**final data project - 5 evaluation**

Evaluation:

This code executes the model evaluation phase of the project, following the modeling stage. The code has been modularized and enhanced to ensure clarity, reliability, and explainability of the results.

It includes comprehensive steps such as data preparation, model evaluation using appropriate metrics, feature importance analysis, and visualization of insights relevant to predicting rabies outbreaks.

This code includes: Data preparation:

* Transformation of the original dataset including parsing of date to extract year and month
* Engineering of a new feature: the number of rabies cases per year-month (Rabies\_Cases\_Per\_Month)
* Retention of relevant temporal features such as Event Per Year
* Encoding categorical columns and scaling of numeric features
* Model training and evaluation:
* Model: GradientBoostingClassifier trained using 5 different splits
* Balanced splits via stratify=y to preserve category distributions
* Option to use 5-fold Cross Validation for accuracy (optional flag)

Performance metrics:

* Evaluation using key classification metrics: Accuracy, Precision, Recall, and F1 Score (macro average)
* All metrics are reported as averaged values across 5 runs (no per-run printouts)

Feature importance analysis:

* Feature importance is calculated and visualized for both prediction targets:
* Region (geographic area of rabies outbreak)
* Month (temporal occurrence of outbreaks)
* A comparison graph highlights the contribution of each feature for both targets

Rabies event analysis:

* Construction and display of a pivot table showing monthly rabies cases across years

This phase provides essential insights into the model's decision-making process and supports the interpretation of which features most influence the predictions. It also ensures the model performs reliably and meets the predefined accuracy goals set in the business understanding phase.

# -------------------- Imports --------------------

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

# -------------------- Data Preparation --------------------

def load\_and\_prepare\_data(path):

    df = pd.read\_excel(path)

    df['Date'] = pd.to\_datetime(df['Date'])

    df['Year'] = df['Date'].dt.year

    df['Month'] = df['Date'].dt.month

    # Count rabies cases per year-month

    rabies\_counts = df.groupby(['Year', 'Month']).size().reset\_index(name='Rabies\_Cases\_Per\_Month')

    df = df.merge(rabies\_counts, on=['Year', 'Month'], how='left')

    # Drop unnecessary columns

    df = df.drop(columns=['Date', 'War Name', 'Index Event ID'])

    df['War in Israel'] = df['War in Israel'].map({'Yes': 1, 'No': 0})

    # Encode categorical columns

    label\_cols = ['Animal Species', 'Rabies Species', 'Region', 'Settlement', 'Region\_Weather', 'Month']

    for col in label\_cols:

        df[col] = LabelEncoder().fit\_transform(df[col].astype(str))

    # Scale numeric columns

    num\_cols = ['x', 'y', 'Avg Temperature', 'Monthly Precipitation (mm)',

                'Rainy Days', 'Rabies\_Cases\_Per\_Month', 'Event Per Year']

    df[num\_cols] = StandardScaler().fit\_transform(df[num\_cols])

    return df

# -------------------- Model Evaluation --------------------

def evaluate\_gradient\_boosting(df, target\_column, n\_runs=5, use\_cross\_val=False):

    X = df.drop(columns=[target\_column])

    y = df[target\_column]

    model = GradientBoostingClassifier(random\_state=42)

    if use\_cross\_val:

        accuracy = cross\_val\_score(model, X, y, cv=5, scoring='accuracy')

        print(f"\nAccuracy via Cross Validation for '{target\_column}': {np.mean(accuracy):.4f} ± {np.std(accuracy):.4f}")

        model.fit(X, y)

    else:

        acc\_list, prec\_list, rec\_list, f1\_list = [], [], [], []

        for i in range(n\_runs):

