold')
    plt.ylabel(metric)
    plt.show()
```




Precision
1.0
0.8
0.6
0.4
0.2
0.0
Precision
−4
−2
0
2
4
Threshold


Recall
1.0
0.8
0.6
0.4
0.2
0.0
Recall
−4
−2
0
2
4
Threshold

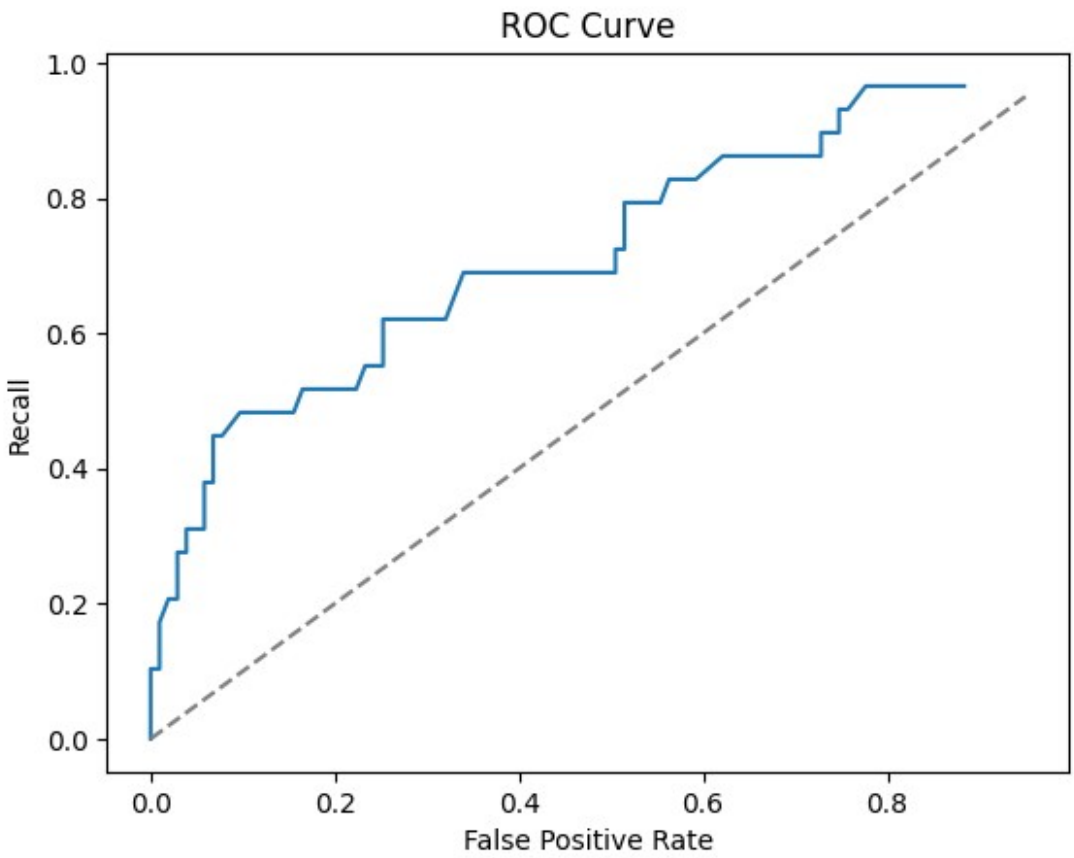

F1
0.25
0.20
0.15
0.10
0.05
0.00
F1
−4
−2
0
2
4
Threshold



T17. Plot the RoC of your classifier.

```
plt.plot(hist['FPR'], hist['Recall'])
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('Recall')

plt.plot(np.arange(0, 1, 0.05), np.arange(0, 1, 0.05), linestyle='--',
color='gray')
```

```
[<matplotlib.lines.Line2D at 0x1703c6bbcd0>]
```



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.

```
model.fit_params(x_train, y_train, 5)

best_acc_bin5 = float('-inf')
best_acc_t_bin5 = 0

best_f1_bin5 = float('-inf')
best_f1_t_bin5 = 0

hist_bin5 = {
    'Accuracy': [],
    'Precision': [],
    'Recall': [],
    'F1': [],
    'FPR': []
}

for threshold in np.arange(-5, 5, 0.05):
```

```
    y_pred = model.predict(x_test, threshold)

    print(f"Threshold: {threshold:.2f}",end='\t| ')
    eva = evaluate(y_test, y_pred, show_result=False)
    for idx, metric in enumerate(['Accuracy', 'Precision', 'Recall',
'F1', 'FPR']):
        print(f"{metric}: {eva[idx]:.2f}", end='\t| ')
        hist_bin5[metric].append(eva[idx])
    print()
    if eva[0] > best_acc_bin5:
        best_acc_bin5 = eva[0]
        best_acc_t_bin5 = threshold
    if eva[3] > best_f1_bin5:
        best_f1_bin5 = eva[3]
        best_f1_t_bin5 = threshold

print(f"Best threshold for accuracy: {best_acc_t_bin5:.2f} (Accuracy:
{best_acc_bin5:.2f})")
print(f"Best threshold for F1: {best_f1_t_bin5:.2f} (F1:
{best_f1_bin5:.2f})")
```

```
Threshold: -5.00| Accuracy: 0.42| Precision: 0.26     | Recall: 0.90
      | F1: 0.20 | FPR: 0.71      |
Threshold: -4.95| Accuracy: 0.42| Precision: 0.26     | Recall: 0.90
      | F1: 0.20 | FPR: 0.71      |
Threshold: -4.90| Accuracy: 0.44| Precision: 0.27     | Recall: 0.90
      | F1: 0.21 | FPR: 0.69      |
Threshold: -4.85| Accuracy: 0.45| Precision: 0.27     | Recall: 0.90
      | F1: 0.21 | FPR: 0.68      |
Threshold: -4.80| Accuracy: 0.45| Precision: 0.27     | Recall: 0.90
      | F1: 0.21 | FPR: 0.68      |
Threshold: -4.75| Accuracy: 0.45| Precision: 0.27     | Recall: 0.90
      | F1: 0.21 | FPR: 0.68      |
Threshold: -4.70| Accuracy: 0.46| Precision: 0.28     | Recall: 0.90
      | F1: 0.21 | FPR: 0.66      |
Threshold: -4.65| Accuracy: 0.46| Precision: 0.27     | Recall: 0.86
      | F1: 0.21 | FPR: 0.65      |
Threshold: -4.60| Accuracy: 0.46| Precision: 0.27     | Recall: 0.86
      | F1: 0.21 | FPR: 0.65      |
Threshold: -4.55| Accuracy: 0.48| Precision: 0.28     | Recall: 0.86
      | F1: 0.21 | FPR: 0.63      |
Threshold: -4.50| Accuracy: 0.47| Precision: 0.26     | Recall: 0.79
      | F1: 0.20 | FPR: 0.62      |
Threshold: -4.45| Accuracy: 0.47| Precision: 0.26     | Recall: 0.79
      | F1: 0.20 | FPR: 0.62      |
Threshold: -4.40| Accuracy: 0.47| Precision: 0.26     | Recall: 0.79
      | F1: 0.20 | FPR: 0.62      |
Threshold: -4.35| Accuracy: 0.50| Precision: 0.28     | Recall: 0.79
      | F1: 0.21 | FPR: 0.58      |
Threshold: -4.30| Accuracy: 0.49| Precision: 0.27     | Recall: 0.76
```

