ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КВАЛІФІКАЦІЙНА РОБОТА

Метод виявлення шкідливого трафіка в комп'ютерній мережі

ст. гр. СПм-23-4 Наумова О.В.

Керівник: проф. каф. ЕОМ Міхаль О.П.

Мета та завдання роботи:

Метою кваліфікаційної роботи є розробка та практична реалізація методу виявлення шкідливого трафіка в комп'ютерній мережі з використанням алгоритму градієнтного бустингу, що забезпечує підвищену точність класифікації мережевих пакетів на основі аналізу їх ознак у реальному або наближеному до реального середовищі.

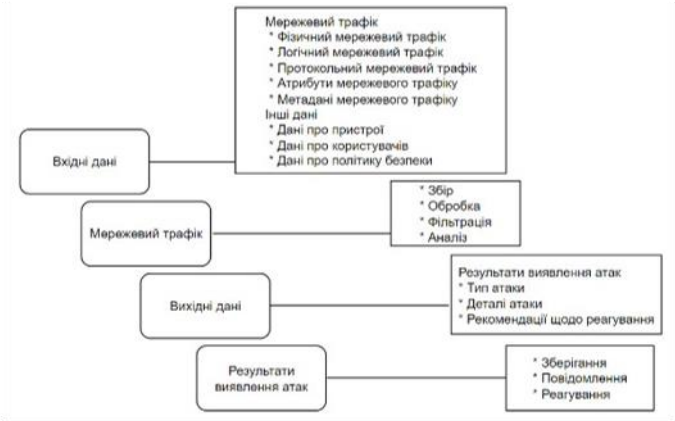
Об'єкт дослідження: мережевий трафік корпоративного середовища, що містить як легітимні, так і шкідливі з'єднання, представлені у вигляді векторів атрибутів у відкритому наборі даних.

Завдання:

- ❑ здійснити огляд сучасних підходів до виявлення шкідливого трафіка в комп'ютерних мережах та визначити їх сильні й слабкі сторони;
- ❑ провести аналіз можливостей алгоритмів машинного навчання для задач класифікації мережевого трафіка;
- ❑ обґрунтувати вибір відкритого набору даних для експериментального дослідження та здійснити його попередню обробку;
- ❑ розробити метод виявлення шкідливого трафіка та реалізувати його в середовищі Google Colab із використанням мови програмування Python;
- ❑ провести серію експериментів для оцінки ефективності побудованого класифікатора, порівнявши результати з існуючими підходами.

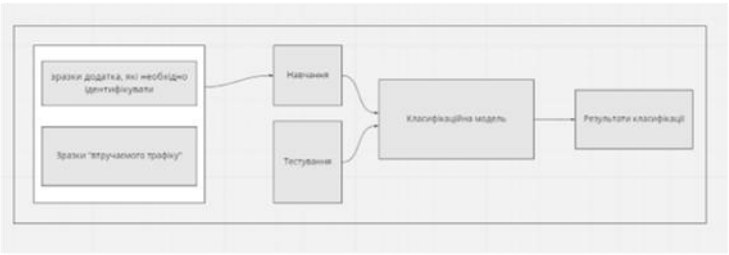
Узагальнена система виявлення шкідливих даних