            X\_train, X\_test, y\_train, y\_test = train\_test\_split(

                X, y, test\_size=0.2, random\_state=i, stratify=y)

            model.fit(X\_train, y\_train)

            y\_pred = model.predict(X\_test)

            acc\_list.append(accuracy\_score(y\_test, y\_pred))

            prec\_list.append(precision\_score(y\_test, y\_pred, average='macro', zero\_division=0))

            rec\_list.append(recall\_score(y\_test, y\_pred, average='macro', zero\_division=0))

            f1\_list.append(f1\_score(y\_test, y\_pred, average='macro', zero\_division=0))

        print(f"\nPerformance summary for '{target\_column}':")

        print(f"Accuracy: {np.mean(acc\_list):.4f} ± {np.std(acc\_list):.4f}")

        print(f"Precision (avg): {np.mean(prec\_list):.4f}")

        print(f"Recall (avg): {np.mean(rec\_list):.4f}")

        print(f"F1 Score (avg): {np.mean(f1\_list):.4f}")

        model.fit(X, y)  # Train final model for feature importance

    importances = pd.Series(model.feature\_importances\_, index=X.columns).sort\_values(ascending=False)

    print("\nFeature Importances:")

    print(importances)

    # Plot

    plt.figure(figsize=(10, 6))

    importances.plot(kind='barh', color='darkcyan')

    plt.title(f'Feature Importance - Gradient Boosting (Target: {target\_column})')

    plt.xlabel('Importance')

    plt.gca().invert\_yaxis()

    plt.grid(True, linestyle='--', alpha=0.6)

    plt.tight\_layout()

    plt.show()

    return importances

# -------------------- Rabies Cases Pivot Table --------------------

def display\_rabies\_cases\_pivot(df):

    rabies\_counts = df.groupby(['Year', 'Month']).size().reset\_index(name='Rabies Case Count')

    pivot\_table = rabies\_counts.pivot(index='Year', columns='Month', values='Rabies Case Count').fillna(0).astype(int)

    print("\nRabies Case Count per Year-Month:")

    print(rabies\_counts)

    print("\nPivot Table:")

    print(pivot\_table)

# -------------------- Main Execution --------------------

file\_path = "/content/Rabies\_\_Weather\_\_War\_Combined\_1.4.25.xlsx"

df = load\_and\_prepare\_data(file\_path)

display\_rabies\_cases\_pivot(df)

# Region prediction

region\_importance = evaluate\_gradient\_boosting(df, target\_column='Region', n\_runs=5)

# Month prediction

month\_importance = evaluate\_gradient\_boosting(df, target\_column='Month', n\_runs=5)

# -------------------- Comparison Plot --------------------

comparison\_df = pd.DataFrame({

    'Feature': sorted(set(region\_importance.index).union(month\_importance.index))

})

comparison\_df['Region Importance'] = comparison\_df['Feature'].map(region\_importance).fillna(0)

comparison\_df['Month Importance'] = comparison\_df['Feature'].map(month\_importance).fillna(0)

comparison\_df = comparison\_df.sort\_values(by='Region Importance', ascending=False)

# Plot

x = np.arange(len(comparison\_df))

width = 0.35

plt.figure(figsize=(12, 6))

plt.bar(x - width/2, comparison\_df['Region Importance'], width, label='Region', color='orange')

plt.bar(x + width/2, comparison\_df['Month Importance'], width, label='Month', color='steelblue')

plt.xlabel('Feature')

plt.ylabel('Importance Score')

plt.title('Feature Importance Comparison: Region vs. Month')

plt.xticks(x, comparison\_df['Feature'], rotation=45, ha='right')

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.tight\_layout()

plt.show()

Rabies Case Count per Year-Month:

Year Month Rabies Case Count

0 2006 1 1

1 2006 4 1

2 2006 6 2

3 2006 7 1

4 2006 9 1

.. ... ... ...

185 2024 10 3

186 2024 11 2

187 2025 0 7

188 2025 4 9

189 2025 5 9

[190 rows x 3 columns]

Pivot Table:

Month 0 1 2 3 4 5 6 7 8 9 10 11

Year

2006 0 1 0 0 1 0 2 1 0 1 0 2

2007 0 6 1 2 0 0 3 1 0 0 1 1

2008 0 2 1 0 0 0 0 1 1 1 0 1

2009 2 15 23 10 4 2 2 0 4 3 10 10

2010 4 2 4 10 6 7 2 3 1 5 5 5

2011 5 1 2 1 6 2 6 4 1 0 1 2

2012 2 1 0 2 2 1 1 3 9 2 0 1

2013 1 3 2 0 2 4 3 6 2 0 1 4

2014 1 0 3 5 0 1 0 0 0 0 0 3

2015 1 4 4 4 3 0 0 4 2 2 1 1

2016 3 0 2 4 4 2 3 0 1 1 3 6

2017 3 14 20 21 3 2 2 2 1 1 2 3

2018 24 1 1 1 11 8 3 6 2 2 1 1

2019 4 3 2 2 1 1 1 0 0 1 2 0

2020 2 9 10 5 5 2 3 4 1 2 1 2

2021 5 3 2 5 6 2 8 3 3 2 1 0

2022 5 1 5 2 3 2 2 0 0 0 5 4

2023 8 1 3 6 4 4 4 3 4 3 4 1

2024 5 6 2 8 7 7 9 3 2 1 3 2

2025 7 0 0 0 9 9 0 0 0 0 0 0

Performance summary for 'Region':

Accuracy: 0.9510 ± 0.0147

Precision (avg): 0.8892

Recall (avg): 0.8859

F1 Score (avg): 0.8758

Feature Importances:

y 0.596275

x 0.328146

Region\_Weather 0.039703

Year 0.009896

Rabies Species 0.008795

Avg Temperature 0.004838

Event Per Year 0.004056

Animal Species 0.003301

Rabies\_Cases\_Per\_Month 0.002208

Settlement 0.001456

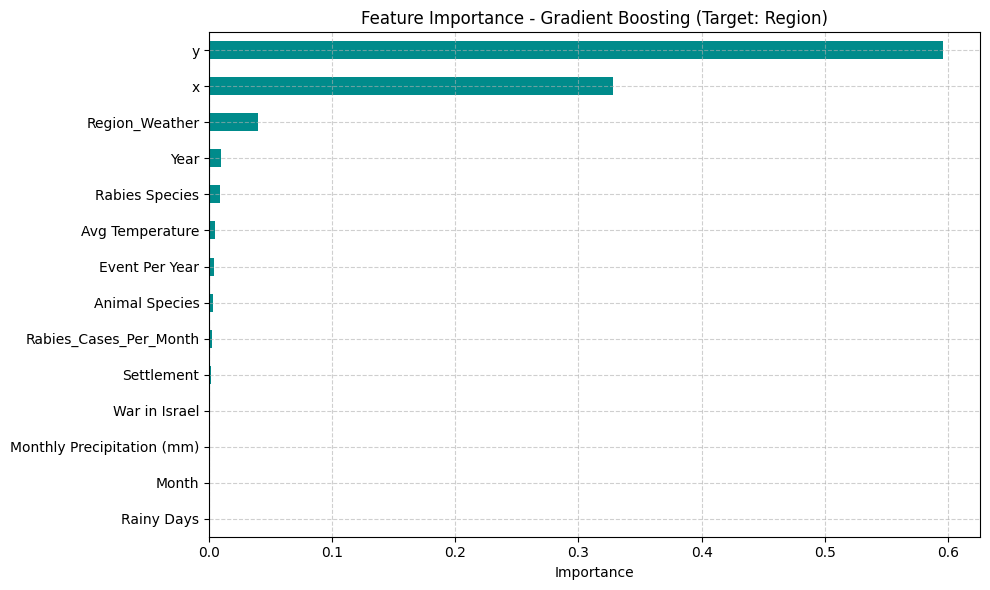
War in Israel 0.000836

Monthly Precipitation (mm) 0.000195

Month 0.000159

Rainy Days 0.000137

dtype: float64



Performance summary for 'Month':

Accuracy: 0.8937 ± 0.0304

Precision (avg): 0.8909

Recall (avg): 0.8775

F1 Score (avg): 0.8770

Feature Importances:

Avg Temperature 0.470151

Event Per Year 0.154299

Rabies\_Cases\_Per\_Month 0.137267

Year 0.063162

Monthly Precipitation (mm) 0.061300

Rainy Days 0.034320

x 0.024183

y 0.015059

Settlement 0.012182

Rabies Species 0.011242

Animal Species 0.006662

Region 0.006527

Region\_Weather 0.002080

War in Israel 0.001566

dtype: float64

תמונה שמכילה טקסט, צילום מסך, מספר, קו

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

תמונה שמכילה טקסט, קו, תרשים, עלילה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