```
      | F1: 0.20 | FPR: 0.58      |
Threshold: -4.25| Accuracy: 0.49| Precision: 0.27     | Recall: 0.76
      | F1: 0.20 | FPR: 0.58      |
Threshold: -4.20| Accuracy: 0.49| Precision: 0.27     | Recall: 0.76
      | F1: 0.20 | FPR: 0.58      |
Threshold: -4.15| Accuracy: 0.51| Precision: 0.27     | Recall: 0.76
      | F1: 0.20 | FPR: 0.56      |
Threshold: -4.10| Accuracy: 0.51| Precision: 0.27     | Recall: 0.76
      | F1: 0.20 | FPR: 0.56      |
Threshold: -4.05| Accuracy: 0.53| Precision: 0.29     | Recall: 0.76
      | F1: 0.21 | FPR: 0.53      |
Threshold: -4.00| Accuracy: 0.53| Precision: 0.29     | Recall: 0.76
      | F1: 0.21 | FPR: 0.53      |
Threshold: -3.95| Accuracy: 0.53| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.52      |
Threshold: -3.90| Accuracy: 0.53| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.52      |
Threshold: -3.85| Accuracy: 0.53| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.52      |
Threshold: -3.80| Accuracy: 0.54| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.51      |
Threshold: -3.75| Accuracy: 0.54| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.51      |
Threshold: -3.70| Accuracy: 0.54| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.51      |
Threshold: -3.65| Accuracy: 0.54| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.51      |
Threshold: -3.60| Accuracy: 0.54| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.51      |
Threshold: -3.55| Accuracy: 0.54| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.51      |
Threshold: -3.50| Accuracy: 0.54| Precision: 0.28     | Recall: 0.72
      | F1: 0.20 | FPR: 0.51      |
Threshold: -3.45| Accuracy: 0.54| Precision: 0.28     | Recall: 0.69
      | F1: 0.20 | FPR: 0.50      |
Threshold: -3.40| Accuracy: 0.54| Precision: 0.28     | Recall: 0.69
      | F1: 0.20 | FPR: 0.50      |
Threshold: -3.35| Accuracy: 0.55| Precision: 0.28     | Recall: 0.69
      | F1: 0.20 | FPR: 0.50      |
Threshold: -3.30| Accuracy: 0.55| Precision: 0.28     | Recall: 0.69
      | F1: 0.20 | FPR: 0.50      |
Threshold: -3.25| Accuracy: 0.55| Precision: 0.29     | Recall: 0.69
      | F1: 0.20 | FPR: 0.49      |
Threshold: -3.20| Accuracy: 0.55| Precision: 0.29     | Recall: 0.69
      | F1: 0.20 | FPR: 0.49      |
Threshold: -3.15| Accuracy: 0.55| Precision: 0.29     | Recall: 0.69
      | F1: 0.20 | FPR: 0.49      |
Threshold: -3.10| Accuracy: 0.56| Precision: 0.29     | Recall: 0.69
      | F1: 0.20 | FPR: 0.48      |
```

```
Threshold: -3.05| Accuracy: 0.58| Precision: 0.30     | Recall: 0.69
      | F1: 0.21 | FPR: 0.46      |
Threshold: -3.00| Accuracy: 0.58| Precision: 0.30     | Recall: 0.69
      | F1: 0.21 | FPR: 0.46      |
Threshold: -2.95| Accuracy: 0.58| Precision: 0.30     | Recall: 0.69
      | F1: 0.21 | FPR: 0.46      |
Threshold: -2.90| Accuracy: 0.59| Precision: 0.31     | Recall: 0.69
      | F1: 0.21 | FPR: 0.44      |
Threshold: -2.85| Accuracy: 0.61| Precision: 0.32     | Recall: 0.69
      | F1: 0.22 | FPR: 0.42      |
Threshold: -2.80| Accuracy: 0.61| Precision: 0.32     | Recall: 0.69
      | F1: 0.22 | FPR: 0.41      |
Threshold: -2.75| Accuracy: 0.63| Precision: 0.33     | Recall: 0.69
      | F1: 0.22 | FPR: 0.39      |
Threshold: -2.70| Accuracy: 0.64| Precision: 0.34     | Recall: 0.69
      | F1: 0.23 | FPR: 0.37      |
Threshold: -2.65| Accuracy: 0.64| Precision: 0.34     | Recall: 0.69
      | F1: 0.23 | FPR: 0.37      |
Threshold: -2.60| Accuracy: 0.65| Precision: 0.35     | Recall: 0.69
      | F1: 0.23 | FPR: 0.36      |
Threshold: -2.55| Accuracy: 0.65| Precision: 0.35     | Recall: 0.66
      | F1: 0.23 | FPR: 0.35      |
Threshold: -2.50| Accuracy: 0.66| Precision: 0.35     | Recall: 0.66
      | F1: 0.23 | FPR: 0.34      |
Threshold: -2.45| Accuracy: 0.66| Precision: 0.35     | Recall: 0.66
      | F1: 0.23 | FPR: 0.34      |
Threshold: -2.40| Accuracy: 0.67| Precision: 0.36     | Recall: 0.66
      | F1: 0.23 | FPR: 0.33      |
Threshold: -2.35| Accuracy: 0.67| Precision: 0.36     | Recall: 0.66
      | F1: 0.23 | FPR: 0.33      |
Threshold: -2.30| Accuracy: 0.67| Precision: 0.36     | Recall: 0.66
      | F1: 0.23 | FPR: 0.33      |
Threshold: -2.25| Accuracy: 0.67| Precision: 0.37     | Recall: 0.66
      | F1: 0.23 | FPR: 0.32      |
Threshold: -2.20| Accuracy: 0.69| Precision: 0.38     | Recall: 0.66
      | F1: 0.24 | FPR: 0.30      |
Threshold: -2.15| Accuracy: 0.70| Precision: 0.39     | Recall: 0.66
      | F1: 0.24 | FPR: 0.29      |
Threshold: -2.10| Accuracy: 0.70| Precision: 0.39     | Recall: 0.66
      | F1: 0.24 | FPR: 0.29      |
Threshold: -2.05| Accuracy: 0.70| Precision: 0.40     | Recall: 0.66
      | F1: 0.25 | FPR: 0.28      |
Threshold: -2.00| Accuracy: 0.70| Precision: 0.40     | Recall: 0.66
      | F1: 0.25 | FPR: 0.28      |
Threshold: -1.95| Accuracy: 0.68| Precision: 0.36     | Recall: 0.55
      | F1: 0.22 | FPR: 0.28      |
Threshold: -1.90| Accuracy: 0.69| Precision: 0.36     | Recall: 0.55
      | F1: 0.22 | FPR: 0.27      |
Threshold: -1.85| Accuracy: 0.69| Precision: 0.36     | Recall: 0.55
```