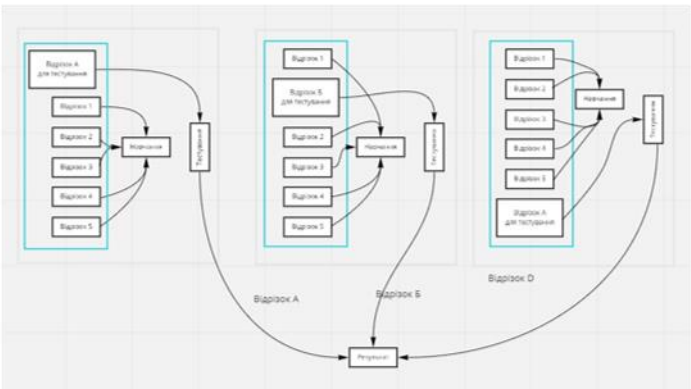
3



Існуючі рішення

4





Огляд датасету для поставленого завдання 5

UNSW_NB15_training-set.csv

1 to 10 of 175341 entries Filter

id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dloa
1	0.121478	tcp	-	FIN	6	4	258	172	74.08749	252	254	14158.94238	8495.36
2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	62	252	8395.112305	503571.
3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	62	252	1572.271851	60929.2
4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	62	252	2740.178955	3358.62
5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	254	252	8561.499023	3987.05
6	0.380537	tcp	-	FIN	10	6	534	268	39.41798	254	252	10112.02539	4709.13
7	0.637109	tcp	-	FIN	10	8	534	354	26.683033	254	252	6039.783203	3892.58
8	0.521584	tcp	-	FIN	10	8	534	354	32.593026	254	252	7377.527344	4754.74
9	0.542905	tcp	-	FIN	10	8	534	354	31.313031	254	252	7087.796387	4568.01
10	0.258687	tcp	-	FIN	10	6	534	268	57.985135	254	252	14875.12012	6927.29

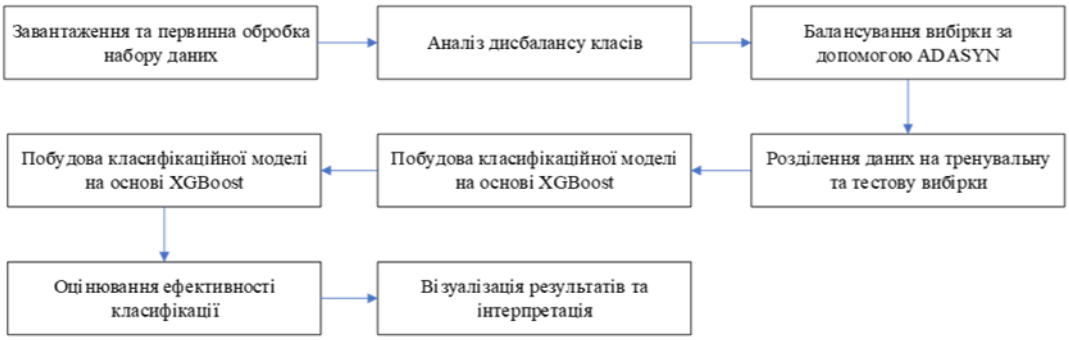
Show 10 per page 1 2 10 100 1000 10000 17000 17500 17530 17535

Обґрунтування вибору алгоритму машинного навчання 6

Модель	Точність	F1-міра	Повнота	Точність передбачення	Швидкість навчання	Інтерпретованість	Стійкість до дисбалансу	Потреба в налаштуванні
Logistic Regression	0,9437	0,94	0,94	0,94	Висока	Висока	Низька	Низька
Decision Tree	1	1	1	1	Висока	Висока	Середня	Низька
Random Forest	1	1	1	1	Середня	Середня	Висока	Середня
XGBoost	1	1	1	1	Низька	Низька	Висока	Висока

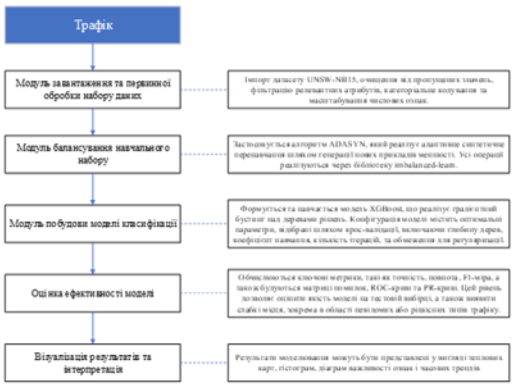
7

Метод виявлення шкідливого трафіка



Алгоритм роботи розробленого ПЗ

8



Система умовно поділяється на п'ять логічно пов'язаних рівнів.

