Asavari Niteen Thoke (21BCE7851)

```
df=pd.read_csv("HR-Employee-Attrition.csv")
```

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4
4	27	No	Travel_Rarely	591	Research & Development	2	1

5 rows × 35 columns

```
from sklearn import set_config
set_config(display = "diagram")
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
pd.set_option("display.max_columns",None)
```

```
df.shape
```

```
(1470, 35)
```

```
df.isnull().sum()
```

```
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
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
df.describe()
```

```
numeric_feature = df.select_dtypes(exclude='object').columns
catagorical_feature = df.select_dtypes(include='object').columns
```

mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.86

```
uni_cat = []
no_cat = []
null_vals = []
for cat_feature in catagorical_feature:
  uni_cat.append(df[cat_feature].unique())
  no_cat.append(df[cat_feature].nunique())
  null_vals.append(df[cat_feature].isnull().sum())

df_info = pd.DataFrame({
                      "FEATURE NAME" : catagorical_feature,
                      "NO UNI CAT" : no_cat,
                      "NULL VALS" : null_vals,
                      "UNIQUE CAT": uni_cat,
                       })
```

```
df_info
```

	FEATURE NAME	NO UNI CAT	NULL VALS	UNIQUE CAT
0	Attrition	2	0	[Yes, No]
1	BusinessTravel	3	0	[Travel_Rarely, Travel_Frequently, Non-Travel]
2	Department	3	0	[Sales, Research & Development, Human Resources]
3	EducationField	6	0	[Life Sciences, Other, Medical, Marketing, Tec...
4	Gender	2	0	[Female, Male]
5	JobRole	9	0	[Sales Executive, Research Scientist, Laborato...
6	MaritalStatus	3	0	[Single, Married, Divorced]
7	Over18	1	0	[Y]

```
df = df.drop(['EmployeeNumber',"EmployeeCount","Over18","Attrition","DistanceFromHome"],axis=1)
```