```
      | F1: 0.22 | FPR: 0.27      |
Threshold: -1.80| Accuracy: 0.70| Precision: 0.38     | Recall: 0.55
      | F1: 0.23 | FPR: 0.25      |
Threshold: -1.75| Accuracy: 0.70| Precision: 0.37     | Recall: 0.52
      | F1: 0.22 | FPR: 0.24      |
Threshold: -1.70| Accuracy: 0.71| Precision: 0.38     | Recall: 0.52
      | F1: 0.22 | FPR: 0.23      |
Threshold: -1.65| Accuracy: 0.70| Precision: 0.37     | Recall: 0.48
      | F1: 0.21 | FPR: 0.23      |
Threshold: -1.60| Accuracy: 0.71| Precision: 0.38     | Recall: 0.48
      | F1: 0.21 | FPR: 0.22      |
Threshold: -1.55| Accuracy: 0.71| Precision: 0.38     | Recall: 0.48
      | F1: 0.21 | FPR: 0.22      |
Threshold: -1.50| Accuracy: 0.73| Precision: 0.40     | Recall: 0.48
      | F1: 0.22 | FPR: 0.20      |
Threshold: -1.45| Accuracy: 0.73| Precision: 0.40     | Recall: 0.48
      | F1: 0.22 | FPR: 0.20      |
Threshold: -1.40| Accuracy: 0.73| Precision: 0.39     | Recall: 0.45
      | F1: 0.21 | FPR: 0.19      |
Threshold: -1.35| Accuracy: 0.74| Precision: 0.42     | Recall: 0.45
      | F1: 0.22 | FPR: 0.17      |
Threshold: -1.30| Accuracy: 0.75| Precision: 0.43     | Recall: 0.45
      | F1: 0.22 | FPR: 0.17      |
Threshold: -1.25| Accuracy: 0.75| Precision: 0.43     | Recall: 0.45
      | F1: 0.22 | FPR: 0.17      |
Threshold: -1.20| Accuracy: 0.75| Precision: 0.43     | Recall: 0.45
      | F1: 0.22 | FPR: 0.17      |
Threshold: -1.15| Accuracy: 0.76| Precision: 0.45     | Recall: 0.45
      | F1: 0.22 | FPR: 0.16      |
Threshold: -1.10| Accuracy: 0.75| Precision: 0.43     | Recall: 0.41
      | F1: 0.21 | FPR: 0.16      |
Threshold: -1.05| Accuracy: 0.77| Precision: 0.46     | Recall: 0.41
      | F1: 0.22 | FPR: 0.14      |
Threshold: -1.00| Accuracy: 0.77| Precision: 0.46     | Recall: 0.41
      | F1: 0.22 | FPR: 0.14      |
Threshold: -0.95| Accuracy: 0.77| Precision: 0.46     | Recall: 0.41
      | F1: 0.22 | FPR: 0.14      |
Threshold: -0.90| Accuracy: 0.77| Precision: 0.46     | Recall: 0.41
      | F1: 0.22 | FPR: 0.14      |
Threshold: -0.85| Accuracy: 0.77| Precision: 0.48     | Recall: 0.41
      | F1: 0.22 | FPR: 0.13      |
Threshold: -0.80| Accuracy: 0.78| Precision: 0.50     | Recall: 0.41
      | F1: 0.23 | FPR: 0.12      |
Threshold: -0.75| Accuracy: 0.78| Precision: 0.50     | Recall: 0.41
      | F1: 0.23 | FPR: 0.12      |
Threshold: -0.70| Accuracy: 0.80| Precision: 0.55     | Recall: 0.41
      | F1: 0.24 | FPR: 0.10      |
Threshold: -0.65| Accuracy: 0.81| Precision: 0.60     | Recall: 0.41
      | F1: 0.24 | FPR: 0.08      |
```

```
Threshold: -0.60| Accuracy: 0.81| Precision: 0.60     | Recall: 0.41
      | F1: 0.24 | FPR: 0.08      |
Threshold: -0.55| Accuracy: 0.82| Precision: 0.63     | Recall: 0.41
      | F1: 0.25 | FPR: 0.07      |
Threshold: -0.50| Accuracy: 0.83| Precision: 0.67     | Recall: 0.41
      | F1: 0.26 | FPR: 0.06      |
Threshold: -0.45| Accuracy: 0.83| Precision: 0.67     | Recall: 0.41
      | F1: 0.26 | FPR: 0.06      |
Threshold: -0.40| Accuracy: 0.83| Precision: 0.67     | Recall: 0.41
      | F1: 0.26 | FPR: 0.06      |
Threshold: -0.35| Accuracy: 0.83| Precision: 0.71     | Recall: 0.41
      | F1: 0.26 | FPR: 0.05      |
Threshold: -0.30| Accuracy: 0.83| Precision: 0.71     | Recall: 0.41
      | F1: 0.26 | FPR: 0.05      |
Threshold: -0.25| Accuracy: 0.83| Precision: 0.71     | Recall: 0.41
      | F1: 0.26 | FPR: 0.05      |
Threshold: -0.20| Accuracy: 0.83| Precision: 0.71     | Recall: 0.41
      | F1: 0.26 | FPR: 0.05      |
Threshold: -0.15| Accuracy: 0.83| Precision: 0.71     | Recall: 0.41
      | F1: 0.26 | FPR: 0.05      |
Threshold: -0.10| Accuracy: 0.83| Precision: 0.71     | Recall: 0.41
      | F1: 0.26 | FPR: 0.05      |
Threshold: -0.05| Accuracy: 0.83| Precision: 0.69     | Recall: 0.38
      | F1: 0.24 | FPR: 0.05      |
Threshold: -0.00| Accuracy: 0.83| Precision: 0.69     | Recall: 0.38
      | F1: 0.24 | FPR: 0.05      |
Threshold: 0.05  | Accuracy: 0.83| Precision: 0.69     | Recall: 0.38
      | F1: 0.24 | FPR: 0.05      |
Threshold: 0.10  | Accuracy: 0.82| Precision: 0.67     | Recall: 0.34
      | F1: 0.23 | FPR: 0.05      |
Threshold: 0.15  | Accuracy: 0.81| Precision: 0.64     | Recall: 0.31
      | F1: 0.21 | FPR: 0.05      |
Threshold: 0.20  | Accuracy: 0.82| Precision: 0.69     | Recall: 0.31
      | F1: 0.21 | FPR: 0.04      |
Threshold: 0.25  | Accuracy: 0.81| Precision: 0.67     | Recall: 0.28
      | F1: 0.20 | FPR: 0.04      |
Threshold: 0.30  | Accuracy: 0.81| Precision: 0.67     | Recall: 0.28
      | F1: 0.20 | FPR: 0.04      |
Threshold: 0.35  | Accuracy: 0.82| Precision: 0.73     | Recall: 0.28
      | F1: 0.20 | FPR: 0.03      |
Threshold: 0.40  | Accuracy: 0.82| Precision: 0.73     | Recall: 0.28
      | F1: 0.20 | FPR: 0.03      |
Threshold: 0.45  | Accuracy: 0.82| Precision: 0.73     | Recall: 0.28
      | F1: 0.20 | FPR: 0.03      |
Threshold: 0.50  | Accuracy: 0.82| Precision: 0.73     | Recall: 0.28
      | F1: 0.20 | FPR: 0.03      |
Threshold: 0.55  | Accuracy: 0.82| Precision: 0.73     | Recall: 0.28
      | F1: 0.20 | FPR: 0.03      |
Threshold: 0.60  | Accuracy: 0.82| Precision: 0.73     | Recall: 0.28
```