На першому рівні реалізується завантаження та попередня обробка даних, що включає імпорт датасету UNSW-NB15, очищення від пропущених значень, фільтрацію релевантних атрибутів, категоріальне кодування та масштабування числових ознак. Цей блок побудовано з використанням інструментів pandas, numpy та sklearn.preprocessing, що дозволяє забезпечити стандартизацію вхідної вибірки та підготовку до машинного аналізу.

Другий рівень відповідає за балансування навчального набору. Застосовується алгоритм ADASYN, який реалізує адаптивне синтетичне перенавчання шляхом генерації нових прикладів меншості. Це дозволяє моделі коректніше навчатися на складних ділянках простору ознак, де можливе часте виникнення помилок другого роду. Усі операції реалізуються через бібліотеку imbalanced-learn.

Третій рівень – модуль побудови моделі класифікації. На цьому етапі формується та навчається модель XGBoost, що реалізує градієнтний бустинг над деревами рішень. Конфігурація моделі містить оптимальні параметри, відібрані шляхом крос-валідації, включаючи глибину дерев, коефіцієнт навчання, кількість ітерацій, та обмеження для регуляризації. Бібліотека xgboost у поєднанні з sklearn забезпечує обчислювальну ефективність, високу точність та контроль перенавчання.

Четвертий рівень системи – оцінка ефективності моделі. Тут обчислюються ключові метрики, такі як точність, повнота, F1-міра, а також будуються матриці помилок, ROC-криві та PR-криві. Цей рівень дозволяє оцінити якість моделі на тестовій вибірці, а також виявити слабкі місця, зокрема в області невідомих або рідкісних типів трафіку.

П'ятий рівень – візуалізація результатів та інтерпретація. Завдяки бібліотекам matplotlib, seaborn, plotly та іншим графічним засобам, результати моделювання можуть бути представлені у вигляді теплових карт, гістограм, діаграм важливості ознак і часових трендів. Цей блок є особливо важливим у контексті прийняття рішень безпековими аналітиками, оскільки дозволяє наочно інтерпретувати природу виявленого шкідливого трафіку.

Програмна реалізація 9

Модель реалізується у середовищі Google Colab із використанням таких бібліотек Python:

✓ pandas, numpy – для обробки даних;

✓ scikit-learn – для попередньої обробки, метрик і валідації;

✓ xgboost – реалізація алгоритму градієнтного бустингу;

✓ imblearn – модуль ADASYN для балансування;

✓ matplotlib, seaborn – для візуалізації результатів класифікації.

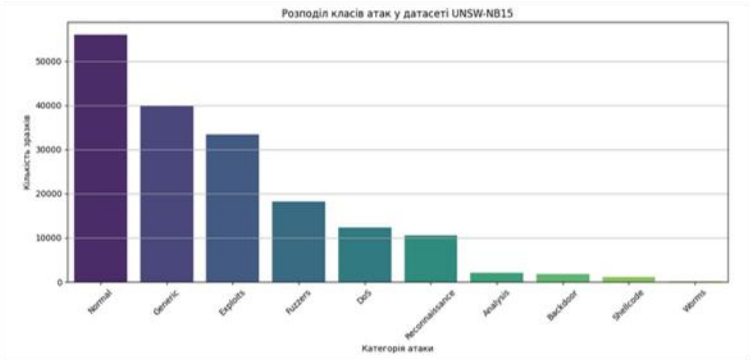
```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split, cross_val_score
    from sklearn.preprocessing import LabelEncoder, StandardScaler
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import classification_report, confusion_matrix, roc_auc_sc
    from imblearn.over_sampling import ADASYN
    from sklearn.decomposition import PCA
    from sklearn.manifold import TSNE
    import warnings
    warnings.filterwarnings('ignore')
```