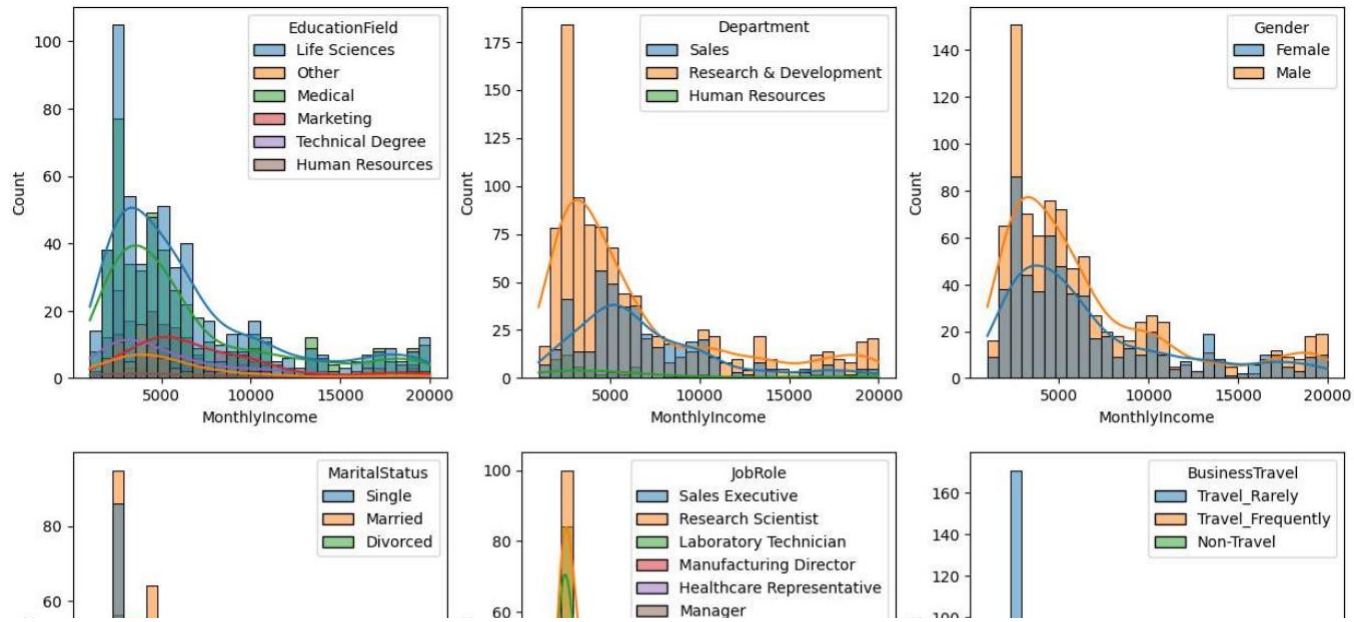
```
df.head()
```

	Age	BusinessTravel	DailyRate	Department	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate	JobInvolvement
0	41	Travel_Rarely	1102	Sales	2	Life Sciences	2	Female	94	3
1	49	Travel_Frequently	279	Research & Development	1	Life Sciences	3	Male	61	2
2	37	Travel_Rarely	1373	Research & Development	2	Other	4	Male	92	2
3	33	Travel_Frequently	1392	Research & Development	4	Life Sciences	4	Female	56	3
4	27	Travel_Rarely	591	Research & Development	1	Medical	1	Male	40	3

```
plt.subplots(3,3,figsize=(15,15))

plt.subplot(331)
sns.histplot(data  = df,
             x     = "MonthlyIncome",
             bins  = 30,
             kde   = True,
             hue   = "EducationField")

plt.subplot(332)
sns.histplot(data  = df,
             x     = "MonthlyIncome",
             bins  = 30,
             kde   = True,
```

```
                hue  = "Department")

    plt.subplot(333)
    sns.histplot(data = df,
                 x    = "MonthlyIncome",
                 bins = 30,
                 kde  = True,
                 hue  = "Gender")

    plt.subplot(334)
    sns.histplot(data = df,
                 x    = "MonthlyIncome",
                 bins = 30,
                 kde  = True,
                 hue  = "MaritalStatus")

    plt.subplot(335)
    sns.histplot(data = df,
                 x    = 'MonthlyIncome',
                 bins = 30,
                 kde  = True,
                 hue  = 'JobRole')

    plt.subplot(336)
    sns.histplot(data = df,
                 x    = "MonthlyIncome",
                 bins = 30,
                 kde  = True,
                 hue  = "BusinessTravel")

    plt.subplot(337)
    sns.histplot(data = df,
                 x    = "MonthlyIncome",
                 bins = 30,
                 kde  = True,
                 hue  = "OverTime")
```

```
<Axes: xlabel='MonthlyIncome', ylabel='Count'>
```

