In [21]:
```
!pip install joblib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,recall_score,confusion_matrix,class
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn. ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.svm import SVC
from joblib import dump,load
from sklearn.compose import ColumnTransformer

df=pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
print(df.head(10))
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: joblib in c:\users\dell\appdata\roaming\python\p
ython313\site-packages (1.5.3)
[notice] A new release of pip is available: 24.3.1 -> 25.3
[notice] To update, run: C:\Python313\python.exe -m pip install --upgrade pip
```

	Age	Attrition	BusinessTravel	DailyRate	Department	\
0	41	Yes	Travel_Rarely	1102	Sales	
1	49	No	Travel_Frequently	279	Research & Development	
2	37	Yes	Travel_Rarely	1373	Research & Development	
3	33	No	Travel_Frequently	1392	Research & Development	
4	27	No	Travel_Rarely	591	Research & Development	
5	32	No	Travel_Frequently	1005	Research & Development	
6	59	No	Travel_Rarely	1324	Research & Development	
7	30	No	Travel_Rarely	1358	Research & Development	
8	38	No	Travel_Frequently	216	Research & Development	
9	36	No	Travel_Rarely	1299	Research & Development	

	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	\
0	1	2	Life Sciences	1	1	
1	8	1	Life Sciences	1	2	
2	2	2	Other	1	4	
3	3	4	Life Sciences	1	5	
4	2	1	Medical	1	7	
5	2	2	Life Sciences	1	8	
6	3	3	Medical	1	10	
7	24	1	Life Sciences	1	11	
8	23	3	Life Sciences	1	12	
9	27	3	Medical	1	13	

	...	RelationshipSatisfaction	StandardHours	StockOptionLevel	\
0	...	1	80	0	
1	...	4	80	1	
2	...	2	80	0	
3	...	3	80	0	
4	...	4	80	1	
5	...	3	80	0	
6	...	1	80	3	
7	...	2	80	1	
8	...	2	80	0	
9	...	2	80	2	

	TotalWorkingYears	TrainingTimesLastYear	WorkLifeBalance	YearsAtCompany	\
0	8	0	1	6	
1	10	3	3	10	
2	7	3	3	0	
3	8	3	3	8	
4	6	3	3	2	
5	8	2	2	7	
6	12	3	2	1	
7	1	2	3	1	
8	10	2	3	9	
9	17	3	2	7	

	YearsInCurrentRole	YearsSinceLastPromotion	YearsWithCurrManager
0	4	0	5
1	7	1	7
2	0	0	0
3	7	3	0
4	2	2	2

```
5                  7                  3                  6
6                  0                  0                  0
7                  0                  0                  0
8                  7                  1                  8
9                  7                  7                  7

[10 rows x 35 columns]
```

In [22]:

```
print("="*20,"Data Information","="*20)
print(df.info())
print("="*20,"Data Description","="*20)
print(df.describe())
print("="*20,"Dataset Shape","="*20)
print(df.shape)
print("="*20,"Identifying and Removing null values","="*20)
print(df.isna())
print(df.dropna())
```

```
==================== Data Information ====================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
None
==================== Data Description ====================
               Age    DailyRate  DistanceFromHome    Education  EmployeeCount
\
count  1470.000000  1470.000000       1470.000000  1470.000000         1470.0
mean     36.923810   802.485714          9.192517     2.912925            1.0
std       9.135373   403.509100          8.106864     1.024165            0.0
min      18.000000   102.000000          1.000000     1.000000            1.0
25%      30.000000   465.000000          2.000000     2.000000            1.0
50%      36.000000   802.000000          7.000000     3.000000            1.0
75%      43.000000  1157.000000         14.000000     4.000000            1.0
```

```
max      60.000000  1499.000000         29.000000     5.000000            1.0

       EmployeeNumber  EnvironmentSatisfaction   HourlyRate  JobInvolvement  \
count     1470.000000              1470.000000  1470.000000     1470.000000
mean      1024.865306                 2.721769    65.891156        2.729932
std        602.024335                 1.093082    20.329428        0.711561
min          1.000000                 1.000000    30.000000        1.000000
25%        491.250000                 2.000000    48.000000        2.000000
50%       1020.500000                 3.000000    66.000000        3.000000
75%       1555.750000                 4.000000    83.750000        3.000000
max       2068.000000                 4.000000   100.000000        4.000000

          JobLevel  ...  RelationshipSatisfaction  StandardHours  \
count  1470.000000  ...               1470.000000         1470.0
mean      2.063946  ...                  2.712245           80.0
std       1.106940  ...                  1.081209            0.0
min       1.000000  ...                  1.000000           80.0
25%       1.000000  ...                  2.000000           80.0
50%       2.000000  ...                  3.000000           80.0
75%       3.000000  ...                  4.000000           80.0
max       5.000000  ...                  4.000000           80.0

       StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
count       1470.000000        1470.000000            1470.000000
mean           0.793878          11.279592               2.799320
std            0.852077           7.780782               1.289271
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      1470.000000     1470.000000         1470.000000
mean          2.761224        7.008163            4.229252
std           0.706476        6.126525            3.623137
min           1.000000        0.000000            0.000000
25%           2.000000        3.000000            2.000000
50%           3.000000        5.000000            3.000000
75%           3.000000        9.000000            7.000000
max           4.000000       40.000000           18.000000

       YearsSinceLastPromotion  YearsWithCurrManager
count              1470.000000           1470.000000
mean                  2.187755              4.123129
std                   3.222430              3.568136
min                   0.000000              0.000000
25%                   0.000000              2.000000
50%                   1.000000              3.000000
75%                   3.000000              7.000000
max                  15.000000             17.000000

[8 rows x 26 columns]
==================== Dataset Shape ====================
```

```
(1470, 35)
==================== Identifying and Removing null values ====================
        Age  Attrition  BusinessTravel  DailyRate  Department  \
0     False      False           False      False       False
1     False      False           False      False       False
2     False      False           False      False       False
3     False      False           False      False       False
4     False      False           False      False       False
...     ...        ...             ...        ...         ...
1465  False      False           False      False       False
1466  False      False           False      False       False
1467  False      False           False      False       False
1468  False      False           False      False       False
1469  False      False           False      False       False

      DistanceFromHome  Education  EducationField  EmployeeCount  \
0                False      False           False          False
1                False      False           False          False
2                False      False           False          False
3                False      False           False          False
4                False      False           False          False
...                ...        ...             ...            ...
1465             False      False           False          False
1466             False      False           False          False
1467             False      False           False          False
1468             False      False           False          False
1469             False      False           False          False

      EmployeeNumber  ...  RelationshipSatisfaction  StandardHours  \
0              False  ...                     False          False
1              False  ...                     False          False
2              False  ...                     False          False
3              False  ...                     False          False
4              False  ...                     False          False
...              ...  ...                       ...            ...
1465           False  ...                     False          False
1466           False  ...                     False          False
1467           False  ...                     False          False
1468           False  ...                     False          False
1469           False  ...                     False          False

      StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
0                False              False                  False
1                False              False                  False
2                False              False                  False
3                False              False                  False
4                False              False                  False
...                ...                ...                    ...
1465             False              False                  False
1466             False              False                  False
1467             False              False                  False
1468             False              False                  False
1469             False              False                  False
```

```
      WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
0               False           False               False
1               False           False               False
2               False           False               False
3               False           False               False
4               False           False               False
...               ...             ...                 ...
1465            False           False               False
1466            False           False               False
1467            False           False               False
1468            False           False               False
1469            False           False               False

      YearsSinceLastPromotion  YearsWithCurrManager
0                       False                False
1                       False                False
2                       False                False
3                       False                False
4                       False                False
...                       ...                  ...
1465                    False                False
1466                    False                False
1467                    False                False
1468                    False                False
1469                    False                False

[1470 rows x 35 columns]
      Age Attrition     BusinessTravel  DailyRate              Department  \
0      41       Yes      Travel_Rarely       1102                   Sales
1      49        No  Travel_Frequently        279  Research & Development
2      37       Yes      Travel_Rarely       1373  Research & Development
3      33        No  Travel_Frequently       1392  Research & Development
4      27        No      Travel_Rarely        591  Research & Development
...   ...       ...                 ...        ...                     ...
1465   36        No  Travel_Frequently        884  Research & Development
1466   39        No      Travel_Rarely        613  Research & Development
1467   27        No      Travel_Rarely        155  Research & Development
1468   49        No  Travel_Frequently       1023                   Sales
1469   34        No      Travel_Rarely        628  Research & Development

      DistanceFromHome  Education EducationField  EmployeeCount  \
0                    1          2  Life Sciences              1
1                    8          1  Life Sciences              1
2                    2          2          Other              1
3                    3          4  Life Sciences              1
4                    2          1        Medical              1
...                ...        ...            ...            ...
1465                23          2        Medical              1
1466                 6          1        Medical              1
1467                 4          3  Life Sciences              1
1468                 2          3        Medical              1
1469                 8          3        Medical              1

      EmployeeNumber  ...  RelationshipSatisfaction StandardHours  \
```

```
0                  1  ...                         1            80
1                  2  ...                         4            80
2                  4  ...                         2            80
3                  5  ...                         3            80
4                  7  ...                         4            80
...              ...  ...                       ...           ...
1465            2061  ...                         3            80
1466            2062  ...                         1            80
1467            2064  ...                         2            80
1468            2065  ...                         4            80
1469            2068  ...                         1            80

      StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
0                    0                  8                      0
1                    1                 10                      3
2                    0                  7                      3
3                    0                  8                      3
4                    1                  6                      3
...                ...                ...                    ...
1465                 1                 17                      3
1466                 1                  9                      5
1467                 1                  6                      0
1468                 0                 17                      3
1469                 0                  6                      3

     WorkLifeBalance  YearsAtCompany YearsInCurrentRole  \
0                  1               6                  4
1                  3              10                  7
2                  3               0                  0
3                  3               8                  7
4                  3               2                  2
...              ...             ...                ...
1465               3               5                  2
1466               3               7                  7
1467               3               6                  2
1468               2               9                  6
1469               4               4                  3

      YearsSinceLastPromotion  YearsWithCurrManager
0                           0                     5
1                           1                     7
2                           0                     0
3                           3                     0
4                           2                     2
...                       ...                   ...
1465                        0                     3
1466                        1                     7
1467                        0                     3
1468                        0                     8
1469                        1                     2

[1470 rows x 35 columns]
```