```
     | F1: 0.20 | FPR: 0.03     |
Threshold: 0.65 | Accuracy: 0.82| Precision: 0.73    | Recall: 0.28
     | F1: 0.20 | FPR: 0.03     |
Threshold: 0.70 | Accuracy: 0.83| Precision: 0.80    | Recall: 0.28
     | F1: 0.21 | FPR: 0.02     |
Threshold: 0.75 | Accuracy: 0.83| Precision: 0.80    | Recall: 0.28
     | F1: 0.21 | FPR: 0.02     |
Threshold: 0.80 | Accuracy: 0.83| Precision: 0.80    | Recall: 0.28
     | F1: 0.21 | FPR: 0.02     |
Threshold: 0.85 | Accuracy: 0.83| Precision: 0.80    | Recall: 0.28
     | F1: 0.21 | FPR: 0.02     |
Threshold: 0.90 | Accuracy: 0.83| Precision: 0.80    | Recall: 0.28
     | F1: 0.21 | FPR: 0.02     |
Threshold: 0.95 | Accuracy: 0.83| Precision: 0.80    | Recall: 0.28
     | F1: 0.21 | FPR: 0.02     |
Threshold: 1.00 | Accuracy: 0.83| Precision: 0.80    | Recall: 0.28
     | F1: 0.21 | FPR: 0.02     |
Threshold: 1.05 | Accuracy: 0.83| Precision: 0.89    | Recall: 0.28
     | F1: 0.21 | FPR: 0.01     |
Threshold: 1.10 | Accuracy: 0.83| Precision: 0.89    | Recall: 0.28
     | F1: 0.21 | FPR: 0.01     |
Threshold: 1.15 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.20 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.25 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.30 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.35 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.40 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.45 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.50 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.55 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.60 | Accuracy: 0.83| Precision: 0.87    | Recall: 0.24
     | F1: 0.19 | FPR: 0.01     |
Threshold: 1.65 | Accuracy: 0.82| Precision: 0.86    | Recall: 0.21
     | F1: 0.17 | FPR: 0.01     |
Threshold: 1.70 | Accuracy: 0.82| Precision: 0.86    | Recall: 0.21
     | F1: 0.17 | FPR: 0.01     |
Threshold: 1.75 | Accuracy: 0.81| Precision: 0.83    | Recall: 0.17
     | F1: 0.14 | FPR: 0.01     |
Threshold: 1.80 | Accuracy: 0.81| Precision: 0.83    | Recall: 0.17
     | F1: 0.14 | FPR: 0.01     |
```

```
Threshold: 1.85 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 1.90 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 1.95 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.00 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.05 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.10 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.15 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.20 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.25 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.30 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.35 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.40 | Accuracy: 0.81| Precision: 0.83     | Recall: 0.17
      | F1: 0.14 | FPR: 0.01      |
Threshold: 2.45 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
      | F1: 0.12 | FPR: 0.01      |
Threshold: 2.50 | Accuracy: 0.80| Precision: 0.75     | Recall: 0.10
      | F1: 0.09 | FPR: 0.01      |
Threshold: 2.55 | Accuracy: 0.80| Precision: 0.75     | Recall: 0.10
      | F1: 0.09 | FPR: 0.01      |
Threshold: 2.60 | Accuracy: 0.80| Precision: 0.75     | Recall: 0.10
      | F1: 0.09 | FPR: 0.01      |
Threshold: 2.65 | Accuracy: 0.80| Precision: 0.75     | Recall: 0.10
      | F1: 0.09 | FPR: 0.01      |
Threshold: 2.70 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 2.75 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 2.80 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.07
      | F1: 0.06 | FPR: 0.00      |
Threshold: 2.85 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.07
      | F1: 0.06 | FPR: 0.00      |
Threshold: 2.90 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.07
      | F1: 0.06 | FPR: 0.00      |
Threshold: 2.95 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.07
      | F1: 0.06 | FPR: 0.00      |
Threshold: 3.00 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.07
      | F1: 0.06 | FPR: 0.00      |
Threshold: 3.05 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.07
```

```
     | F1: 0.06 | FPR: 0.00     |
Threshold: 3.10 | Accuracy: 0.79| Precision: 1.00    | Recall: 0.03
     | F1: 0.03 | FPR: 0.00     |
Threshold: 3.15 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.20 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.25 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.30 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.35 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.40 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.45 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.50 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.55 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.60 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.65 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.70 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.75 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.80 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.85 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.90 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 3.95 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 4.00 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 4.05 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 4.10 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 4.15 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 4.20 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
Threshold: 4.25 | Accuracy: 0.78| Precision: 0.00    | Recall: 0.00
     | F1: 0.00 | FPR: 0.00     |
```

```
Threshold: 4.30 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.35 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.40 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.45 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.50 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.55 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.60 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.65 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.70 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.75 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.80 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.85 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.90 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Threshold: 4.95 | Accuracy: 0.78| Precision: 0.00     | Recall: 0.00
      | F1: 0.00 | FPR: 0.00      |
Best threshold for accuracy: -0.35 (Accuracy: 0.83)
Best threshold for F1: -0.35 (F1: 0.26)

model.fit_params(x_train, y_train, 10)

best_acc_bin10 = float('-inf')
best_acc_t_bin10 = 0

best_f1_bin10 = float('-inf')
best_f1_t_bin10 = 0

hist_bin10 = {
    'Accuracy': [],
    'Precision': [],
    'Recall': [],
    'F1': [],
    'FPR': []
}

for threshold in np.arange(-5, 5, 0.05):
    y_pred = model.predict(x_test, threshold)
```

```
    print(f"Threshold: {threshold:.2f}",end='\t| ')
    eva = evaluate(y_test, y_pred, show_result=False)
    for idx, metric in enumerate(['Accuracy', 'Precision', 'Recall',
'F1', 'FPR']):
        print(f"{metric}: {eva[idx]:.2f}", end='\t| ')
        hist_bin10[metric].append(eva[idx])
    print()
    if eva[0] > best_acc_bin10:
        best_acc_bin10 = eva[0]
        best_acc_t_bin10 = threshold
    if eva[3] > best_f1_bin10:
        best_f1_bin10 = eva[3]
        best_f1_t_bin10 = threshold

print(f"Best threshold for accuracy: {best_acc_t_bin10:.2f} (Accuracy:
{best_acc_bin10:.2f})")
print(f"Best threshold for F1: {best_f1_t_bin10:.2f} (F1:
{best_f1_bin10:.2f})")
```

```
Threshold: -5.00| Accuracy: 0.49| Precision: 0.28     | Recall: 0.86
      | F1: 0.21 | FPR: 0.61      |
Threshold: -4.95| Accuracy: 0.50| Precision: 0.29     | Recall: 0.86
      | F1: 0.22 | FPR: 0.60      |
Threshold: -4.90| Accuracy: 0.50| Precision: 0.29     | Recall: 0.86
      | F1: 0.22 | FPR: 0.60      |
Threshold: -4.85| Accuracy: 0.51| Precision: 0.29     | Recall: 0.86
      | F1: 0.22 | FPR: 0.59      |
Threshold: -4.80| Accuracy: 0.52| Precision: 0.29     | Recall: 0.86
      | F1: 0.22 | FPR: 0.58      |
Threshold: -4.75| Accuracy: 0.52| Precision: 0.29     | Recall: 0.86
      | F1: 0.22 | FPR: 0.58      |
Threshold: -4.70| Accuracy: 0.52| Precision: 0.29     | Recall: 0.86
      | F1: 0.22 | FPR: 0.58      |
Threshold: -4.65| Accuracy: 0.52| Precision: 0.30     | Recall: 0.86
      | F1: 0.22 | FPR: 0.57      |
Threshold: -4.60| Accuracy: 0.53| Precision: 0.30     | Recall: 0.86
      | F1: 0.22 | FPR: 0.56      |
Threshold: -4.55| Accuracy: 0.53| Precision: 0.30     | Recall: 0.86
      | F1: 0.22 | FPR: 0.56      |
Threshold: -4.50| Accuracy: 0.53| Precision: 0.30     | Recall: 0.86
      | F1: 0.22 | FPR: 0.56      |
Threshold: -4.45| Accuracy: 0.53| Precision: 0.30     | Recall: 0.86
      | F1: 0.22 | FPR: 0.56      |
Threshold: -4.40| Accuracy: 0.53| Precision: 0.30     | Recall: 0.86
      | F1: 0.22 | FPR: 0.56      |
Threshold: -4.35| Accuracy: 0.55| Precision: 0.30     | Recall: 0.83
      | F1: 0.22 | FPR: 0.53      |
Threshold: -4.30| Accuracy: 0.55| Precision: 0.31     | Recall: 0.83
      | F1: 0.22 | FPR: 0.52      |
Threshold: -4.25| Accuracy: 0.55| Precision: 0.30     | Recall: 0.79
```

