TITLE 1: Enhancing social media influence prediction accuracy for marketing strategy through comparative analysis of Random Forest and Artificial Neural Network

P – low accuracy in the prediction

I – Random Forest

C – ANN

O – Accuracy

Title 2: Optimizing Social Media Influence Prediction Accuracy for Marketing Strategy in Comparison of Random Forest And Support Vector Machine

P – low accuracy in the prediction

I – Random Forest

C – SVM

O – Accuracy

Title 3: improving social media impact prediction accuracy for marketing strategy an analysis of Random Forest and Logistic Regression

P – low accuracy in the prediction

I – Random Forest

C – LOGISTIC REGRESSION

O – Accuracy

Title 4: Refining Social Media Influence Prediction A Comparative Evaluation Of Random Forest And Ada Boost For Improved Accuracy In Marketing Strategy

P – low accuracy in the prediction

I – Random forest

C – ADA BOOST

O – Accuracy

Random forest

import pandas as pd

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

# Load your CSV data into a DataFrame

data = pd.read\_csv('/final\_Data.csv')

# Assuming the last column is the target variable (label)

X = data.iloc[:, :-1]  # Features

y = data.iloc[:, -1]   # Target variable

# Encode categorical variables (if any)

label\_encoder = LabelEncoder()

X\_encoded = X.apply(label\_encoder.fit\_transform)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_encoded, y, test\_size=0.2, random\_state=42)

# Create a random forest classifier

rf\_model = RandomForestClassifier(random\_state=42)

# Define hyperparameters to tune

param\_grid = {

    'n\_estimators': [50, 100, 200],

    'max\_depth': [None, 10, 20],

    'min\_samples\_split': [2, 5, 10],

    'min\_samples\_leaf': [1, 2, 4]

}

# Use GridSearchCV for hyperparameter tuning

grid\_search = GridSearchCV(estimator=rf\_model, param\_grid=param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

# Get the best model from the search

best\_rf\_model = grid\_search.best\_estimator\_

# Make predictions on the testing set

y\_pred = best\_rf\_model.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy \* 100:.2f}%')

Accuracy: 95.24%

Logistic regression

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

# Load your CSV data into a DataFrame

data = pd.read\_csv('/content/final\_Data.csv')

# Assuming the last column is the target variable (label)

X = data.iloc[:, :-1]  # Features

y = data.iloc[:, -1]   # Target variable

# Encode categorical variables (if any)

label\_encoder = LabelEncoder()

X\_encoded = X.apply(label\_encoder.fit\_transform)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_encoded, y, test\_size=0.2, random\_state=42)

# Create a logistic regression classifier

logistic\_model = LogisticRegression(random\_state=42)

# Fit the model on the training data

logistic\_model.fit(X\_train, y\_train)

# Make predictions on the testing set

y\_pred = logistic\_model.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy \* 100:.2f}%')

Accuracy: 24.21%

/usr/local/lib/python3.10/dist-packages/sklearn/linear\_model/\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

<https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression>

n\_iter\_i = \_check\_optimize\_result(

Ada Boost

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

# Load your CSV data into a DataFrame

data = pd.read\_csv('/content/final\_Data.csv')

# Assuming the last column is the target variable (label)

X = data.iloc[:, :-1]  # Features

y = data.iloc[:, -1]   # Target variable

# Encode categorical variables (if any)

label\_encoder = LabelEncoder()

X\_encoded = X.apply(label\_encoder.fit\_transform)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_encoded, y, test\_size=0.2, random\_state=42)

# Create an AdaBoost classifier

adaboost\_model = AdaBoostClassifier(random\_state=42)

# Fit the model on the training data

adaboost\_model.fit(X\_train, y\_train)

# Make predictions on the testing set

y\_pred = adaboost\_model.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy \* 100:.2f}%')

Accuracy: 23.02%

Artifical Neural Network

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neural\_network import MLPClassifier

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load your dataset

# Replace 'your\_dataset.csv' with the actual file name

df = pd.read\_csv('/content/final\_Data.csv')

# Assuming 'target\_variable' is the column you want to predict

# Replace 'target\_variable' with the actual name

X = df.drop('QOL', axis=1)

y = df['QOL']

# Handle categorical variables using label encoding

label\_encoder = LabelEncoder()

X\_encoded = X.apply(label\_encoder.fit\_transform)