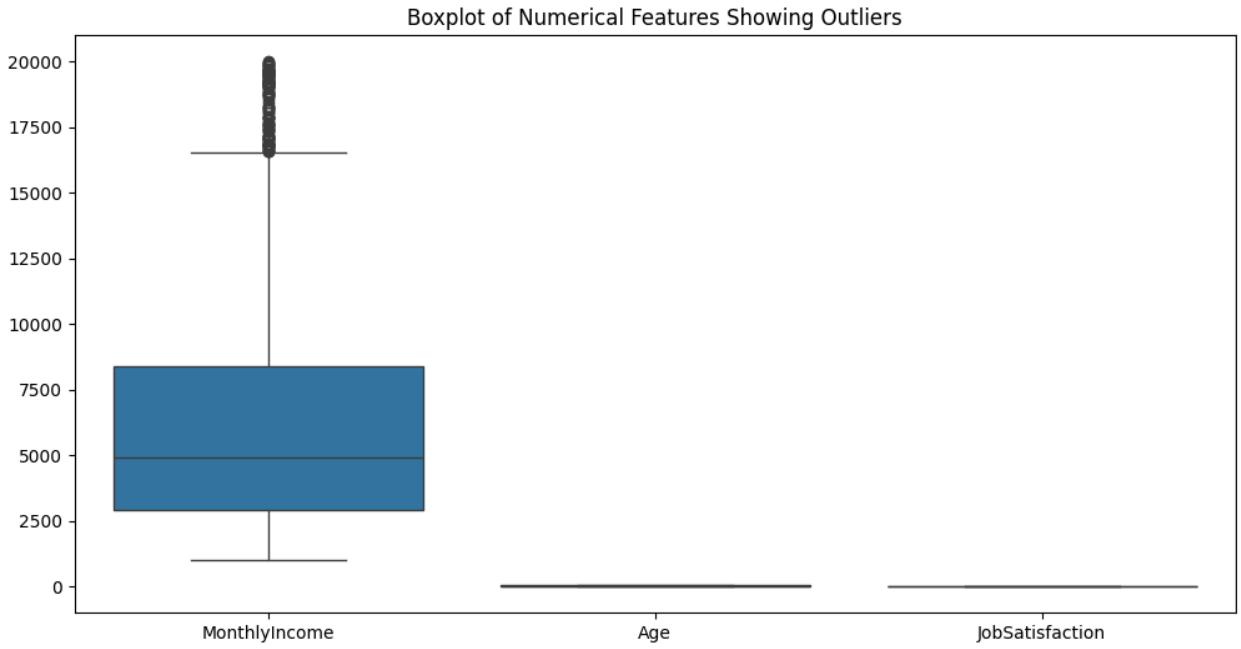
In [23]:
```
df.isna().sum()
```

Out[23]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [24]:
```python
def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]

    return outliers, lower_bound, upper_bound

def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
```

```
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

In [25]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[['MonthlyIncome', 'Age', 'JobSatisfaction']])
plt.title("Boxplot of Numerical Features Showing Outliers")
plt.show()
```