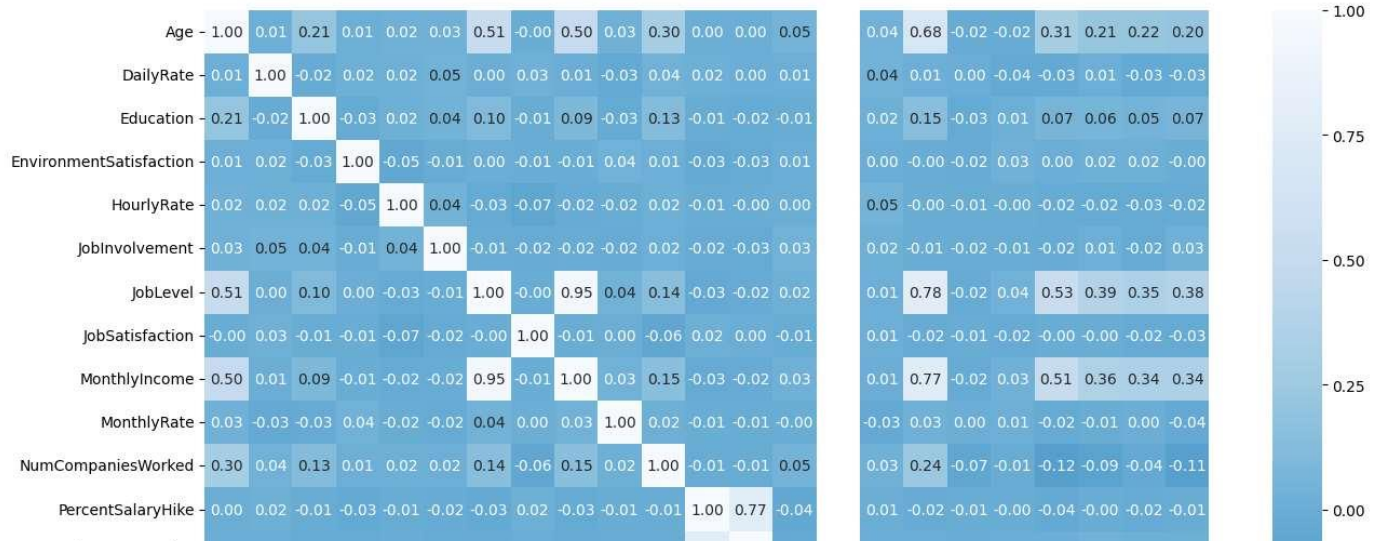



```
group_data = df.groupby("EducationField")
group_data.get_group("Human Resources")["MonthlyIncome"].unique()
```

```
array([ 2073, 18844, 17328,  6347,  6410,  9950, 19141, 19189,  6389,
        2143,  3737,  2277,  2844,  3600, 16799,  6077,  5743,  1555,
        6430,  2180,  2592,  2956,  2335,  3886,  2863, 19636,  2187])
```

```
plt.figure(figsize=(15,12))
sns.heatmap(df.corr(),annot=True,fmt=".2f",vmin=-1,vmax=1,cmap=plt.cm.Blues_r)
plt.show()
```

```
<ipython-input-18-062427f61837>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  sns.heatmap(df.corr(),annot=True,fmt=".2f",vmin=-1,vmax=1,cmap=plt.cm.Blues_r)
```




```
def correlation(dataset,threshold):
  col_corr = set()
  corr_matrix = dataset.corr()
  for i in range(len(corr_matrix.columns)):
    for j in range(i):
      # if abs(corr_matrix.iloc[i,j]) > threshold:  # this will remove the highly correlated feature including +ve and -ve
      if abs(corr_matrix.iloc[i,j]) > threshold:    # this will only remove the highly +ve correlated features
        colname = corr_matrix.column[i]   # gets the column name
        col_corr.add(colname)
    return col_corr
```

```
corr_features = correlation(df,0.85)
corr_features
```

```
<ipython-input-19-ec6c49401ebd>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  corr_matrix = dataset.corr()
set()
```

```
set()
```

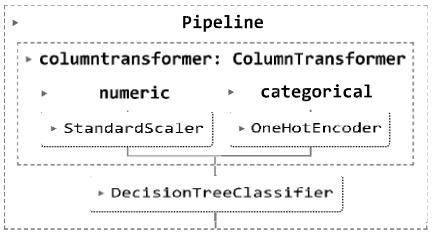
```
set()
```

DECISION TREE

```
y = df.iloc[:,11]
# the independant features will not include the 14th column
x = df.iloc[:,list(range(11)) + list(range(12,len(df.columns)))]
```

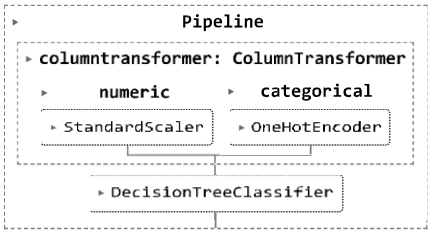
CREATING A COMPLETE PIPELINE FOR DECISION TREE

```
catagorical_processor = Pipeline(
    steps = [("one hot",OneHotEncoder(handle_unknown="ignore"))]
)
numeric_processor = Pipeline(
    steps = [("scaler",StandardScaler())]
)

preprocessing = ColumnTransformer(
    transformers=[
        ("numeric", numeric_processor, x.select_dtypes(exclude='object').columns),
        ("categorical", catagorical_processor, x.select_dtypes(include='object').columns)
    ]
)

pipe = make_pipeline(preprocessing,DecisionTreeClassifier())
pipe
```




```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 20,random_state=42)
```

```
pipe.fit(x_train,y_train)
```




```
pred = pipe.predict(x_test)
```

```
temp = pd.DataFrame({"actual_values":y_test,
                     "predicted_values":pred
                    })
temp
```

