This is the output of the testing of the first and second model (dir: classification\_training\_results/ Output\_folder\_{first/second}\_run)

using the FINAL\_TEST\_DATASET\_PREPROCESSED.csv (dir: .DATASETS/PPI\_TESTING\_DATASET/Test\_Dataset\_Preprocessed)

There are present the output testing files, using the ensemble testing method of the best models from the final Pareto Front and the testing using the best model from the Front. Both are present in the corresponding file names.

For finding the predicted labels for the reduced dataset (the one using the BLAST-P similarity check) you have to map each folder to the ‘test\_ds\_similarity\_check.csv’ using this script:

import pandas as pd

import numpy as np

# Load the original dataset

csv\_file = "test\_ds\_similarity\_check.csv"  # Update with the actual CSV filename

df = pd.read\_csv(csv\_file)

# List of result label files in order

label\_files = [

    "Output\_folder\_testing\_first\_run\_best\_model/result\_labels.txt",

    "Output\_folder\_testing\_first\_run\_ensemble\_model/result\_labels.txt",

    "Output\_folder\_testing\_second\_run\_best\_model/result\_labels.txt",

    "Output\_folder\_testing\_second\_run\_ensemble/result\_labels.txt"

]

# Column names for new labels

new\_columns = [

    "First\_Run\_Best\_Model",

    "First\_Run\_Ensemble\_Model",

    "Second\_Run\_Best\_Model",

    "Second\_Run\_Ensemble\_Model"

]

# Load label files and append as new columns

for file, col in zip(label\_files, new\_columns):

    labels = np.loadtxt(file, dtype=int)  # Assuming binary labels (0/1)

    df[col] = labels  # Append labels to DataFrame

# Save the updated dataset

output\_file = "test\_ds\_with\_labels.csv"

df.to\_csv(output\_file, index=False)

print(f"Updated dataset saved as {output\_file}")

Then for calculating the performance metrics on the similarity-filtered dataset you have to select only the columns with ‘0’ on the ‘similarity\_check’ column of "test\_ds\_with\_labels.csv” dataset that is produced in the previous step and run this script:

import pandas as pd

import numpy as np

from sklearn.metrics import accuracy\_score, precision\_recall\_fscore\_support, roc\_auc\_score, confusion\_matrix

# Load the dataset

df = pd.read\_csv("test\_ds\_with\_labels.csv")

# Extract true labels

y\_true = df["label"]

# Define model prediction columns

model\_columns = [

    "First\_Run\_Best\_Model",

    "First\_Run\_Ensemble\_Model",

    "Second\_Run\_Best\_Model",

    "Second\_Run\_Ensemble\_Model"

]

# Count rows before removal

initial\_rows = len(df)

# Remove rows where any y\_pred column has NaN

df = df.dropna(subset=model\_columns)

# Count rows after removal

final\_rows = len(df)

# Calculate the number of removed rows

num\_removed = initial\_rows - final\_rows

# Print removed rows count

print(f"Removed {num\_removed} rows that contained NaN in prediction columns.")

# Re-define model\_predictions with remaining rows

model\_predictions = {col: df[col] for col in model\_columns}

# Function to compute metrics

def compute\_metrics(y\_true, y\_pred):

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

    precision, recall, f1, \_ = precision\_recall\_fscore\_support(y\_true, y\_pred, average="binary")

    f2 = (5 \* precision \* recall) / (4 \* precision + recall) if (precision + recall) != 0 else 0

    roc\_auc = roc\_auc\_score(y\_true, y\_pred)

    # Compute Specificity (True Negative Rate)

    tn, fp, fn, tp = confusion\_matrix(y\_true, y\_pred).ravel()

    specificity = tn / (tn + fp) if (tn + fp) != 0 else 0

    return accuracy, specificity, recall, f1, f2, roc\_auc

# Store metrics for both runs

metrics\_results = {

    "Best\_Model": [],

    "Ensemble\_Model": []

}

# Compute metrics for each run

metrics\_results["Best\_Model"].append(compute\_metrics(y\_true[df.index], model\_predictions["First\_Run\_Best\_Model"]))

metrics\_results["Best\_Model"].append(compute\_metrics(y\_true[df.index], model\_predictions["Second\_Run\_Best\_Model"]))

metrics\_results["Ensemble\_Model"].append(compute\_metrics(y\_true[df.index], model\_predictions["First\_Run\_Ensemble\_Model"]))

metrics\_results["Ensemble\_Model"].append(compute\_metrics(y\_true[df.index], model\_predictions["Second\_Run\_Ensemble\_Model"]))

# Convert results to numpy arrays for easy computation of mean and std

metrics\_results["Best\_Model"] = np.array(metrics\_results["Best\_Model"])

metrics\_results["Ensemble\_Model"] = np.array(metrics\_results["Ensemble\_Model"])

# Metrics names

metric\_names = ["Accuracy", "Specificity", "Sensitivity (Recall)", "F1 Score", "F2 Score", "ROC-AUC Score"]

# Print results

for model\_type, metrics in metrics\_results.items():

    print(f"\n{model\_type} Metrics (Mean ± Std):")

    for i, metric\_name in enumerate(metric\_names):

        mean = np.mean(metrics[:, i])

        std = np.std(metrics[:, i])

        print(f"{metric\_name}: {mean:.4f} ± {std:.4f}")

The final similarity-check reduced test with features and predictions is this: TEST\_DS\_W\_FEATURES\_AND\_PREDICTIONS.csv