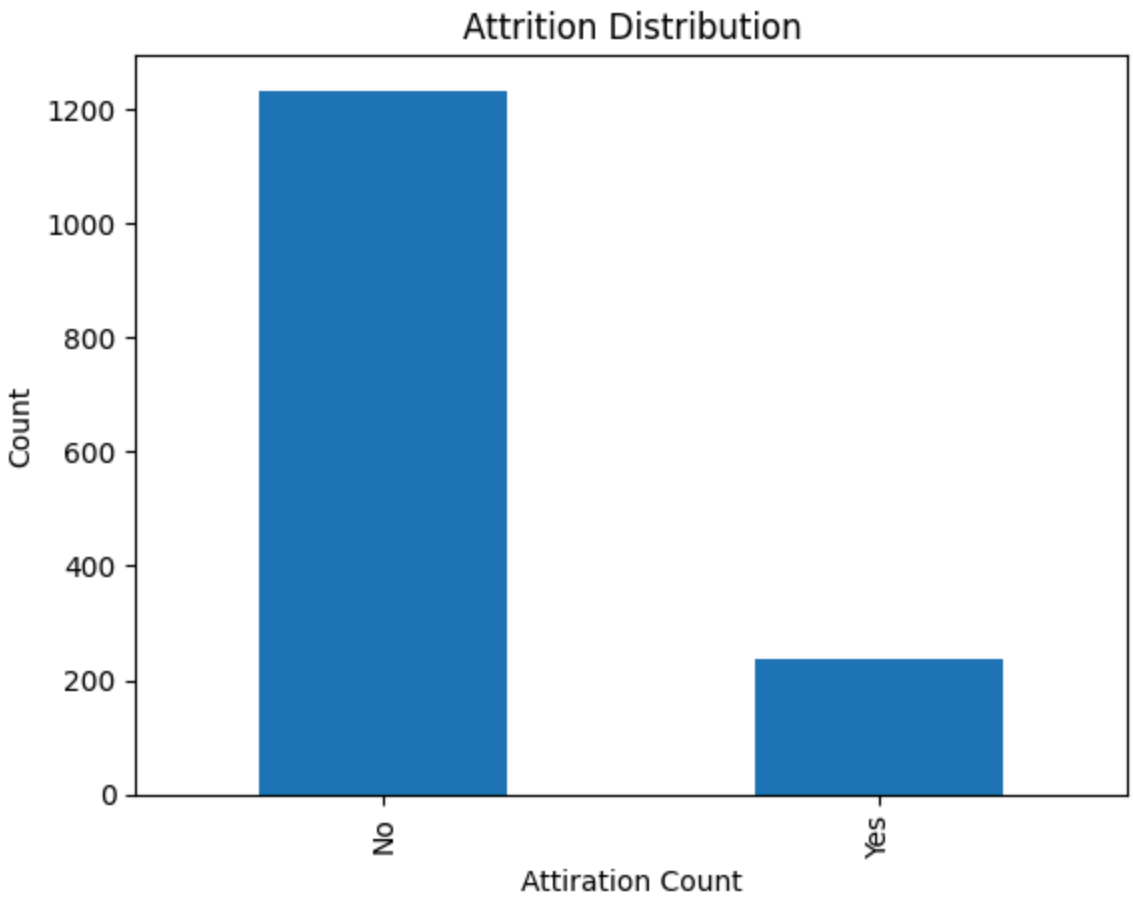
In [26]:
```
from sklearn.preprocessing import LabelEncoder

# Select categorical columns
cat_cols = df.select_dtypes(include="object").columns

# Dictionary to store label encoding mappings
label_encodings = {}

print("======= Label Encoding Table =======")
for col in cat_cols:
    le = LabelEncoder()
    df[col + "_encoded"] = le.fit_transform(df[col])
    mapping = dict(zip(le.classes_, le.transform(le.classes_)))
    label_encodings[col] = mapping

    # Display mapping for this column
    print(f"\nColumn: {col}")
    for key, val in mapping.items():
        print(f"  {key} -> {val}")

# Optionally, convert the dictionary to a DataFrame for a nice table view
import pandas as pd
```

```
all_mappings = []
for col, mapping in label_encodings.items():
    for key, val in mapping.items():
        all_mappings.append([col, key, val])

label_encoding_df = pd.DataFrame(all_mappings, columns=["Feature", "Category",
print("\nFull Label Encoding Table:")
print(label_encoding_df)
```

```
======= Label Encoding Table =======

Column: Attrition
  No -> 0
  Yes -> 1

Column: BusinessTravel
  Non-Travel -> 0
  Travel_Frequently -> 1
  Travel_Rarely -> 2

Column: Department
  Human Resources -> 0
  Research & Development -> 1
  Sales -> 2

Column: EducationField
  Human Resources -> 0
  Life Sciences -> 1
  Marketing -> 2
  Medical -> 3
  Other -> 4
  Technical Degree -> 5

Column: Gender
  Female -> 0
  Male -> 1

Column: JobRole
  Healthcare Representative -> 0
  Human Resources -> 1
  Laboratory Technician -> 2
  Manager -> 3
  Manufacturing Director -> 4
  Research Director -> 5
  Research Scientist -> 6
  Sales Executive -> 7
  Sales Representative -> 8

Column: MaritalStatus
  Divorced -> 0
  Married -> 1
  Single -> 2

Column: Over18
  Y -> 0

Column: OverTime
  No -> 0
  Yes -> 1

Full Label Encoding Table:
           Feature                  Category  Encoded Value
0        Attrition                        No              0
```

1	Attrition	Yes	1
2	BusinessTravel	Non-Travel	0
3	BusinessTravel	Travel_Frequently	1
4	BusinessTravel	Travel_Rarely	2
5	Department	Human Resources	0
6	Department	Research & Development	1
7	Department	Sales	2
8	EducationField	Human Resources	0
9	EducationField	Life Sciences	1
10	EducationField	Marketing	2
11	EducationField	Medical	3
12	EducationField	Other	4
13	EducationField	Technical Degree	5
14	Gender	Female	0
15	Gender	Male	1
16	JobRole	Healthcare Representative	0
17	JobRole	Human Resources	1
18	JobRole	Laboratory Technician	2
19	JobRole	Manager	3
20	JobRole	Manufacturing Director	4
21	JobRole	Research Director	5
22	JobRole	Research Scientist	6
23	JobRole	Sales Executive	7
24	JobRole	Sales Representative	8
25	MaritalStatus	Divorced	0
26	MaritalStatus	Married	1
27	MaritalStatus	Single	2
28	Over18	Y	0
29	OverTime	No	0
30	OverTime	Yes	1

In [27]:
```
plt.Figure()
df["Attrition"].value_counts().plot(kind="bar")
plt.xlabel("Attiration Count")
plt.ylabel("Count")
plt.title("Attrition Distribution")
plt.show()
```



In [28]:
```
type(df)

X = df[["MonthlyIncome", "Age", "JobSatisfaction","Department","Education","Ge
y = df["Attrition"]
```

In [29]:
```
cat_cols=df.select_dtypes(include="object")
num_cols=df.select_dtypes(include="number")

for col in cat_cols:
    print(f"\nCategorical Columns:{col}vs Attrition")
    print(pd.crosstab(df[col],y,margins=True,normalize="index"))
```

Categorical Columns:Attritionvs Attrition

Attrition	No	Yes
Attrition		
No	1.000000	0.000000
Yes	0.000000	1.000000
All	0.838776	0.161224

Categorical Columns:BusinessTravelvs Attrition

Attrition	No	Yes
BusinessTravel		
Non-Travel	0.920000	0.080000
Travel_Frequently	0.750903	0.249097
Travel_Rarely	0.850431	0.149569
All	0.838776	0.161224

Categorical Columns:Departmentvs Attrition

Attrition	No	Yes
Department		
Human Resources	0.809524	0.190476
Research & Development	0.861602	0.138398
Sales	0.793722	0.206278
All	0.838776	0.161224