	actual_values	predicted_values
1041	Sales Executive	Sales Executive
184	Manufacturing Director	Healthcare Representative
1222	Human Resources	Human Resources
67	Research Scientist	Manufacturing Director
220	Laboratory Technician	Research Scientist
494	Sales Representative	Sales Representative
430	Laboratory Technician	Laboratory Technician
240	Laboratory Technician	Laboratory Technician
218	Sales Executive	Sales Executive
49	Laboratory Technician	Research Scientist
665	Sales Representative	Sales Representative
926	Sales Executive	Sales Executive
617	Healthcare Representative	Healthcare Representative
361	Laboratory Technician	Research Scientist
1423	Research Scientist	Research Scientist
1244	Research Scientist	Research Scientist
1250	Healthcare Representative	Laboratory Technician
752	Laboratory Technician	Research Scientist
271	Manager	Manager
1055	Research Director	Research Director

```
confusion_matrix(y_test,pred)
```

```
                          0, 0, 0, 0],
                          0, 0, 0, 0],
                          0, 4, 0, 0],
                          0,    0, 0],
                          0,    0, 2]
```

```
array([[1, 0, 1, 0, 0,
        [0, 1, 0, 0, 1,
        [0, 0, 2, 0, 0,
        [0, 0, 0, 1, 0,    0,
        [1, 0, 0, 0, 0,    0,
```

```
            [0, 0, 0, 0, 0, 1, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 2, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 3, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 2]])
```

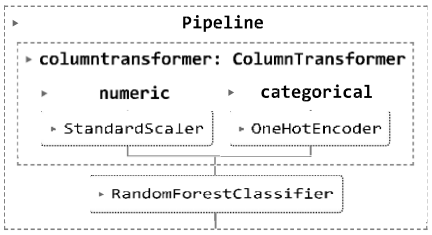
```
print(classification_report(y_test,pred))
```

```
                              precision    recall  f1-score   support

     Healthcare Representative       0.50      0.50      0.50         2
                Human Resources       1.00      1.00      1.00         1
          Laboratory Technician       0.67      0.33      0.67         6
                        Manager       1.00      1.00      1.00         1
       Manufacturing Director       0.00      0.00      0.00         1
            Research Director       1.00      1.00      1.00         1
          Research Scientist       0.33      0.67      0.33         3
             Sales Executive       1.00      1.00      1.00         3
          Sales Representative       1.00      1.00      1.00         2

                    accuracy                           0.65        20
                   macro avg       0.72      0.72      0.71        20
                weighted avg       0.70      0.65      0.65        20
```

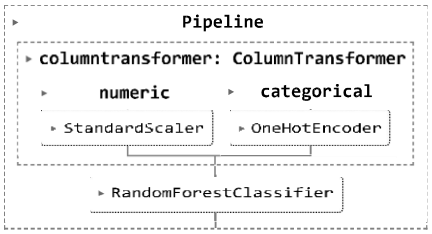
RANDOM FOREST

CREATING A COMPLETE PIPELIINE FOR RANDOMFOREST

```
pipe2 = make_pipeline(preprocessing,RandomForestClassifier(n_estimators=20))
pipe2
```




```
pipe2.fit(x_train,y_train)
```




```
pred = pipe2.predict(x_test)
```

```
temp = pd.DataFrame({"actual_values":y_test,
                     "predicted_values":pred
                    })
temp
```

	actual_values	predicted_values
1041	Sales Executive	Sales Executive
184	Manufacturing Director	Manufacturing Director
1222	Human Resources	Human Resources
67	Research Scientist	Healthcare Representative
220	Laboratory Technician	Healthcare Representative
494	Sales Representative	Sales Representative
430	Laboratory Technician	Research Scientist
240	Laboratory Technician	Research Scientist
218	Sales Executive	Sales Executive
49	Laboratory Technician	Research Scientist
665	Sales Representative	Sales Representative
926	Sales Executive	Sales Executive

```
confusion_matrix(y_test,pred)
```