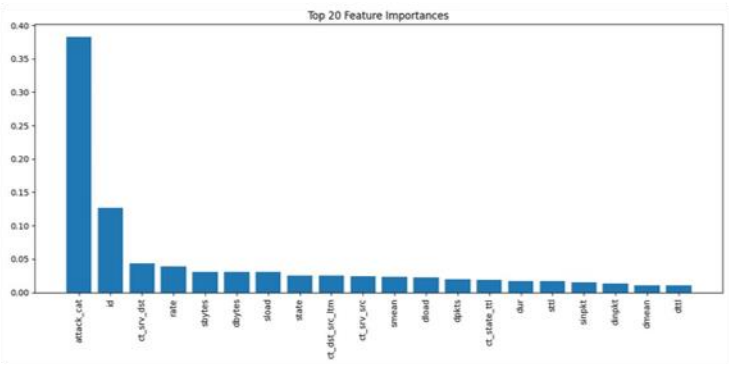
```
# шлях до файлу тут:
file_path = "/content/UNSW_NB15_training-set.csv"
df = pd.read_csv(file_path)
df.head()
```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826

5 rows × 45 columns

Аналіз результатів 10

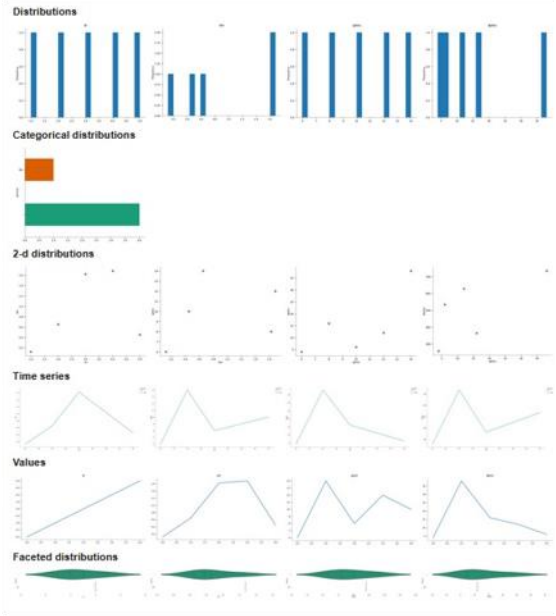


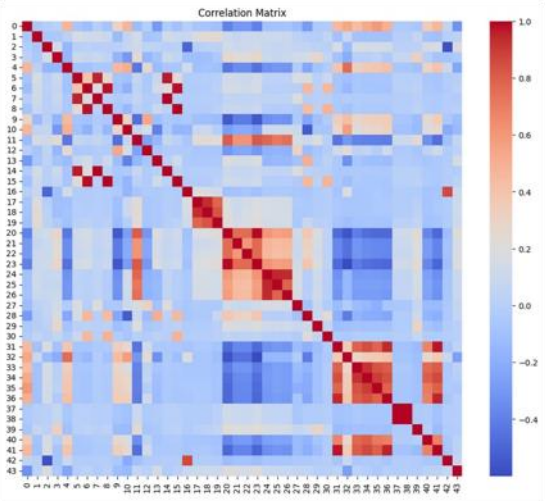



```
Logistic Regression Accuracy: 0.9437
[[16065   742]
 [ 1260 17480]]
              precision    recall  f1-score   support

           0       0.93      0.96      0.94     16807
           1       0.96      0.93      0.95     18740

    accuracy                           0.94     35547
   macro avg       0.94      0.94      0.94     35547
weighted avg       0.94      0.94      0.94     35547


Decision Tree Accuracy: 1.0000
[[16807     0]
 [    0 18740]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16807
           1       1.00      1.00      1.00     18740

    accuracy                           1.00     35547
   macro avg       1.00      1.00      1.00     35547
weighted avg       1.00      1.00      1.00     35547


Random Forest Accuracy: 1.0000
[[16807     0]
 [    0 18740]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16807
           1       1.00      1.00      1.00     18740

    accuracy                           1.00     35547
   macro avg       1.00      1.00      1.00     35547
weighted avg       1.00      1.00      1.00     35547


XGBoost Accuracy: 1.0000
[[16807     0]
 [    0 18740]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16807
           1       1.00      1.00      1.00     18740

    accuracy                           1.00     35547
   macro avg       1.00      1.00      1.00     35547
weighted avg       1.00      1.00      1.00     35547
```