```
      | F1: 0.22 | FPR: 0.52      |
Threshold: -4.20| Accuracy: 0.55| Precision: 0.30     | Recall: 0.79
      | F1: 0.22 | FPR: 0.52      |
Threshold: -4.15| Accuracy: 0.55| Precision: 0.30     | Recall: 0.79
      | F1: 0.22 | FPR: 0.52      |
Threshold: -4.10| Accuracy: 0.55| Precision: 0.30     | Recall: 0.79
      | F1: 0.22 | FPR: 0.51      |
Threshold: -4.05| Accuracy: 0.56| Precision: 0.31     | Recall: 0.79
      | F1: 0.22 | FPR: 0.50      |
Threshold: -4.00| Accuracy: 0.56| Precision: 0.31     | Recall: 0.79
      | F1: 0.22 | FPR: 0.50      |
Threshold: -3.95| Accuracy: 0.57| Precision: 0.31     | Recall: 0.79
      | F1: 0.22 | FPR: 0.50      |
Threshold: -3.90| Accuracy: 0.57| Precision: 0.31     | Recall: 0.79
      | F1: 0.22 | FPR: 0.50      |
Threshold: -3.85| Accuracy: 0.57| Precision: 0.31     | Recall: 0.79
      | F1: 0.22 | FPR: 0.50      |
Threshold: -3.80| Accuracy: 0.58| Precision: 0.32     | Recall: 0.79
      | F1: 0.23 | FPR: 0.49      |
Threshold: -3.75| Accuracy: 0.58| Precision: 0.31     | Recall: 0.76
      | F1: 0.22 | FPR: 0.48      |
Threshold: -3.70| Accuracy: 0.58| Precision: 0.30     | Recall: 0.72
      | F1: 0.21 | FPR: 0.47      |
Threshold: -3.65| Accuracy: 0.59| Precision: 0.31     | Recall: 0.72
      | F1: 0.22 | FPR: 0.45      |
Threshold: -3.60| Accuracy: 0.59| Precision: 0.31     | Recall: 0.72
      | F1: 0.22 | FPR: 0.45      |
Threshold: -3.55| Accuracy: 0.59| Precision: 0.31     | Recall: 0.72
      | F1: 0.22 | FPR: 0.45      |
Threshold: -3.50| Accuracy: 0.59| Precision: 0.31     | Recall: 0.72
      | F1: 0.22 | FPR: 0.45      |
Threshold: -3.45| Accuracy: 0.59| Precision: 0.31     | Recall: 0.72
      | F1: 0.22 | FPR: 0.45      |
Threshold: -3.40| Accuracy: 0.60| Precision: 0.32     | Recall: 0.72
      | F1: 0.22 | FPR: 0.44      |
Threshold: -3.35| Accuracy: 0.60| Precision: 0.32     | Recall: 0.72
      | F1: 0.22 | FPR: 0.44      |
Threshold: -3.30| Accuracy: 0.61| Precision: 0.32     | Recall: 0.72
      | F1: 0.22 | FPR: 0.43      |
Threshold: -3.25| Accuracy: 0.61| Precision: 0.32     | Recall: 0.72
      | F1: 0.22 | FPR: 0.43      |
Threshold: -3.20| Accuracy: 0.62| Precision: 0.33     | Recall: 0.72
      | F1: 0.23 | FPR: 0.41      |
Threshold: -3.15| Accuracy: 0.62| Precision: 0.33     | Recall: 0.72
      | F1: 0.23 | FPR: 0.41      |
Threshold: -3.10| Accuracy: 0.62| Precision: 0.33     | Recall: 0.72
      | F1: 0.23 | FPR: 0.41      |
Threshold: -3.05| Accuracy: 0.63| Precision: 0.34     | Recall: 0.72
      | F1: 0.23 | FPR: 0.40      |
```

```
Threshold: -3.00| Accuracy: 0.64| Precision: 0.34     | Recall: 0.72
      | F1: 0.23 | FPR: 0.39      |
Threshold: -2.95| Accuracy: 0.64| Precision: 0.34     | Recall: 0.72
      | F1: 0.23 | FPR: 0.39      |
Threshold: -2.90| Accuracy: 0.64| Precision: 0.35     | Recall: 0.72
      | F1: 0.24 | FPR: 0.38      |
Threshold: -2.85| Accuracy: 0.65| Precision: 0.36     | Recall: 0.72
      | F1: 0.24 | FPR: 0.37      |
Threshold: -2.80| Accuracy: 0.65| Precision: 0.36     | Recall: 0.72
      | F1: 0.24 | FPR: 0.37      |
Threshold: -2.75| Accuracy: 0.65| Precision: 0.36     | Recall: 0.72
      | F1: 0.24 | FPR: 0.37      |
Threshold: -2.70| Accuracy: 0.65| Precision: 0.36     | Recall: 0.72
      | F1: 0.24 | FPR: 0.37      |
Threshold: -2.65| Accuracy: 0.65| Precision: 0.36     | Recall: 0.72
      | F1: 0.24 | FPR: 0.37      |
Threshold: -2.60| Accuracy: 0.65| Precision: 0.36     | Recall: 0.72
      | F1: 0.24 | FPR: 0.37      |
Threshold: -2.55| Accuracy: 0.66| Precision: 0.36     | Recall: 0.72
      | F1: 0.24 | FPR: 0.36      |
Threshold: -2.50| Accuracy: 0.67| Precision: 0.37     | Recall: 0.72
      | F1: 0.25 | FPR: 0.34      |
Threshold: -2.45| Accuracy: 0.69| Precision: 0.39     | Recall: 0.72
      | F1: 0.25 | FPR: 0.32      |
Threshold: -2.40| Accuracy: 0.69| Precision: 0.39     | Recall: 0.72
      | F1: 0.25 | FPR: 0.32      |
Threshold: -2.35| Accuracy: 0.70| Precision: 0.40     | Recall: 0.72
      | F1: 0.26 | FPR: 0.30      |
Threshold: -2.30| Accuracy: 0.71| Precision: 0.41     | Recall: 0.72
      | F1: 0.26 | FPR: 0.29      |
Threshold: -2.25| Accuracy: 0.72| Precision: 0.42     | Recall: 0.72
      | F1: 0.27 | FPR: 0.28      |
Threshold: -2.20| Accuracy: 0.72| Precision: 0.42     | Recall: 0.72
      | F1: 0.27 | FPR: 0.28      |
Threshold: -2.15| Accuracy: 0.71| Precision: 0.41     | Recall: 0.69
      | F1: 0.26 | FPR: 0.28      |
Threshold: -2.10| Accuracy: 0.73| Precision: 0.43     | Recall: 0.69
      | F1: 0.26 | FPR: 0.26      |
Threshold: -2.05| Accuracy: 0.72| Precision: 0.41     | Recall: 0.66
      | F1: 0.25 | FPR: 0.26      |
Threshold: -2.00| Accuracy: 0.73| Precision: 0.43     | Recall: 0.66
      | F1: 0.26 | FPR: 0.24      |
Threshold: -1.95| Accuracy: 0.73| Precision: 0.43     | Recall: 0.66
      | F1: 0.26 | FPR: 0.24      |
Threshold: -1.90| Accuracy: 0.74| Precision: 0.44     | Recall: 0.66
      | F1: 0.26 | FPR: 0.23      |
Threshold: -1.85| Accuracy: 0.74| Precision: 0.44     | Recall: 0.62
      | F1: 0.26 | FPR: 0.22      |
Threshold: -1.80| Accuracy: 0.74| Precision: 0.44     | Recall: 0.62
```