Categorical Columns:EducationFieldvs Attrition

Attrition	No	Yes
EducationField		
Human Resources	0.740741	0.259259
Life Sciences	0.853135	0.146865
Marketing	0.779874	0.220126
Medical	0.864224	0.135776
Other	0.865854	0.134146
Technical Degree	0.757576	0.242424
All	0.838776	0.161224

Categorical Columns:Gendervs Attrition

Attrition	No	Yes
Gender		
Female	0.852041	0.147959
Male	0.829932	0.170068
All	0.838776	0.161224

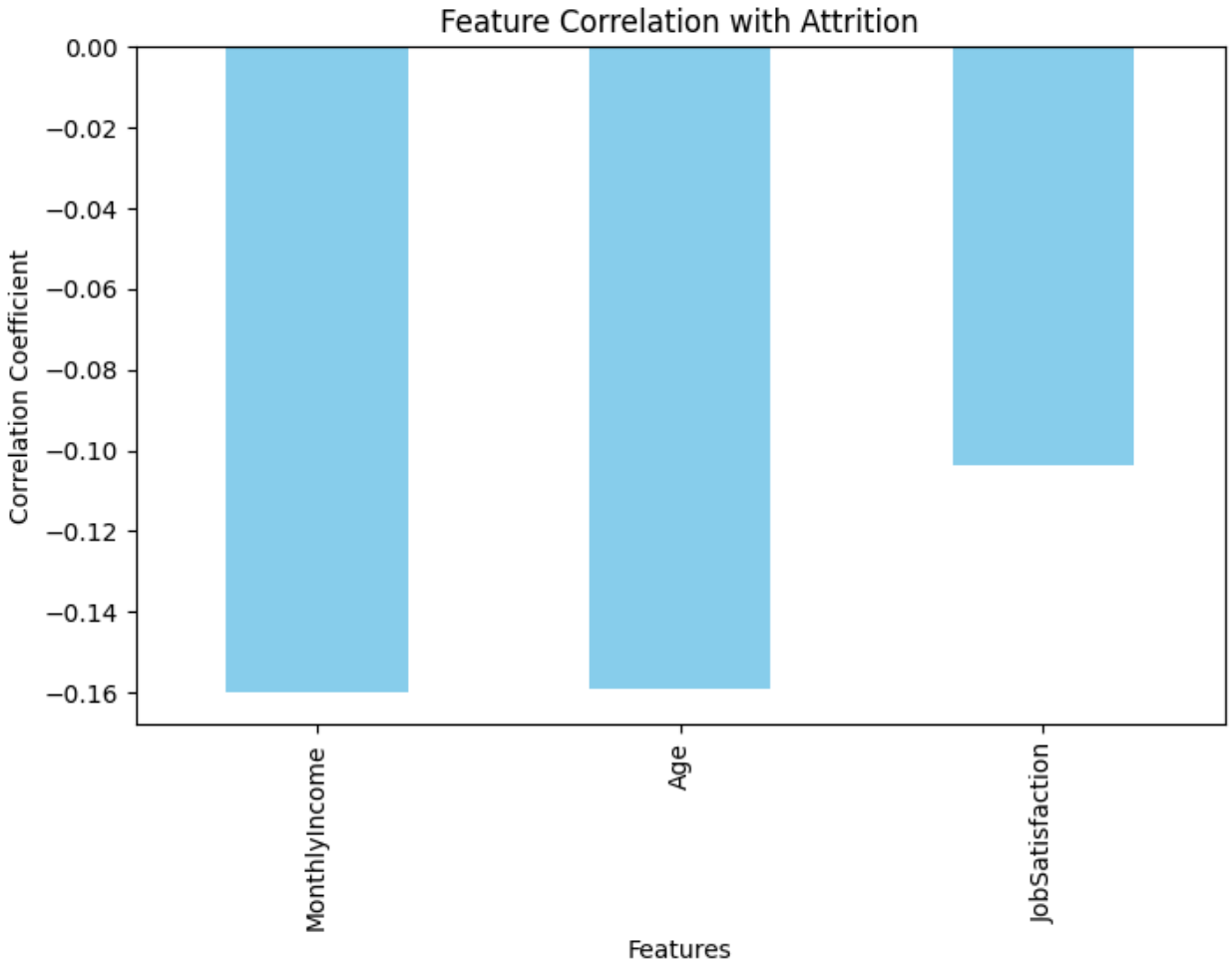
Categorical Columns:JobRolevs Attrition

Attrition	No	Yes
JobRole		
Healthcare Representative	0.931298	0.068702
Human Resources	0.769231	0.230769
Laboratory Technician	0.760618	0.239382
Manager	0.950980	0.049020
Manufacturing Director	0.931034	0.068966
Research Director	0.975000	0.025000
Research Scientist	0.839041	0.160959
Sales Executive	0.825153	0.174847
Sales Representative	0.602410	0.397590
All	0.838776	0.161224

```
Categorical Columns:MaritalStatusvs Attrition
Attrition            No       Yes
MaritalStatus
Divorced       0.899083  0.100917
Married        0.875186  0.124814
Single         0.744681  0.255319
All            0.838776  0.161224

Categorical Columns:Over18vs Attrition
Attrition        No       Yes
Over18
Y          0.838776  0.161224
All        0.838776  0.161224

Categorical Columns:OverTimevs Attrition
Attrition        No       Yes
OverTime
No         0.895636  0.104364
Yes        0.694712  0.305288
All        0.838776  0.161224
```

In [30]:

```
df_corr = df.copy()
df_corr['Attrition_encoded'] = df_corr['Attrition'].map({'No': 0, 'Yes': 1})

# Select only numeric columns for correlation
numeric_cols = ['MonthlyIncome', 'Age', 'JobSatisfaction', 'Attrition_encoded'

correlations = df_corr[numeric_cols].corr()['Attrition_encoded'].drop('Attriti

# Plot as bar chart
plt.figure(figsize=(8,5))
correlations.sort_values().plot(kind='bar', color='skyblue')
plt.title("Feature Correlation with Attrition")
plt.ylabel("Correlation Coefficient")
plt.xlabel("Features")
plt.show()
```



In [31]:
```
X_train, X_test, y_train, y_test = train_test_split(
        X, y, train_size=0.7, random_state=42, stratify=y
)
```

In [32]:
```
# preprocessor = ColumnTransformer(
#     transformers=[
#         ("num", StandardScaler(), num_cols.columns.tolist()),
#         ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols.columns.tol
#     ]
# )

# X_train_processed = preprocessor.fit_transform(X_train)
# X_test_processed = preprocessor.transform(X_test)
# Define the correct columns for your feature set X
numeric_features = ["MonthlyIncome", "Age", "JobSatisfaction", "Education"]
categorical_features = ["Department", "Gender"]

preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_features),
        ("cat", OneHotEncoder(handle_unknown="ignore"), categorical_features)
    ]
)
```

```
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)
```

In [33]:
```
models={
    "Logistic Regression":LogisticRegression(max_iter=1000),
    "Decision Tree":DecisionTreeClassifier(max_depth=10,random_state=42),
    "Random Forest":RandomForestClassifier(n_estimators=100,random_state=42),
    "SVM": SVC(kernel="rbf",probability=True)
}
```