Аналіз результатів

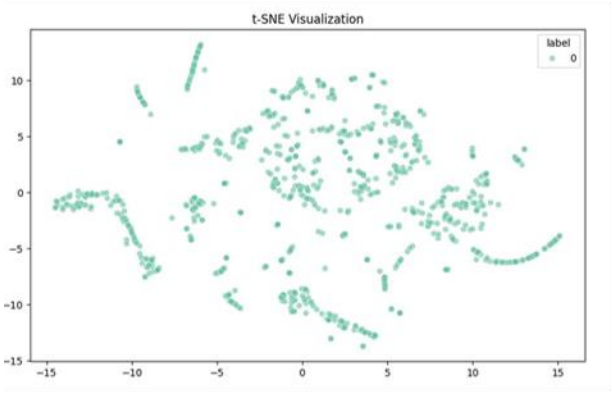
11

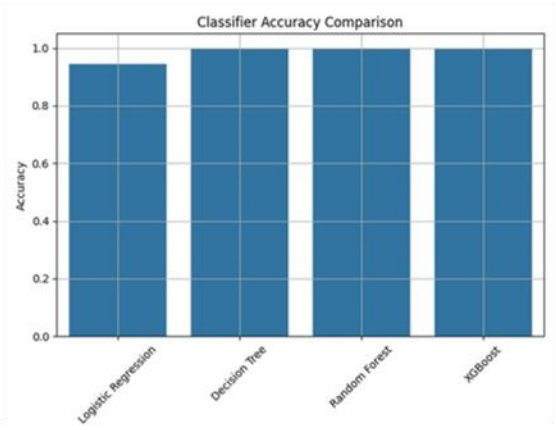




Аналіз результатів

12

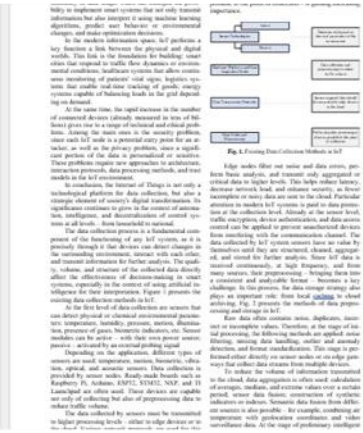




Публікація в фаховому виданні 13

Do K., Klymova I., Naumova E., Herevych M., Yankovskyi O. Data processing and analysis methods in IOT using machine learning. Системи управління, навігації та зв'язку, вип2. Полтава, 2025. С. 119-124.

DATA PROCESSING AND ANALYSIS METHODS IN IOT USING MACHINE LEARNING



Висновки 14

У ході виконання кваліфікаційної роботи було реалізовано систему виявлення атак у комп'ютерній мережі на основі даних про мережевий трафік. Розроблено метод класифікації шкідливого трафіку з урахуванням складності даних, дисбалансу класів та вимог до точності і швидкодії. Проведено теоретичний аналіз предметної області, обрано відповідні алгоритми машинного навчання, сформовано експериментальне середовище та виконано глибоку аналітику результатів.

На основі датасету UNSW-NB15, який містить широке представлення сучасних типів мережевих атак, було проведено попередню обробку даних, включаючи кодування ознак, масштабування та балансування класів за допомогою техніки ADASYN, що дало змогу зменшити вплив нерівномірного розподілу зразків на результати класифікації. Для виявлення найбільш інформативних параметрів було застосовано аналіз кореляційної матриці та оцінку важливості ознак.

У рамках порівняння алгоритмів було протестовано чотири популярні моделі: Logistic Regression, Decision Tree, Random Forest та XGBoost. Результати експериментів показали, що ансамблеві методи досягли найвищої точності, повноти й F1-міри, демонструючи ефективність у виявленні як нормального, так і шкідливого трафіку. При цьому Logistic Regression показала дещо нижчі, але стабільні результати, що є прийнятним у задачах, де пріоритет надається швидкості й інтерпретованості.

Візуалізація результатів із використанням методів PCA та t-SNE дозволила підтвердити наявність кластераційних структур у багатовимірному просторі ознак, а аналіз важливості ознак і SHAP-пояснень забезпечив інтерпретацію роботи моделей на прикладах.

Таким чином, розроблений метод успішно реалізує завдання інтелектуального виявлення атак у мережевому трафіку з використанням сучасних підходів машинного навчання. Запропоноване рішення може бути інтегроване в системи інформаційної безпеки для підвищення рівня захисту корпоративних мереж. Надалі перспективним напрямом розвитку є використання глибоких нейронних мереж, а також адаптація моделі до потокових даних для реального часу.

ДОДАТОК Б

Програмний код

Б.1 Підготовка середовища та даних

```
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,
cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
confusion_matrix, roc_auc_score, roc_curve
from imblearn.over_sampling import ADASYN
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import warnings
warnings.filterwarnings('ignore')

# шлях до файлу тут:
file_path = "/content/UNSW_NB15_training-set.csv"
df = pd.read_csv(file_path)
df.head()
# Оновлення назв колонок відповідно до специфікації UNSW-NB15
df.columns = [col.strip() for col in df.columns]
df.columns
```

Б.2 Кодування міток та тренування моделі

```
# Кодування міток
if 'label' not in df.columns:
    df.rename(columns={df.columns[-1]: 'label'}, inplace=True)
X = df.drop('label', axis=1)
y = df['label']

# Кодування категоріальних ознак
for col in X.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])

# Масштабування
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Балансування ADASYN
X_resampled, y_resampled = ADASYN().fit_resample(X_scaled, y)
# Тренування моделі
X_train, X_test, y_train, y_test = train_test_split(X_resampled,
y_resampled, test_size=0.3, random_state=42)
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred))
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'XGBoost': XGBClassifier(use_label_encoder=False,
eval_metric='logloss')
}

results = {}
for name, model in models.items():
    model.fit(X_resampled, y_resampled)
    y_pred = model.predict(X_test_scaled)
    acc = accuracy_score(y_test, y_pred)
    results[name] = acc
    print(f"\n{name} Accuracy: {acc:.4f}")
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))
```

Б.3 Візуалізація результатів

```
corr_matrix = pd.DataFrame(X_resampled).corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, cmap='coolwarm', annot=False)
plt.title("Correlation Matrix")
plt.savefig("/mnt/data/correlation_matrix.png")
plt.show()

importances = model.feature_importances_
indices = np.argsort(importances)[::-1][:20]
plt.figure(figsize=(12, 6))
plt.title("Top 20 Feature Importances")
plt.bar(range(len(indices)), importances[indices],
align="center")
plt.xticks(range(len(indices)), [df.columns[i] for i in
indices], rotation=90)
plt.tight_layout()
plt.savefig("/mnt/data/feature_importance.png")
plt.show()
```

```
# Preprocessing: drop nulls, encode labels
df = df.dropna()
label_col = 'label' if 'label' in df.columns else df.columns[-1]
X = df.drop(columns=[label_col])
y = df[label_col]
# Encode non-numeric columns
X = X.apply(lambda col: LabelEncoder().fit_transform(col) if
col.dtypes == 'object' else col)
y = LabelEncoder().fit_transform(y)
# Split and scale
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
plt.figure(figsize=(8,5))
sns.barplot(x=list(results.keys()), y=list(results.values()))
plt.ylabel('Accuracy')
plt.title('Classifier Accuracy Comparison')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```