```
      | F1: 0.26 | FPR: 0.22      |
Threshold: -1.75| Accuracy: 0.75| Precision: 0.45     | Recall: 0.62
      | F1: 0.26 | FPR: 0.21      |
Threshold: -1.70| Accuracy: 0.75| Precision: 0.45     | Recall: 0.62
      | F1: 0.26 | FPR: 0.21      |
Threshold: -1.65| Accuracy: 0.75| Precision: 0.45     | Recall: 0.62
      | F1: 0.26 | FPR: 0.21      |
Threshold: -1.60| Accuracy: 0.75| Precision: 0.45     | Recall: 0.62
      | F1: 0.26 | FPR: 0.21      |
Threshold: -1.55| Accuracy: 0.75| Precision: 0.45     | Recall: 0.62
      | F1: 0.26 | FPR: 0.21      |
Threshold: -1.50| Accuracy: 0.77| Precision: 0.47     | Recall: 0.62
      | F1: 0.27 | FPR: 0.19      |
Threshold: -1.45| Accuracy: 0.77| Precision: 0.49     | Recall: 0.62
      | F1: 0.27 | FPR: 0.18      |
Threshold: -1.40| Accuracy: 0.77| Precision: 0.47     | Recall: 0.59
      | F1: 0.26 | FPR: 0.18      |
Threshold: -1.35| Accuracy: 0.77| Precision: 0.49     | Recall: 0.59
      | F1: 0.27 | FPR: 0.17      |
Threshold: -1.30| Accuracy: 0.78| Precision: 0.50     | Recall: 0.59
      | F1: 0.27 | FPR: 0.17      |
Threshold: -1.25| Accuracy: 0.78| Precision: 0.50     | Recall: 0.59
      | F1: 0.27 | FPR: 0.17      |
Threshold: -1.20| Accuracy: 0.78| Precision: 0.50     | Recall: 0.59
      | F1: 0.27 | FPR: 0.17      |
Threshold: -1.15| Accuracy: 0.77| Precision: 0.48     | Recall: 0.55
      | F1: 0.26 | FPR: 0.17      |
Threshold: -1.10| Accuracy: 0.77| Precision: 0.48     | Recall: 0.52
      | F1: 0.25 | FPR: 0.16      |
Threshold: -1.05| Accuracy: 0.77| Precision: 0.48     | Recall: 0.52
      | F1: 0.25 | FPR: 0.16      |
Threshold: -1.00| Accuracy: 0.77| Precision: 0.48     | Recall: 0.52
      | F1: 0.25 | FPR: 0.16      |
Threshold: -0.95| Accuracy: 0.77| Precision: 0.48     | Recall: 0.52
      | F1: 0.25 | FPR: 0.16      |
Threshold: -0.90| Accuracy: 0.77| Precision: 0.48     | Recall: 0.52
      | F1: 0.25 | FPR: 0.16      |
Threshold: -0.85| Accuracy: 0.77| Precision: 0.48     | Recall: 0.52
      | F1: 0.25 | FPR: 0.16      |
Threshold: -0.80| Accuracy: 0.77| Precision: 0.48     | Recall: 0.52
      | F1: 0.25 | FPR: 0.16      |
Threshold: -0.75| Accuracy: 0.77| Precision: 0.48     | Recall: 0.52
      | F1: 0.25 | FPR: 0.16      |
Threshold: -0.70| Accuracy: 0.77| Precision: 0.47     | Recall: 0.48
      | F1: 0.24 | FPR: 0.16      |
Threshold: -0.65| Accuracy: 0.77| Precision: 0.48     | Recall: 0.48
      | F1: 0.24 | FPR: 0.15      |
Threshold: -0.60| Accuracy: 0.78| Precision: 0.50     | Recall: 0.48
      | F1: 0.25 | FPR: 0.14      |
```