In [34]:
```
# result = []

# print("================Model Prediction===============")

# for name, model in models.items():
#     # Use scaled data for all models since you already scaled X_train and X_
#     model.fit(X_train_scaled, y_train)
#     y_pred = model.predict(X_test_scaled)
#     # Some models (like SVC with probability=True) have predict_proba, other
#     if hasattr(model, "predict_proba"):
#         y_prob = model.predict_proba(X_test_scaled)[:, 1]
#     else:
#         # For models without predict_proba, use decision_function or fallbac
#         if hasattr(model, "decision_function"):
#             y_prob = model.decision_function(X_test_scaled)
#         else:
#             y_prob = None  # or np.zeros_like(y_pred) if you want to avoid e

# result.append([
#         name,
#         accuracy_score(y_test, y_pred),
#         precision_score(y_test, y_pred, pos_label="Yes"),
#         recall_score(y_test, y_pred, pos_label="Yes"),
#         f1_score(y_test, y_pred, pos_label="Yes"),
#         roc_auc_score((y_test == "Yes").astype(int),y_prob)
#     ])

# print(f"{name} completed")

result = []

print("================Model Prediction===============")

for name, model in models.items():
    # Use processed (scaled + encoded) data for all models
    model.fit(X_train_processed, y_train)
    y_pred = model.predict(X_test_processed)
    # Some models (like SVC with probability=True) have predict_proba, others
    if hasattr(model, "predict_proba"):
        y_prob = model.predict_proba(X_test_processed)[:, 1]
    else:
        # For models without predict_proba, use decision_function or fallback
```

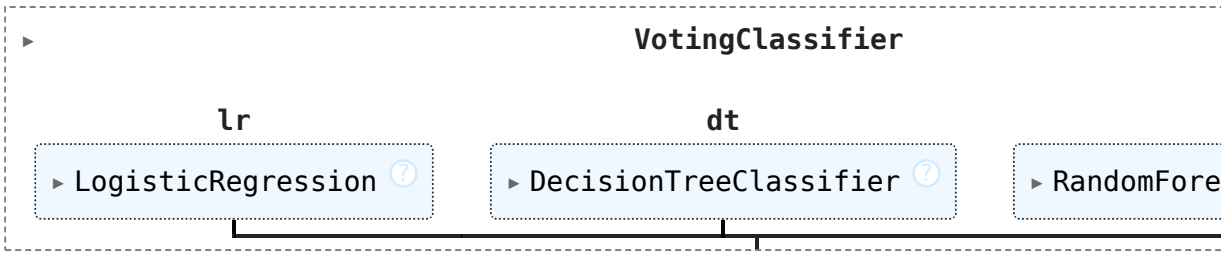
```
        if hasattr(model, "decision_function"):
            y_prob = model.decision_function(X_test_processed)
        else:
            y_prob = None  # or np.zeros_like(y_pred) if you want to avoid err

    result.append([
        name,
        accuracy_score(y_test, y_pred),
        precision_score(y_test, y_pred, pos_label="Yes"),
        recall_score(y_test, y_pred, pos_label="Yes"),
        f1_score(y_test, y_pred, pos_label="Yes"),
        roc_auc_score((y_test == "Yes").astype(int), y_prob)
    ])

    print(f"{name} completed")
```

```
================Model Prediction================
Logistic Regression completed
Decision Tree completed
Random Forest completed
SVM completed
```

```
c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklear
n\metrics\_classification.py:1731: UndefinedMetricWarning: Precision is ill-def
ined and being set to 0.0 due to no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```

In [35]:
```
final_model = VotingClassifier(
    estimators=[
        ("lr", LogisticRegression(max_iter=1000)),
        ("dt", DecisionTreeClassifier(max_depth=10, random_state=42)),
        ("rf", RandomForestClassifier(n_estimators=100, random_state=42))
    ],
    voting="soft"
)

final_model.fit(X_train_processed, y_train)
```

Out[35]:



In [36]:
```
results_df=pd.DataFrame(
    result,
    columns=["Model","Accuracy","Precision","Recall","F1 Score","ROC-AUC"]
)
print("====================Final Prediction====================")
print(results_df)
```

```
====================Final Prediction====================
                 Model  Accuracy  Precision    Recall  F1 Score   ROC-AUC
0  Logistic Regression  0.839002   0.500000  0.014085  0.027397  0.666997
1        Decision Tree  0.755102   0.239437  0.239437  0.239437  0.549848
2        Random Forest  0.807256   0.250000  0.098592  0.141414  0.621545
3                  SVM  0.839002   0.000000  0.000000  0.000000  0.601865
```

In [ ]:

```
# # def GetDataForPrediction():
# #     print("\nInputs")

# #     Monthly_Income = float(input("Enter your monthly income:"))
# #     data = pd.DataFrame(
# #         [[Monthly_Income]],
# #         columns=["MonthlyIncome"]
# #     )
# #     return data

# # new_employee = GetDataForPrediction()

# # new_employee_scaled=scaler.fit_transform(new_employee)

# # attrition_pred=final_model.predict(new_employee_scaled)[0]

# # attrition_prob=final_model.predict_proba(new_employee_scaled)[0][1]

# # print("=====Attrition Prediction=======")

# # print("Will Employee Leave?",attrition_pred)

# # print(f"Probability of Leaving:{attrition_prob:.2f}")

# def get_int(prompt):
#     while True:
#         value = input(prompt)
#         if value.strip().isdigit():
#             return int(value)
#         print("❌ Please enter a valid integer.")

# def get_float(prompt):
#     while True:
#         value = input(prompt)
#         try:
#             return float(value)
#         except ValueError:
#             print("❌ Please enter a valid number.")

# def get_int(prompt):
#     while True:
#         value = input(prompt).strip()
#         if value.isdigit():
#             return int(value)
#         print("❌ Please enter a valid integer.")

# def get_float(prompt):
```

```
#     while True:
#         value = input(prompt).strip()
#         try:
#             return float(value)
#         except ValueError:
#             print("❌ Please enter a valid number.")