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 1, 1, 1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 4, 4, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 2, 2, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 2]])
```

```
print(classification_report(y_test,pred))
```

```
                              precision    recall  f1-score   support

     Healthcare Representative       0.00      0.00      0.00         2
                Human Resources       1.00      1.00      1.00         1
            Laboratory Technician       1.00      1.00      1.00         6
                        Manager       0.00      0.00      0.00         1
           Manufacturing Director       0.33      0.33      0.33         1
                Research Director       0.00      0.00      0.00         1
               Research Scientist       0.33      0.33      0.33         3
                  Sales Executive       1.00      1.00      1.00         3
             Sales Representative       1.00      1.00      1.00         2

                         accuracy                           0.50        20
                        macro avg       0.52      0.54      0.47        20
                     weighted avg       0.67      0.50      0.48        20
```

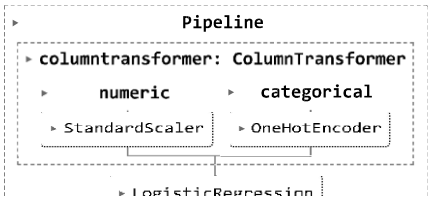
LOGISTIC REGRESSION

```
y1 = df.iloc[:,7]
x1 = df.iloc[:,list(range(7)) plt.subplots(3,3,figsize=(15,15))
```

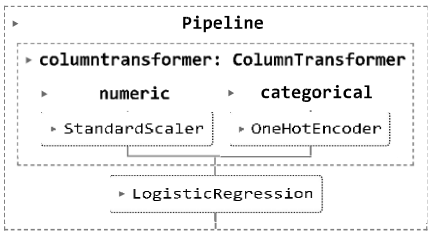
```
x1_train,x1_test,y1_train,y1_test = train_test_split(x1,y1,test_size = 20,random_state=42)
```

CREATING A COMPLETE PIPE LINE FOR LOGISTIC REGRESSION

```
preprocessing = ColumnTransformer(
    transformers=[
        ("numeric", numeric_processor, x1.select_dtypes(exclude='object').columns),
        ("categorical", catagorical_processor, x1.select_dtypes(include='object').columns)
    ]
)
pipe2 = make_pipeline(preprocessing,LogisticRegression())
pipe2
```




```
pipe1.fit(x1_train,y1_train)
```




```
pred = pipe1.predict(x1_test)
```

```
temp = pd.DataFrame({"actual_values":y1_test,
                     "predicted_values":pred
                    })
temp
```

	actual_values	predicted_values
1041	Male	Male
184	Female	Male
1222	Male	Male
67	Male	Male
220	Male	Male
494	Female	Male
430	Male	Male
240	Female	Male
218	Female	Male
49	Male	Male
665	Female	Male
926	Female	Male
617	Male	Male
361	Female	Male
1423	Male	Male
1244	Female	Male
1250	Male	Female
752	Female	Male
271	Male	Male
1055	Male	Male

```
print(classification_report(y_test,pred))
```

```
                           precision    recall  f1-score   support

                   Female       0.00      0.00      0.00       0.0
Healthcare Representative       1.00      1.00      1.00       2.0
          Human Resources       1.00      1.00      1.00       1.0
      Laboratory Technician       0.00      0.00      0.00       6.0
                     Male       0.33      0.33      0.33       0.0
                  Manager       0.00      0.00      0.00       1.0
     Manufacturing Director       0.33      0.33      0.33       1.0
         Research Director       1.00      1.00      1.00       1.0
        Research Scientist       1.00      1.00      1.00       3.0
```

```
      Sales Executive       0.00      0.00      0.00       3.0
 Sales Representative       0.00      0.00      0.00       2.0

             accuracy                           0.00      20.0
            macro avg       0.00      0.00      0.00      20.0
         weighted avg       0.00      0.00      0.00      20.0

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-c
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-c
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-c
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
```