**A different version with an Active Cross Validation Evaluation function:**

# -------------------- Imports --------------------

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, cross\_validate

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, make\_scorer

# -------------------- Data Preparation --------------------

def load\_and\_prepare\_data(path):

    df = pd.read\_excel(path)

    df['Date'] = pd.to\_datetime(df['Date'])

    df['Year'] = df['Date'].dt.year

    df['Month'] = df['Date'].dt.month

    # Count rabies cases per year-month

    rabies\_counts = df.groupby(['Year', 'Month']).size().reset\_index(name='Rabies\_Cases\_Per\_Month')

    df = df.merge(rabies\_counts, on=['Year', 'Month'], how='left')

    # Drop unnecessary columns

    df = df.drop(columns=['Date', 'War Name', 'Index Event ID'])

    df['War in Israel'] = df['War in Israel'].map({'Yes': 1, 'No': 0})

    # Encode categorical columns

    label\_cols = ['Animal Species', 'Rabies Species', 'Region', 'Settlement', 'Region\_Weather', 'Month']

    for col in label\_cols:

        df[col] = LabelEncoder().fit\_transform(df[col].astype(str))

    # Scale numeric columns

    num\_cols = ['x', 'y', 'Avg Temperature', 'Monthly Precipitation (mm)',

                'Rainy Days', 'Rabies\_Cases\_Per\_Month', 'Event Per Year']

    df[num\_cols] = StandardScaler().fit\_transform(df[num\_cols])

    return df

# -------------------- Model Evaluation (5 splits) --------------------

def evaluate\_gradient\_boosting(df, target\_column, n\_runs=5):

    X = df.drop(columns=[target\_column])

    y = df[target\_column]

    model = GradientBoostingClassifier(random\_state=42)

    acc\_list, prec\_list, rec\_list, f1\_list = [], [], [], []

    for i in range(n\_runs):

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(

            X, y, test\_size=0.2, random\_state=i, stratify=y)

        model.fit(X\_train, y\_train)

        y\_pred = model.predict(X\_test)

        acc\_list.append(accuracy\_score(y\_test, y\_pred))

        prec\_list.append(precision\_score(y\_test, y\_pred, average='macro', zero\_division=0))

        rec\_list.append(recall\_score(y\_test, y\_pred, average='macro', zero\_division=0))

        f1\_list.append(f1\_score(y\_test, y\_pred, average='macro', zero\_division=0))

    print(f"\nPerformance summary for '{target\_column}' (based on 5 random splits):")

    print(f"Accuracy: {np.mean(acc\_list):.4f} ± {np.std(acc\_list):.4f}")

    print(f"Precision (avg): {np.mean(prec\_list):.4f}")

    print(f"Recall (avg): {np.mean(rec\_list):.4f}")

    print(f"F1 Score (avg): {np.mean(f1\_list):.4f}")

    model.fit(X, y)

    importances = pd.Series(model.feature\_importances\_, index=X.columns).sort\_values(ascending=False)

    print("\nFeature Importances:")

    print(importances)

    plt.figure(figsize=(10, 6))

    importances.plot(kind='barh', color='darkcyan')

    plt.title(f'Feature Importance - Gradient Boosting (Target: {target\_column})')

    plt.xlabel('Importance')

    plt.gca().invert\_yaxis()

    plt.grid(True, linestyle='--', alpha=0.6)

    plt.tight\_layout()

    plt.show()

    return importances

# -------------------- Cross Validation Evaluation --------------------

def evaluate\_with\_cross\_validation(df, target\_column):

    X = df.drop(columns=[target\_column])

    y = df[target\_column]

    model = GradientBoostingClassifier(random\_state=42)

    scoring = {

        'accuracy': make\_scorer(accuracy\_score),

        'precision': make\_scorer(precision\_score, average='macro', zero\_division=0),

        'recall': make\_scorer(recall\_score, average='macro', zero\_division=0),

        'f1': make\_scorer(f1\_score, average='macro', zero\_division=0)

    }

    results = cross\_validate(model, X, y, cv=5, scoring=scoring)

    print(f"\nCross Validation results for '{target\_column}':")

    print(f"Accuracy: {results['test\_accuracy'].mean():.4f} ± {results['test\_accuracy'].std():.4f}")

    print(f"Precision (avg): {results['test\_precision'].mean():.4f}")

    print(f"Recall (avg): {results['test\_recall'].mean():.4f}")

    print(f"F1 Score (avg): {results['test\_f1'].mean():.4f}")

# -------------------- Rabies Cases Pivot Table --------------------

def display\_rabies\_cases\_pivot(df):

    rabies\_counts = df.groupby(['Year', 'Month']).size().reset\_index(name='Rabies Case Count')

    pivot\_table = rabies\_counts.pivot(index='Year', columns='Month', values='Rabies Case Count').fillna(0).astype(int)

    print("\nRabies Case Count per Year-Month:")

    print(rabies\_counts)

    print("\nPivot Table:")

    print(pivot\_table)

# -------------------- Main Execution --------------------

file\_path = "/content/Rabies\_\_Weather\_\_War\_Combined\_1.4.25.xlsx"

df = load\_and\_prepare\_data(file\_path)

display\_rabies\_cases\_pivot(df)

# Region prediction (train/test repeated splits)

region\_importance = evaluate\_gradient\_boosting(df, target\_column='Region', n\_runs=5)

# Month prediction

month\_importance = evaluate\_gradient\_boosting(df, target\_column='Month', n\_runs=5)

# Optional: Cross Validation based evaluation

evaluate\_with\_cross\_validation(df, target\_column='Region')

evaluate\_with\_cross\_validation(df, target\_column='Month')

# -------------------- Comparison Plot --------------------

comparison\_df = pd.DataFrame({

    'Feature': sorted(set(region\_importance.index).union(month\_importance.index))

})

comparison\_df['Region Importance'] = comparison\_df['Feature'].map(region\_importance).fillna(0)

comparison\_df['Month Importance'] = comparison\_df['Feature'].map(month\_importance).fillna(0)

comparison\_df = comparison\_df.sort\_values(by='Region Importance', ascending=False)

x = np.arange(len(comparison\_df))

width = 0.35

plt.figure(figsize=(12, 6))

plt.bar(x - width/2, comparison\_df['Region Importance'], width, label='Region', color='orange')

plt.bar(x + width/2, comparison\_df['Month Importance'], width, label='Month', color='steelblue')

plt.xlabel('Feature')

plt.ylabel('Importance Score')

plt.title('Feature Importance Comparison: Region vs. Month')

plt.xticks(x, comparison\_df['Feature'], rotation=45, ha='right')

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.tight\_layout()

plt.show()

Rabies Case Count per Year-Month:

Year Month Rabies Case Count

0 2006 1 1

1 2006 4 1

2 2006 6 2

3 2006 7 1

4 2006 9 1

.. ... ... ...

185 2024 10 3

186 2024 11 2

187 2025 0 7

188 2025 4 9

189 2025 5 9

[190 rows x 3 columns]

Pivot Table:

Month 0 1 2 3 4 5 6 7 8 9 10 11

Year

2006 0 1 0 0 1 0 2 1 0 1 0 2

2007 0 6 1 2 0 0 3 1 0 0 1 1

2008 0 2 1 0 0 0 0 1 1 1 0 1

2009 2 15 23 10 4 2 2 0 4 3 10 10

2010 4 2 4 10 6 7 2 3 1 5 5 5

2011 5 1 2 1 6 2 6 4 1 0 1 2

2012 2 1 0 2 2 1 1 3 9 2 0 1

2013 1 3 2 0 2 4 3 6 2 0 1 4

2014 1 0 3 5 0 1 0 0 0 0 0 3

2015 1 4 4 4 3 0 0 4 2 2 1 1

2016 3 0 2 4 4 2 3 0 1 1 3 6

2017 3 14 20 21 3 2 2 2 1 1 2 3

2018 24 1 1 1 11 8 3 6 2 2 1 1

2019 4 3 2 2 1 1 1 0 0 1 2 0

2020 2 9 10 5 5 2 3 4 1 2 1 2

2021 5 3 2 5 6 2 8 3 3 2 1 0

2022 5 1 5 2 3 2 2 0 0 0 5 4

2023 8 1 3 6 4 4 4 3 4 3 4 1

2024 5 6 2 8 7 7 9 3 2 1 3 2

2025 7 0 0 0 9 9 0 0 0 0 0 0

Performance summary for 'Region' (based on 5 random splits):