# def GetDataForPrediction():
#     data = {
#         "MonthlyIncome": [float(input("Monthly Income: "))],
#         "Age": [int(input("Age: "))],
#         "JobSatisfaction": [int(input("Job Satisfaction (1-4): "))],
#         "Department": [input("Department: ")],
#         "Education": [input("Education: ")],
#         "Gender": [input("Gender: ")]
#     }
#     return pd.DataFrame(data)

# import numpy as np

# employee = GetDataForPrediction()
# employee.replace("", np.nan, inplace=True)

# # Convert numeric columns to correct type
# employee["MonthlyIncome"] = pd.to_numeric(employee["MonthlyIncome"], errors=
# employee["Age"] = pd.to_numeric(employee["Age"], errors="coerce")
# employee["JobSatisfaction"] = pd.to_numeric(employee["JobSatisfaction"], err

# # Ensure all expected columns exist
# expected_cols = [
#     "MonthlyIncome",
#     "Age",
#     "JobSatisfaction",
#     "Department",
#     "Education",
#     "Gender"
# ]
# for col in expected_cols:
#     if col not in employee.columns:
#         employee[col] = np.nan

# employee = employee[expected_cols]

# # Fill missing values with a default or most frequent value from training da
# for col in employee.columns:
#     if employee[col].isnull().any():
#         if col in X_train.columns:
#             # Use mode for categorical, median for numeric
#             if X_train[col].dtype == "O":
#                 employee[col].fillna(X_train[col].mode()[0], inplace=True)
#             else:
#                 employee[col].fillna(X_train[col].median(), inplace=True)
#         else:
```

```
#             employee[col].fillna(0, inplace=True)

# # Encode categorical columns using label_encodings
# for col in ["Department", "Education", "Gender"]:
#     if col in employee.columns:
#         mapping = label_encodings.get(col, {})
#         employee[col] = employee[col].map(mapping)
#         # If mapping fails (unknown value), fill with mode from X_train
#         if employee[col].isnull().any() and col in X_train.columns:
#             employee[col].fillna(X_train[col].mode()[0], inplace=True)

# new_employee_preprocessed = preprocessor.transform(employee)

# attrition_prediction = final_model.predict(new_employee_preprocessed)[0]
# attrition_prediction_probability = final_model.predict_proba(new_employee_pr
# print("Will Employee Leave?", attrition_prediction)
# print("What is the probability", attrition_prediction_probability)
# def GetDataForPrediction():
#     print("\nInputs")

#     Monthly_Income = float(input("Enter your monthly income:"))
#     data = pd.DataFrame(
#         [[Monthly_Income]],
#         columns=["MonthlyIncome"]
#     )
#     return data

# new_employee = GetDataForPrediction()

# new_employee_scaled=scaler.fit_transform(new_employee)

# attrition_pred=final_model.predict(new_employee_scaled)[0]

# attrition_prob=final_model.predict_proba(new_employee_scaled)[0][1]

# print("=====Attrition Prediction=======")

# print("Will Employee Leave?",attrition_pred)

# print(f"Probability of Leaving:{attrition_prob:.2f}")









# def get_int(prompt):
#     while True:
#         value = input(prompt)
#         if value.strip().isdigit():
#             return int(value)
```

```
#         print("❌ Please enter a valid integer.")

# def get_float(prompt):
#     while True:
#         value = input(prompt)
#         try:
#             return float(value)
#         except ValueError:
#             print("❌ Please enter a valid number.")

# def get_int(prompt):
#     while True:
#         value = input(prompt).strip()
#         if value.isdigit():
#             return int(value)
#         print("❌ Please enter a valid integer.")

# def get_float(prompt):
#     while True:
#         value = input(prompt).strip()
#         try:
#             return float(value)
#         except ValueError:
#             print("❌ Please enter a valid number.")

# def GetDataForPrediction():
#     data = {
# "MonthlyIncome": [get_float("Monthly Income: ")],
# "Age": [get_int("Age: ")],
#         "JobSatisfaction": [get_int("Job Satisfaction (1-4): ")],
#         "Department": [input("Department: ").strip()],
#         "Education": [input("Education: ").strip()],
#         "Gender": [input("Gender: ").strip()]
#     }
#     return pd.DataFrame(data)

# import numpy as np

# employee = GetDataForPrediction()
# employee.replace("", np.nan, inplace=True)

# # Convert numeric columns to correct type
# employee["MonthlyIncome"] = pd.to_numeric(employee["MonthlyIncome"], errors=
# employee["Age"] = pd.to_numeric(employee["Age"], errors="coerce")
# employee["JobSatisfaction"] = pd.to_numeric(employee["JobSatisfaction"], err

# # Ensure all expected columns exist
# expected_cols = [
#     "MonthlyIncome",
#     "Age",
#     "JobSatisfaction",
#     "Department",
#     "Education",
```

```
#     "Gender"
# ]
# for col in expected_cols:
#     if col not in employee.columns:
#         employee[col] = np.nan

# employee = employee[expected_cols]

# # Fill missing values with a default or most frequent value from training da
# for col in employee.columns:
#     if employee[col].isnull().any():
#         if col in X_train.columns:
#             # Use mode for categorical, median for numeric
#             if X_train[col].dtype == "O":
#                 employee[col].fillna(X_train[col].mode()[0], inplace=True)
#             else:
#                 employee[col].fillna(X_train[col].median(), inplace=True)
#         else:
#             employee[col].fillna(0, inplace=True)

# # Encode categorical columns using label_encodings
# for col in ["Department", "Education", "Gender"]:
#     if col in employee.columns:
#         mapping = label_encodings.get(col, {})
#         employee[col] = employee[col].map(mapping)
#         # Convert to numeric to avoid isnan errors
#         employee[col] = pd.to_numeric(employee[col], errors='coerce')
#         # If mapping fails (unknown value), fill with mode from X_train
#         if employee[col].isnull().any() and col in X_train.columns:
#             # Use mode from X_train (already encoded)
#             employee[col].fillna(X_train[col].mode()[0], inplace=True)

# new_employee_preprocessed = preprocessor.transform(employee)

# attrition_prediction = final_model.predict(new_employee_preprocessed)[0]
# attrition_prediction_probability = final_model.predict_proba(new_employee_pr
# print("Will Employee Leave?", attrition_prediction)
# print("What is the probability", attrition_prediction_probability)
import pandas as pd
import numpy as np

# ------------------------------
# Input helpers
# ------------------------------
def get_int(prompt):
    while True:
        value = input(prompt).strip()
        if value.isdigit():
            return int(value)
        print("❌ Please enter a valid integer.")

def get_float(prompt):
    while True:
```

```
        value = input(prompt).strip()
        try:
            return float(value)
        except ValueError:
            print("❌ Please enter a valid number.")

# ------------------------------
# Collect user input (RAW DATA)
# ------------------------------
def GetDataForPrediction():
    data = {
        "MonthlyIncome": [get_float("Monthly Income: ")],
        "Age": [get_int("Age: ")],
        "JobSatisfaction": [get_int("Job Satisfaction (1-4): ")],
        "Department": [input("Department: ").strip()],
        "Education": [input("Education: ").strip()],
        "Gender": [input("Gender: ").strip()]
    }
    return pd.DataFrame(data)

# ------------------------------
# Prediction pipeline
# ------------------------------
employee = GetDataForPrediction()

# Replace empty strings with None for categorical columns
for col in ["Department", "Education", "Gender"]:
    if col in employee.columns:
        employee[col] = employee[col].replace("", None).astype("object")

# Fill missing values for categorical columns with mode from X_train
for col in ["Department", "Education", "Gender"]:
    if col in employee.columns and employee[col].isnull().any():
        if col in X_train.columns:
            mode_val = X_train[col].mode()[0]
            employee[col] = employee[col].fillna(mode_val)
        else:
            employee[col] = employee[col].fillna("Unknown")

# Fill missing values for numeric columns with median from X_train
for col in ["MonthlyIncome", "Age", "JobSatisfaction"]:
    if col in employee.columns and employee[col].isnull().any():
        if col in X_train.columns:
            median_val = X_train[col].median()
            employee[col] = employee[col].fillna(median_val)
        else:
            employee[col] = employee[col].fillna(0)