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_encoded, y, test\_size=0.2, random\_state=42)

# Standardize the data

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Create an MLP (Multi-Layer Perceptron) Classifier

ann\_model = MLPClassifier(hidden\_layer\_sizes=(100,), max\_iter=1000, random\_state=42)

ann\_model.fit(X\_train\_scaled, y\_train)

# Predictions on the test set

ann\_predictions = ann\_model.predict(X\_test\_scaled)

# Evaluate model performance

accuracy = accuracy\_score(y\_test, ann\_predictions)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Display classification report and confusion matrix

print("\nClassification Report:")

print(classification\_report(y\_test, ann\_predictions))

print("\nConfusion Matrix:")

sns.heatmap(confusion\_matrix(y\_test, ann\_predictions), annot=True, fmt="d", cmap="Blues", cbar=False)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

Accuracy: 63.10%

Classification Report:

precision recall f1-score support

2 0.00 0.00 0.00 2

3 0.88 1.00 0.93 7

4 0.88 0.54 0.67 13

5 0.62 0.45 0.53 22

6 0.66 0.76 0.71 38

7 0.62 0.70 0.65 53

8 0.60 0.75 0.67 56

9 0.57 0.49 0.53 43

10 0.67 0.33 0.44 18

accuracy 0.63 252

macro avg 0.61 0.56 0.57 252

weighted avg 0.63 0.63 0.62 252

Confusion Matrix:

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

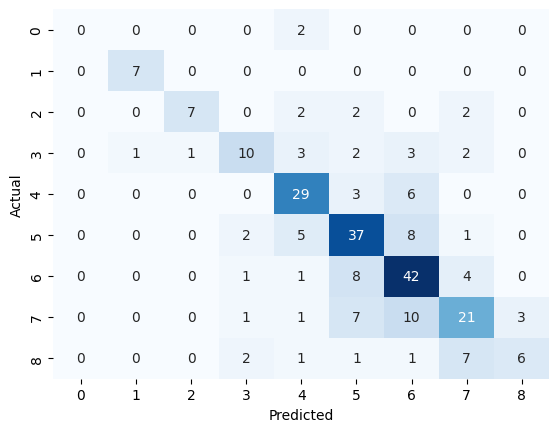
\_warn\_prf(average, modifier, msg\_start, len(result))

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_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-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))



Support vector machine

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import make\_pipeline

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load your dataset

# Replace 'your\_dataset.csv' with the actual file name

df = pd.read\_csv('/content/final\_Data.csv')

# Assuming 'target\_variable' is the column you want to predict

# Replace 'target\_variable' with the actual name

X = df.drop('QOL', axis=1)

y = df['QOL']

# Identify categorical columns

categorical\_columns = X.select\_dtypes(include=['object']).columns

# Create a column transformer for preprocessing

preprocessor = ColumnTransformer(

    transformers=[

        ('num', StandardScaler(), X.columns.difference(categorical\_columns)),

        ('cat', OneHotEncoder(), categorical\_columns)

    ])

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create an SVM (Support Vector Machine) Classifier with preprocessing pipeline

svm\_model = make\_pipeline(preprocessor, SVC(kernel='linear', C=1.0, random\_state=42))

svm\_model.fit(X\_train, y\_train)

# Predictions on the test set

svm\_predictions = svm\_model.predict(X\_test)

# Evaluate model performance

accuracy = accuracy\_score(y\_test, svm\_predictions)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Display classification report and confusion matrix

print("\nClassification Report:")

print(classification\_report(y\_test, svm\_predictions))

print("\nConfusion Matrix:")

sns.heatmap(confusion\_matrix(y\_test, svm\_predictions), annot=True, fmt="d", cmap="Blues", cbar=False)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

Accuracy: 42.46%

Classification Report:

precision recall f1-score support

2 0.00 0.00 0.00 2

3 0.42 0.71 0.53 7

4 0.60 0.23 0.33 13

5 0.33 0.27 0.30 22

6 0.34 0.29 0.31 38

7 0.51 0.60 0.55 53

8 0.39 0.62 0.48 56

9 0.54 0.30 0.39 43

10 0.22 0.11 0.15 18

accuracy 0.42 252

macro avg 0.37 0.35 0.34 252

weighted avg 0.43 0.42 0.41 252

Confusion Matrix:

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

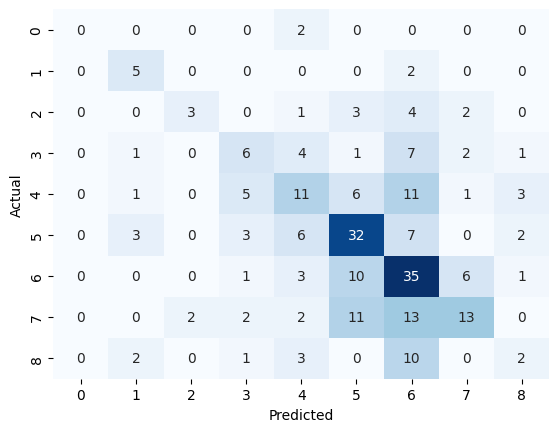
\_warn\_prf(average, modifier, msg\_start, len(result))

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_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-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))



Random forest

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import make\_pipeline

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load your dataset

# Replace 'your\_dataset.csv' with the actual file name

df = pd.read\_csv('/content/final\_Data.csv')

# Assuming 'target\_variable' is the column you want to predict

# Replace 'target\_variable' with the actual name

X = df.drop('QOL', axis=1)

y = df['QOL']

# Identify categorical columns

categorical\_columns = X.select\_dtypes(include=['object']).columns

# Create a column transformer for preprocessing

preprocessor = ColumnTransformer(

    transformers=[

        ('num', 'passthrough', X.columns.difference(categorical\_columns)),

        ('cat', OneHotEncoder(), categorical\_columns)

    ])

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Random Forest Classifier with preprocessing pipeline

rf\_model = make\_pipeline(preprocessor, RandomForestClassifier(n\_estimators=100, random\_state=42))

rf\_model.fit(X\_train, y\_train)

# Predictions on the test set

rf\_predictions = rf\_model.predict(X\_test)

# Evaluate model performance

accuracy = accuracy\_score(y\_test, rf\_predictions)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Display classification report and confusion matrix

print("\nClassification Report:")

print(classification\_report(y\_test, rf\_predictions))

print("\nConfusion Matrix:")

sns.heatmap(confusion\_matrix(y\_test, rf\_predictions), annot=True, fmt="d", cmap="Blues", cbar=False)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

Accuracy: 95.63%

Classification Report:

precision recall f1-score support

2 0.00 0.00 0.00 2

3 1.00 1.00 1.00 7

4 1.00 1.00 1.00 13

5 0.96 1.00 0.98 22

6 0.95 0.97 0.96 38

7 0.96 0.94 0.95 53

8 0.96 0.95 0.95 56

9 0.93 0.95 0.94 43

10 0.95 1.00 0.97 18

accuracy 0.96 252

macro avg 0.86 0.87 0.86 252

weighted avg 0.95 0.96 0.95 252

Confusion Matrix:

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

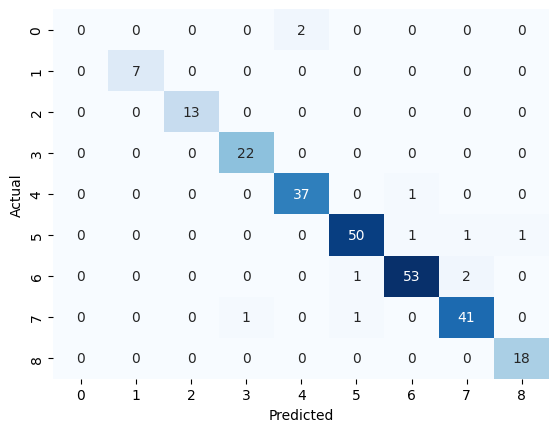
\_warn\_prf(average, modifier, msg\_start, len(result))

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_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-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))



Logistic regression

Accuracy: 31.75%

Classification Report:

precision recall f1-score support

2 0.00 0.00 0.00 2

3 0.56 0.71 0.63 7

4 1.00 0.15 0.27 13

5 0.50 0.36 0.42 22

6 0.28 0.24 0.26 38

7 0.30 0.26 0.28 53

8 0.31 0.50 0.38 56

9 0.29 0.28 0.28 43

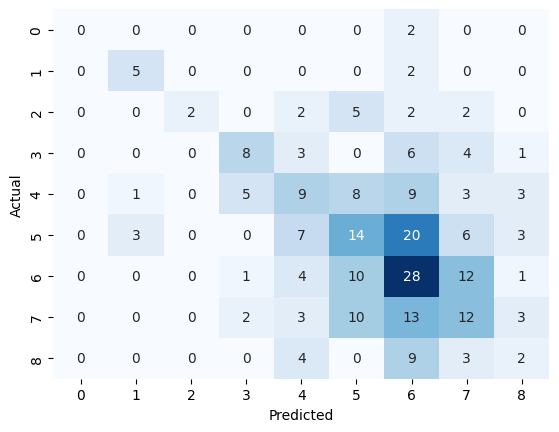
10 0.15 0.11 0.13 18

accuracy 0.32 252

macro avg 0.38 0.29 0.29 252

weighted avg 0.34 0.32 0.31 252

Confusion Matrix:



# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import make\_pipeline

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load your dataset

# Replace 'your\_dataset.csv' with the actual file name

df = pd.read\_csv('/content/final\_Data.csv')

# Assuming 'target\_variable' is the column you want to predict

# Replace 'target\_variable' with the actual name

X = df.drop('QOL', axis=1)

y = df['QOL']

# Identify categorical columns

categorical\_columns = X.select\_dtypes(include=['object']).columns

# Create a column transformer for preprocessing

preprocessor = ColumnTransformer(

    transformers=[

        ('num', StandardScaler(), X.columns.difference(categorical\_columns)),

        ('cat', OneHotEncoder(), categorical\_columns)

    ])

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Logistic Regression model with preprocessing pipeline

logreg\_model = make\_pipeline(preprocessor, LogisticRegression(random\_state=42))

logreg\_model.fit(X\_train, y\_train)

# Predictions on the test set

logreg\_predictions = logreg\_model.predict(X\_test)

# Evaluate model performance

accuracy = accuracy\_score(y\_test, logreg\_predictions)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Display classification report and confusion matrix

print("\nClassification Report:")

print(classification\_report(y\_test, logreg\_predictions))

print("\nConfusion Matrix:")

sns.heatmap(confusion\_matrix(y\_test, logreg\_predictions), annot=True, fmt="d", cmap="Blues", cbar=False)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

Ada Boost

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import make\_pipeline

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load your dataset

# Replace 'your\_dataset.csv' with the actual file name

df = pd.read\_csv('/content/final\_Data.csv')

# Assuming 'target\_variable' is the column you want to predict

# Replace 'target\_variable' with the actual name

X = df.drop('QOL', axis=1)

y = df['QOL']

# Identify categorical columns

categorical\_columns = X.select\_dtypes(include=['object']).columns

# Create a column transformer for preprocessing

preprocessor = ColumnTransformer(

    transformers=[

        ('num', 'passthrough', X.columns.difference(categorical\_columns)),

        ('cat', OneHotEncoder(), categorical\_columns)

    ])

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a base Decision Tree Classifier

base\_classifier = DecisionTreeClassifier(max\_depth=3)

# Create an AdaBoost Classifier with the base classifier and preprocessing pipeline

adaboost\_model = make\_pipeline(preprocessor, AdaBoostClassifier(base\_classifier, n\_estimators=50, learning\_rate=1.0, random\_state=42))

adaboost\_model.fit(X\_train, y\_train)

# Predictions on the test set

adaboost\_predictions = adaboost\_model.predict(X\_test)

# Evaluate model performance

accuracy = accuracy\_score(y\_test, adaboost\_predictions)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Display classification report and confusion matrix

print("\nClassification Report:")

print(classification\_report(y\_test, adaboost\_predictions))

print("\nConfusion Matrix:")

sns.heatmap(confusion\_matrix(y\_test, adaboost\_predictions), annot=True, fmt="d", cmap="Blues", cbar=False)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

Accuracy: 38.89%

Classification Report:

precision recall f1-score support

2 0.00 0.00 0.00 2

3 1.00 1.00 1.00 7

4 0.47 0.62 0.53 13

5 0.41 0.32 0.36 22

6 0.31 0.29 0.30 38

7 0.43 0.40 0.41 53

8 0.35 0.48 0.40 56

9 0.33 0.30 0.31 43

10 0.44 0.22 0.30 18

accuracy 0.39 252

macro avg 0.42 0.40 0.40 252

weighted avg 0.39 0.39 0.38 252

Confusion Matrix:

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_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-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_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-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