Accuracy: 0.9510 ± 0.0147

Precision (avg): 0.8892

Recall (avg): 0.8859

F1 Score (avg): 0.8758

Feature Importances:

y 0.596275

x 0.328146

Region\_Weather 0.039703

Year 0.009896

Rabies Species 0.008795

Avg Temperature 0.004838

Event Per Year 0.004056

Animal Species 0.003301

Rabies\_Cases\_Per\_Month 0.002208

Settlement 0.001456

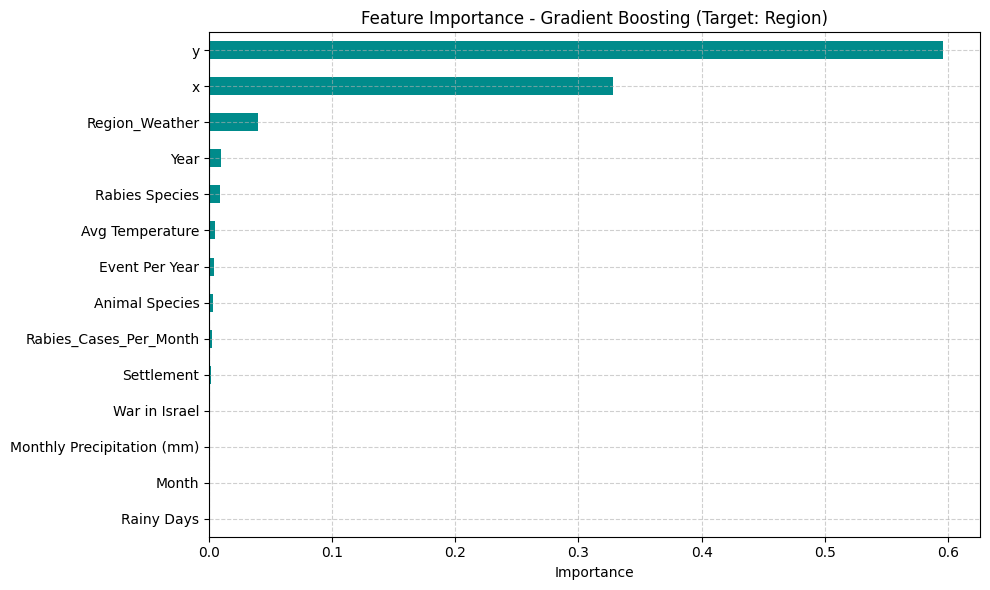
War in Israel 0.000836

Monthly Precipitation (mm) 0.000195

Month 0.000159

Rainy Days 0.000137

dtype: float64



Performance summary for 'Month' (based on 5 random splits):

Accuracy: 0.8937 ± 0.0304

Precision (avg): 0.8909

Recall (avg): 0.8775

F1 Score (avg): 0.8770

Feature Importances:

Avg Temperature 0.470151

Event Per Year 0.154299

Rabies\_Cases\_Per\_Month 0.137267

Year 0.063162

Monthly Precipitation (mm) 0.061300

Rainy Days 0.034320

x 0.024183

y 0.015059

Settlement 0.012182

Rabies Species 0.011242

Animal Species 0.006662

Region 0.006527

Region\_Weather 0.002080

War in Israel 0.001566

dtype: float64

תמונה שמכילה טקסט, צילום מסך, מספר, קו

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

Cross Validation results for 'Region':

Accuracy: 0.9061 ± 0.0647

Precision (avg): 0.8458

Recall (avg): 0.8318

F1 Score (avg): 0.8153

Cross Validation results for 'Month':

Accuracy: 0.5745 ± 0.0654

Precision (avg): 0.5706

Recall (avg): 0.5474

F1 Score (avg): 0.5138

תמונה שמכילה טקסט, קו, תרשים, עלילה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.