# Ensure correct dtypes
employee["MonthlyIncome"] = pd.to_numeric(employee["MonthlyIncome"], errors="c
employee["Age"] = pd.to_numeric(employee["Age"], errors="coerce")
employee["JobSatisfaction"] = pd.to_numeric(employee["JobSatisfaction"], error
for col in ["Department", "Education", "Gender"]:
```

```
    employee[col] = employee[col].astype("object")

# Apply the SAME preprocessor used during training
employee_processed = preprocessor.transform(employee)

# Predict
attrition_pred = final_model.predict(employee_processed)[0]
attrition_prob = final_model.predict_proba(employee_processed)[0][1]

# Output
print("\n===== ATTRITION PREDICTION =====")
print("Will Employee Leave?:", "Yes" if attrition_pred == 1 else "No")
print(f"Probability of Leaving: {attrition_prob:.2f}")
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_32344\2515441770.py:286: FutureWarni
ng: Downcasting behavior in `replace` is deprecated and will be removed in a fu
ture version. To retain the old behavior, explicitly call `result.infer_object
s(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.n
o_silent_downcasting', True)`
  employee = employee.replace("", np.nan)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[39], line 292
    286 employee = employee.replace("", np.nan)
    288 # 🚨 DO NOT manually encode / scale / fill here
    289 # Let the trained preprocessor handle everything
    290
    291 # Apply the SAME preprocessor used during training
--> 292 employee_processed = preprocessor.transform(employee)
    294 # Predict
    295 attrition_pred = final_model.predict(employee_processed)[0]

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\utils\_set_output.py:316, in _wrap_method_output.<locals>.wrapped(self,
X, *args, **kwargs)
    314 @wraps(f)
    315 def wrapped(self, X, *args, **kwargs):
--> 316     data_to_wrap = f(self, X, *args, **kwargs)
    317     if isinstance(data_to_wrap, tuple):
    318         # only wrap the first output for cross decomposition
    319         return_tuple = (
    320             _wrap_data_with_container(method, data_to_wrap[0], X, sel
f),
    321             *data_to_wrap[1:],
    322         )

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\compose\_column_transformer.py:1096, in ColumnTransformer.transform(self,
X, **params)
   1093 else:
   1094     routed_params = self._get_empty_routing()
-> 1096 Xs = self._call_func_on_transformers(
   1097     X,
   1098     None,
   1099     _transform_one,
   1100     column_as_labels=fit_dataframe_and_transform_dataframe,
   1101     routed_params=routed_params,
   1102 )
   1103 self._validate_output(Xs)
   1105 if not Xs:
   1106     # All transformers are None

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\compose\_column_transformer.py:897, in ColumnTransformer._call_func_on_tr
ansformers(self, X, y, func, column_as_labels, routed_params)
    885             extra_args = {}
    886         jobs.append(
    887             delayed(func)(
    888                 transformer=clone(trans) if not fitted else trans,
   (...)
    894             )
    895         )
--> 897     return Parallel(n_jobs=self.n_jobs)(jobs)
    899 except ValueError as e:
```

```
    900     if "Expected 2D array, got 1D array instead" in str(e):

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\utils\parallel.py:82, in Parallel.__call__(self, iterable)
     73 warning_filters = warnings.filters
     74 iterable_with_config_and_warning_filters = (
     75     (
     76         _with_config_and_warning_filters(delayed_func, config, warnin
g_filters),
   (...)
     80     for delayed_func, args, kwargs in iterable
     81 )
---> 82 return super().__call__(iterable_with_config_and_warning_filters)

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\jo
blib\parallel.py:1986, in Parallel.__call__(self, iterable)
   1984     output = self._get_sequential_output(iterable)
   1985     next(output)
-> 1986     return output if self.return_generator else list(output)
   1988 # Let's create an ID that uniquely identifies the current call. If the
   1989 # call is interrupted early and that the same instance is immediately
   1990 # reused, this id will be used to prevent workers that were
   1991 # concurrently finalizing a task from the previous call to run the
   1992 # callback.
   1993 with self._lock:

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\jo
blib\parallel.py:1914, in Parallel._get_sequential_output(self, iterable)
   1912 self.n_dispatched_batches += 1
   1913 self.n_dispatched_tasks += 1
-> 1914 res = func(*args, **kwargs)
   1915 self.n_completed_tasks += 1
   1916 self.print_progress()

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\utils\parallel.py:147, in _FuncWrapper.__call__(self, *args, **kwargs)
    145 with config_context(**config), warnings.catch_warnings():
    146     warnings.filters = warning_filters
--> 147     return self.function(*args, **kwargs)

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\pipeline.py:1520, in _transform_one(transformer, X, y, weight, params)
   1498 def _transform_one(transformer, X, y, weight, params):
   1499     """Call transform and apply weight to output.
   1500
   1501     Parameters
   (...)
   1518         This should be of the form ``process_routing()["step_name"]``.
   1519     """
-> 1520     res = transformer.transform(X, **params.transform)
   1521     # if we have a weight for this transformer, multiply output
   1522     if weight is None:

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
```

```
learn\utils\_set_output.py:316, in _wrap_method_output.<locals>.wrapped(self,
X, *args, **kwargs)
    314 @wraps(f)
    315 def wrapped(self, X, *args, **kwargs):
--> 316     data_to_wrap = f(self, X, *args, **kwargs)
    317     if isinstance(data_to_wrap, tuple):
    318         # only wrap the first output for cross decomposition
    319         return_tuple = (
    320             _wrap_data_with_container(method, data_to_wrap[0], X, sel
f),
    321             *data_to_wrap[1:],
    322         )

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\preprocessing\_encoders.py:1043, in OneHotEncoder.transform(self, X)
   1038     warn_on_unknown = self.drop is not None and self.handle_unknown in
{
   1039         "ignore",
   1040         "infrequent_if_exist",
   1041     }
   1042     handle_unknown = self.handle_unknown
-> 1043 X_int, X_mask = self._transform(
   1044     X,
   1045     handle_unknown=handle_unknown,
   1046     ensure_all_finite="allow-nan",
   1047     warn_on_unknown=warn_on_unknown,
   1048 )
   1050 n_samples, n_features = X_int.shape
   1052 if self._drop_idx_after_grouping is not None:

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\preprocessing\_encoders.py:210, in _BaseEncoder._transform(self, X, handl
e_unknown, ensure_all_finite, warn_on_unknown, ignore_category_indices)
    208 for i in range(n_features):
    209     Xi = X_list[i]
--> 210     diff, valid_mask = _check_unknown(Xi, self.categories_[i], return_m
ask=True)
    212     if not np.all(valid_mask):
    213         if handle_unknown == "error":

File c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\utils\_encode.py:313, in _check_unknown(values, known_values, return_mas
k)
    310         valid_mask = xp.ones(len(values), dtype=xp.bool)
    312 # check for nans in the known_values
--> 313 if xp.any(xp.isnan(known_values)):
    314     diff_is_nan = xp.isnan(diff)
    315     if xp.any(diff_is_nan):
    316         # removes nan from valid_mask