```
Threshold: -0.55| Accuracy: 0.78| Precision: 0.50     | Recall: 0.48
      | F1: 0.25 | FPR: 0.14      |
Threshold: -0.50| Accuracy: 0.80| Precision: 0.54     | Recall: 0.48
      | F1: 0.25 | FPR: 0.12      |
Threshold: -0.45| Accuracy: 0.80| Precision: 0.54     | Recall: 0.48
      | F1: 0.25 | FPR: 0.12      |
Threshold: -0.40| Accuracy: 0.80| Precision: 0.54     | Recall: 0.48
      | F1: 0.25 | FPR: 0.12      |
Threshold: -0.35| Accuracy: 0.80| Precision: 0.56     | Recall: 0.48
      | F1: 0.26 | FPR: 0.11      |
Threshold: -0.30| Accuracy: 0.80| Precision: 0.56     | Recall: 0.48
      | F1: 0.26 | FPR: 0.11      |
Threshold: -0.25| Accuracy: 0.80| Precision: 0.56     | Recall: 0.48
      | F1: 0.26 | FPR: 0.11      |
Threshold: -0.20| Accuracy: 0.79| Precision: 0.52     | Recall: 0.41
      | F1: 0.23 | FPR: 0.11      |
Threshold: -0.15| Accuracy: 0.79| Precision: 0.52     | Recall: 0.41
      | F1: 0.23 | FPR: 0.11      |
Threshold: -0.10| Accuracy: 0.79| Precision: 0.52     | Recall: 0.41
      | F1: 0.23 | FPR: 0.11      |
Threshold: -0.05| Accuracy: 0.79| Precision: 0.52     | Recall: 0.41
      | F1: 0.23 | FPR: 0.11      |
Threshold: -0.00| Accuracy: 0.79| Precision: 0.52     | Recall: 0.41
      | F1: 0.23 | FPR: 0.11      |
Threshold: 0.05 | Accuracy: 0.79| Precision: 0.52     | Recall: 0.41
      | F1: 0.23 | FPR: 0.11      |
Threshold: 0.10 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.38
      | F1: 0.22 | FPR: 0.11      |
Threshold: 0.15 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.38
      | F1: 0.22 | FPR: 0.11      |
Threshold: 0.20 | Accuracy: 0.79| Precision: 0.52     | Recall: 0.38
      | F1: 0.22 | FPR: 0.10      |
Threshold: 0.25 | Accuracy: 0.79| Precision: 0.52     | Recall: 0.38
      | F1: 0.22 | FPR: 0.10      |
Threshold: 0.30 | Accuracy: 0.80| Precision: 0.55     | Recall: 0.38
      | F1: 0.22 | FPR: 0.09      |
Threshold: 0.35 | Accuracy: 0.80| Precision: 0.58     | Recall: 0.38
      | F1: 0.23 | FPR: 0.08      |
Threshold: 0.40 | Accuracy: 0.80| Precision: 0.58     | Recall: 0.38
      | F1: 0.23 | FPR: 0.08      |
Threshold: 0.45 | Accuracy: 0.80| Precision: 0.56     | Recall: 0.34
      | F1: 0.21 | FPR: 0.08      |
Threshold: 0.50 | Accuracy: 0.80| Precision: 0.56     | Recall: 0.34
      | F1: 0.21 | FPR: 0.08      |
Threshold: 0.55 | Accuracy: 0.80| Precision: 0.56     | Recall: 0.34
      | F1: 0.21 | FPR: 0.08      |
Threshold: 0.60 | Accuracy: 0.80| Precision: 0.56     | Recall: 0.34
      | F1: 0.21 | FPR: 0.08      |
Threshold: 0.65 | Accuracy: 0.80| Precision: 0.56     | Recall: 0.34
```

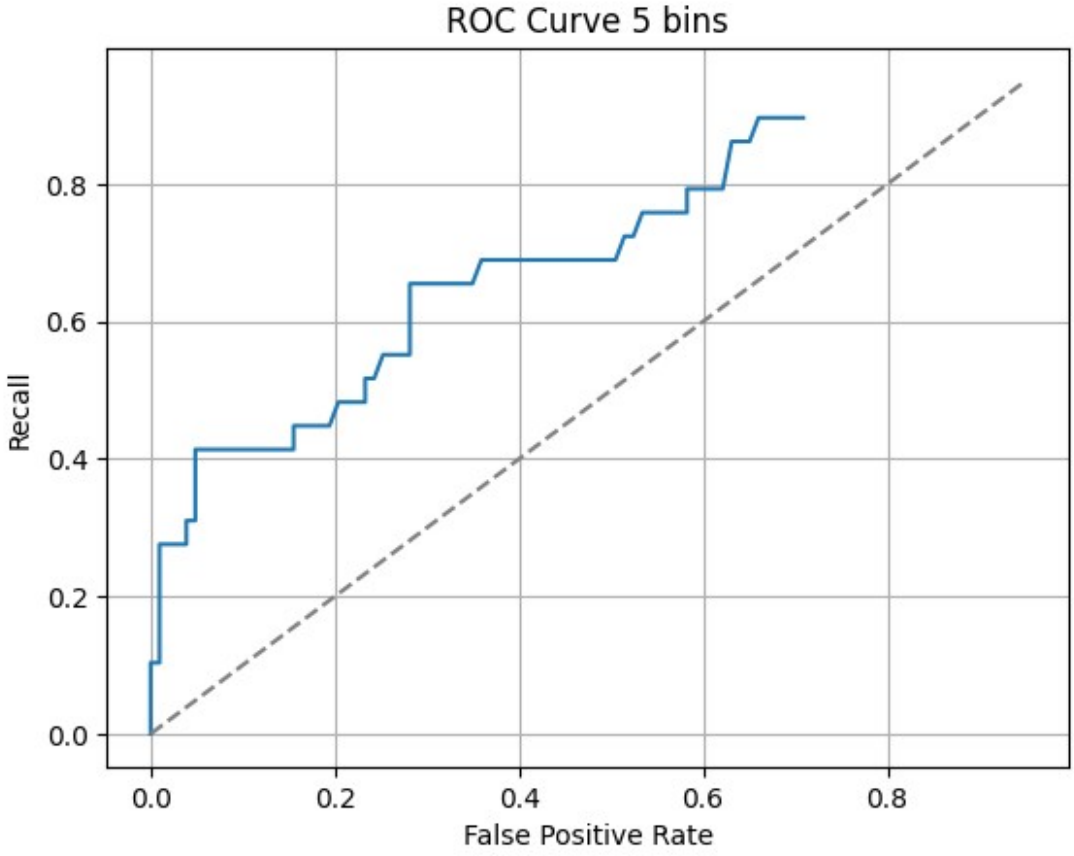
```
     | F1: 0.21 | FPR: 0.08     |
Threshold: 0.70 | Accuracy: 0.80| Precision: 0.56     | Recall: 0.34
     | F1: 0.21 | FPR: 0.08     |
Threshold: 0.75 | Accuracy: 0.80| Precision: 0.56     | Recall: 0.34
     | F1: 0.21 | FPR: 0.08     |
Threshold: 0.80 | Accuracy: 0.79| Precision: 0.53     | Recall: 0.31
     | F1: 0.20 | FPR: 0.08     |
Threshold: 0.85 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.28
     | F1: 0.18 | FPR: 0.08     |
Threshold: 0.90 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.28
     | F1: 0.18 | FPR: 0.08     |
Threshold: 0.95 | Accuracy: 0.77| Precision: 0.47     | Recall: 0.24
     | F1: 0.16 | FPR: 0.08     |
Threshold: 1.00 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.24
     | F1: 0.16 | FPR: 0.07     |
Threshold: 1.05 | Accuracy: 0.77| Precision: 0.46     | Recall: 0.21
     | F1: 0.14 | FPR: 0.07     |
Threshold: 1.10 | Accuracy: 0.77| Precision: 0.46     | Recall: 0.21
     | F1: 0.14 | FPR: 0.07     |
Threshold: 1.15 | Accuracy: 0.77| Precision: 0.46     | Recall: 0.21
     | F1: 0.14 | FPR: 0.07     |
Threshold: 1.20 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.21
     | F1: 0.15 | FPR: 0.06     |
Threshold: 1.25 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.21
     | F1: 0.15 | FPR: 0.06     |
Threshold: 1.30 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.21
     | F1: 0.15 | FPR: 0.06     |
Threshold: 1.35 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.40 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.45 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.50 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.55 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.60 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.65 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.70 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.75 | Accuracy: 0.79| Precision: 0.55     | Recall: 0.21
     | F1: 0.15 | FPR: 0.05     |
Threshold: 1.80 | Accuracy: 0.79| Precision: 0.56     | Recall: 0.17
     | F1: 0.13 | FPR: 0.04     |
Threshold: 1.85 | Accuracy: 0.79| Precision: 0.56     | Recall: 0.17
     | F1: 0.13 | FPR: 0.04     |
```

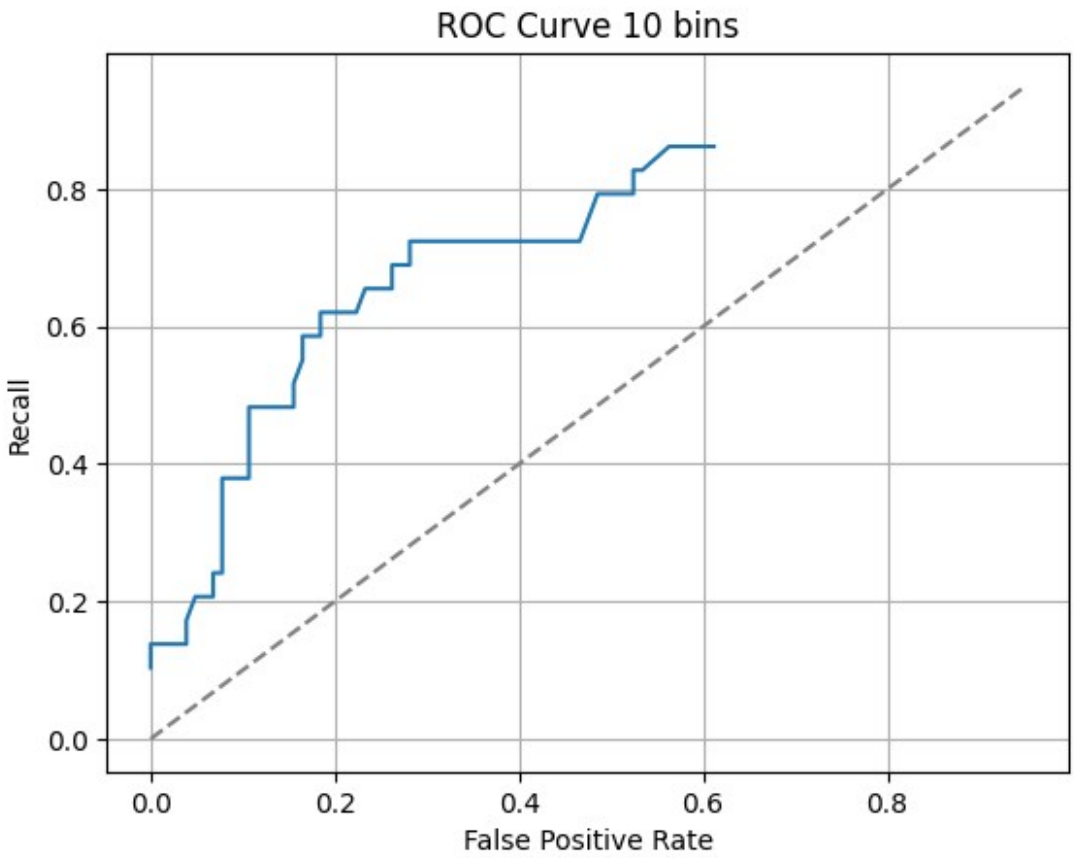
```
Threshold: 1.90 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 1.95 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.00 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.05 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.10 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.15 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.20 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.25 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.30 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.35 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.40 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.45 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.50 | Accuracy: 0.78| Precision: 0.50     | Recall: 0.14
      | F1: 0.11 | FPR: 0.04      |
Threshold: 2.55 | Accuracy: 0.79| Precision: 0.57     | Recall: 0.14
      | F1: 0.11 | FPR: 0.03      |
Threshold: 2.60 | Accuracy: 0.79| Precision: 0.57     | Recall: 0.14
      | F1: 0.11 | FPR: 0.03      |
Threshold: 2.65 | Accuracy: 0.79| Precision: 0.57     | Recall: 0.14
      | F1: 0.11 | FPR: 0.03      |
Threshold: 2.70 | Accuracy: 0.79| Precision: 0.57     | Recall: 0.14
      | F1: 0.11 | FPR: 0.03      |
Threshold: 2.75 | Accuracy: 0.79| Precision: 0.57     | Recall: 0.14
      | F1: 0.11 | FPR: 0.03      |
Threshold: 2.80 | Accuracy: 0.79| Precision: 0.57     | Recall: 0.14
      | F1: 0.11 | FPR: 0.03      |
Threshold: 2.85 | Accuracy: 0.80| Precision: 0.67     | Recall: 0.14
      | F1: 0.11 | FPR: 0.02      |
Threshold: 2.90 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
      | F1: 0.12 | FPR: 0.01      |
Threshold: 2.95 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
      | F1: 0.12 | FPR: 0.01      |
Threshold: 3.00 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
      | F1: 0.12 | FPR: 0.01      |
Threshold: 3.05 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
      | F1: 0.12 | FPR: 0.01      |
Threshold: 3.10 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
```

```
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.15 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.20 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.25 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.30 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.35 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.40 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.45 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.50 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.55 | Accuracy: 0.80| Precision: 0.80     | Recall: 0.14
     | F1: 0.12 | FPR: 0.01     |
Threshold: 3.60 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 3.65 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 3.70 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 3.75 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 3.80 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 3.85 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 3.90 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 3.95 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 4.00 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 4.05 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 4.10 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 4.15 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 4.20 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 4.25 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
Threshold: 4.30 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
     | F1: 0.12 | FPR: 0.00     |
```

```
Threshold: 4.35 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
      | F1: 0.12 | FPR: 0.00      |
Threshold: 4.40 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
      | F1: 0.12 | FPR: 0.00      |
Threshold: 4.45 | Accuracy: 0.81| Precision: 1.00     | Recall: 0.14
      | F1: 0.12 | FPR: 0.00      |
Threshold: 4.50 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.55 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.60 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.65 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.70 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.75 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.80 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.85 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.90 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Threshold: 4.95 | Accuracy: 0.80| Precision: 1.00     | Recall: 0.10
      | F1: 0.09 | FPR: 0.00      |
Best threshold for accuracy: 3.60 (Accuracy: 0.81)
Best threshold for F1: -1.45 (F1: 0.27)

plt.plot(hist_bin5['FPR'], hist_bin5['Recall'])
plt.title('ROC Curve 5 bins')
plt.xlabel('False Positive Rate')
plt.ylabel('Recall')

plt.plot(np.arange(0, 1, 0.05), np.arange(0, 1, 0.05), linestyle='--',
color='gray')
plt.grid()

plt.show()


plt.plot(hist_bin10['FPR'], hist_bin10['Recall'])
plt.title('ROC Curve 10 bins')
plt.xlabel('False Positive Rate')
plt.ylabel('Recall')

plt.plot(np.arange(0, 1, 0.05), np.arange(0, 1, 0.05), linestyle='--',
color='gray')
plt.grid()
```

```
plt.show()
```





T19

Submitted the code

OT4

```
n_round=10
lst_accuracy=[]
for i in range(n_round):
#########
    X_train, X_test, y_train, y_test=train_test_split(df.loc[:,
~df.columns.isin(['Attrition'])],df["Attrition"],test_size=0.1,shuffle
=True)
    X_train_leave = X_train.loc[df["Attrition"] == 1.0].copy()
    X_train_stay  = X_train.loc[df["Attrition"] == 0.0].copy()
    OT3model = SimpleBayesClassifier(n_pos =X_train_leave.shape[0] ,
n_neg = X_train_stay.shape[0])
    a,b=model.fit_params(X_train.to_numpy(), y_train.to_numpy())
    y_pred = model.predict(X_test.to_numpy())
    accuracy, precision, recall, F1, fpr =
evaluate(y_test.to_numpy(),y_pred, show_result=False)
    lst_accuracy.append(accuracy)
    lst_accuracy += [accuracy]
```

```python
print("Mean : ", np.mean(lst_accuracy))
print("Variance :", np.var(lst_accuracy))
```

```
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
```

```
Mean :  0.8204081627072053
Variance : 0.0008626035436456782
```

```
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
c:\Users\Vivobook\github\my-chula-courses\2110573-patt-recog\HW2\
SimpleBayesClassifier.py:94: RuntimeWarning: divide by zero
encountered in log
  h += np.log(self.leave_params[j][0][leave_bin_idx])
```