TypeError: ufunc 'isnan' not supported for the input types, and the inputs coul
d not be safely coerced to any supported types according to the casting rule
''safe''
```

In [ ]:
```
print("\n=== RAW EMPLOYEE DATA ===")
print(employee)
print("\n=== DTYPES ===")
print(employee.dtypes)
```

```
=== RAW EMPLOYEE DATA ===
   MonthlyIncome  Age  JobSatisfaction              Department  Education  \
0          221.0   21               21  Research & Development        3.0

  Gender
0   Male

=== DTYPES ===
MonthlyIncome      float64
Age                  int64
JobSatisfaction      int64
Department          object
Education          float64
Gender              object
dtype: object
```

In [ ]:
```
from joblib import dump
dump(final_model, "Attrition_detection_model.pkl")

# dump(Scaler, "Scaler.pkl")

dump(X.columns.tolist(), "feature_names.pkl")

print("\nModel, Scaler and Feature name Saved Successfully!")
```

```
Model, Scaler and Feature name Saved Successfully!
```

In [ ]:
```
# %pip install streamlit
# import streamlit as st
# import pandas as pd
# import joblib

# # Load trained model and scaler
# model = joblib.load("Attrition_detection_model.pkl")
# scaler = joblib.load("scaler.pkl")

# st.set_page_config(page_title="Employee Attrition Predictor", layout="center

# st.title("📉 Employee Attrition Prediction")
# st.write("Predict whether an employee is likely to leave the company.")

# # User Input
# monthly_income = st.number_input(
#     "Monthly Income",
#     min_value=0.0,
#     step=500.0
# )

# if st.button("Predict Attrition"):
```

```
#     input_df = pd.DataFrame(
#         [[monthly_income]],
#         columns=["MonthlyIncome"]
#     )

#     input_scaled = scaler.transform(input_df)

#     prediction = model.predict(input_scaled)[0]
#     probability = model.predict_proba(input_scaled)[0][1]

#     if prediction == "Yes":
#         st.error(f"⚠ Employee likely to leave (Risk: {probability:.2%})")
#     else:
#         st.success(f"✅ Employee likely to stay (Risk: {probability:.2%})")

# %pip install streamlit

# import streamlit as st
# import pandas as pd
# import joblib
# import numpy as np

# # Load full pipeline (preprocessor + model)
# pipeline = joblib.load("Attrition_detection_model.pkl")

# st.set_page_config(
#     page_title="Employee Attrition Predictor",
#     layout="centered"
# )

# st.title("📉 Employee Attrition Prediction")
# st.write("Predict whether an employee is likely to leave the company.")

# # ========================
# # USER INPUTS
# # ========================

# monthly_income = st.number_input(
#     "Monthly Income",
#     min_value=0.0,
#     step=500.0
# )

# age = st.number_input(
#     "Age",
#     min_value=18,
#     max_value=65,
#     step=1
# )

# job_satisfaction = st.selectbox(
#     "Job Satisfaction (1 = Low, 4 = High)",
#     [1, 2, 3, 4]
```

```
# )

# department = st.selectbox(
#     "Department",
#     ["Sales", "Research & Development", "Human Resources"]
# )

# education = st.selectbox(
#     "Education Level",
#     ["Below College", "College", "Bachelor", "Master", "Doctor"]
# )

# gender = st.selectbox(
#     "Gender",
#     ["Male", "Female"]
# )

# # ======================
# # PREDICTION
# # ======================

# if st.button("Predict Attrition"):

#     input_df = pd.DataFrame({
#         "MonthlyIncome": [monthly_income],
#         "Age": [age],
#         "JobSatisfaction": [job_satisfaction],
#         "Department": [department],
#         "Education": [education],
#         "Gender": [gender]
#     })

#     # safety: handle empty strings
#     input_df.replace("", np.nan, inplace=True)

#     prediction = pipeline.predict(input_df)[0]
#     probability = pipeline.predict_proba(input_df)[0][1]

#     st.subheader("Result")

#     if prediction == "Yes":
#         st.error(f"⚠ Employee likely to leave\n\nRisk: **{probability:.2%}**
#     else:
#         st.success(f"✅ Employee likely to stay\n\nRisk: **{probability:.2%}
import streamlit as st
import pandas as pd
import joblib
import numpy as np

# Load full trained pipeline (preprocessor + model)
pipeline = joblib.load("Attrition_detection_model.pkl")

st.set_page_config(
```

```
    page_title="Employee Attrition Detection System",
    layout="centered"
)

st.title("📉 Employee Attrition Detection System")
st.write("Predict whether an employee is likely to leave the company.")

st.subheader("Enter Employee Details")

monthly_income = st.number_input(
    "Monthly Income",
    min_value=0.0,
    step=500.0
)

age = st.number_input(
    "Age",
    min_value=18,
    max_value=65,
    step=1
)

job_satisfaction = st.selectbox(
    "Job Satisfaction (1 = Low, 4 = High)",
    [1, 2, 3, 4]
)

department = st.selectbox(
    "Department",
    ["Sales", "Research & Development", "Human Resources"]
)

education = st.selectbox(
    "Education Level",
    ["Below College", "College", "Bachelor", "Master", "Doctor"]
)

gender = st.selectbox(
    "Gender",
    ["Male", "Female"]
)

if st.button("Predict Attrition"):

    input_data = pd.DataFrame({
        "MonthlyIncome": [monthly_income],
        "Age": [age],
        "JobSatisfaction": [job_satisfaction],
        "Department": [department],
        "Education": [education],
        "Gender": [gender]
    })
```

```
    input_data.replace("", np.nan, inplace=True)

    prediction = pipeline.predict(input_data)[0]
    probability = pipeline.predict_proba(input_data)[0][1]

    st.subheader("Prediction Result")

    if prediction == "Yes":
        st.error(f"⚠ Employee likely to leave\n\nRisk Probability: **{probabil
    else:
        st.success(f"✅ Employee likely to stay\n\nRisk Probability: **{probab
```

```
2026-01-20 12:44:03.641 WARNING streamlit.runtime.scriptrunner_utils.script_ru
n_context: Thread 'MainThread': missing ScriptRunContext! This warning can be i
gnored when running in bare mode.
2026-01-20 12:44:03.641 WARNING streamlit.runtime.scriptrunner_utils.script_ru
n_context: Thread 'MainThread': missing ScriptRunContext! This warning can be i
gnored when running in bare mode.
2026-01-20 12:44:04.656
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run c:\Users\DELL\AppData\Local\Programs\Python\Python312\Lib\sit
e-packages\ipykernel_launcher.py [ARGUMENTS]
2026-01-20 12:44:04.656 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.656 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.656 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.656 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.656 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.664 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Session state does not function when running a script w
ithout `streamlit run`
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.665 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
```

```
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.672 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.680 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.681 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.681 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.681 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.681 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.681 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.681 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.681 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.681 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.688 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.688 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.689 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
```

```
2026-01-20 12:44:04.690 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.696 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.697 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.698 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.698 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.698 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.698 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.698 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
2026-01-20 12:44:04.698 Thread 'MainThread': missing ScriptRunContext! This war
ning can be ignored when running in bare mode.
```

In [ ]: