

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи
із дисципліни «СИСТЕМИ ГЛИБИННОГО НАВЧАННЯ»
на тему:
Нейромережеве розпізнавання кібератак

Виконав:
студент групи КМ-13
Онищенко В.С.

Перевірив:
Терейковський І. А.

Київ — 2024

ЗМІСТ

1	Теорія	3
2	Попередня обробка даних	4
3	Результати для атак типу Perl	5
4	Результати для атак типу Neptune	6
5	Висновки	8
6	Код роботи програми	9

1. Теорія

Ймовірнісна нейронна мережа (PNN) базується на використанні ймовірностей для класифікації даних. Основною ідеєю PNN є оцінка апіорних ймовірностей класів та ймовірностей за умови певного значення атрибутів. PNN зазвичай використовується для задач класифікації з дискретними або числовими ознаками. Для роботи моделі необхідно виконати попередню обробку даних: масштабування числових атрибутів та кодування категоріальних.

У цьому завданні метою було виявлення атак у датасеті NSL-KDD, який містить з'єднання, класифіковані як нормальні або атакуючі. Спочатку фокус був на атаках типу "perl", але через їх недостатню кількість фокус було змінено на атаку "neptune", яка є популярнішою в цьому датасеті.

2. Попередня обробка даних

Для обробки даних ми розділили їх на категоріальні та числові колонки. Для категоріальних було використано метод One-Hot Encoding, а числові дані масштабувалися.

Listing 1: Функція для попередньої обробки даних

```
1 categorical_columns = ['protocol_type', 'service', 'flag']
2 numeric_columns = [col for col in df.columns if col not in
3                     categorical_columns + ['is_perl_attack']]
4
5 preprocessor = ColumnTransformer(
6     transformers=[
7         ('num', StandardScaler(), numeric_columns),
8         ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_columns)
9     ]
10 )
11 X = preprocessor.fit_transform(X)
```

Ця частина коду дозволяє стандартизувати числові дані (наприклад, обсяг трафіку) та перетворити текстові значення (наприклад, 'protocol_type') у числові вектори.

3. Результати для атак типу Perl

Після навчання моделі було отримано наступний розподіл даних:

- У навчальній вибірці: 2 приклади атак типу "perl".
- У тестовій вибірці: 1 приклад атак типу "perl".

Через недостатню кількість даних модель не змогла ефективно розпізнати цей тип атак. Це підтверджується звітом про класифікацію, в якому точність, повнота та F1-метрика для цього класу є невизначеними.

```
[17] train_and_evaluate_pnn(X_train, X_test, y_train, y_test)
```

Звіт про класифікацію:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25194
1	0.00	0.00	0.00	1
accuracy			1.00	25195
macro avg	0.50	0.50	0.50	25195
weighted avg	1.00	1.00	1.00	25195

Точність моделі: 1.00
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-de
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-de
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-de
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

```
y_train.value_counts()
```

is_perl_attack	count
0	100776
1	2

dtype: int64

```
[19] y_test.value_counts()
```

is_perl_attack	count
0	25194
1	1

dtype: int64

На рисунку показано результати класифікації для атак типу "perl". Видно, що модель працює добре для інших класів, але не змогла коректно класифікувати "perl" через відсутність достатньої кількості прикладів.

4. Результати для атак типу Neptune

Щоб перевірити ефективність моделі, ми змінили досліджуваний тип атак на "neptune", який є більш поширеним у датасеті. У результаті було отримано такий розподіл даних:

- У навчальній вибірці: 32,986 прикладів атак типу "neptune".
- У тестовій вибірці: 8,228 прикладів атак типу "neptune".

Результати класифікації показують високу точність (99%), що підтверджує ефективність роботи моделі. Звіт про класифікацію для цього випадку наведено нижче:

```
[23] train_and_evaluate_pnn(X_train, X_test, y_train, y_test)
```

Звіт про класифікацію:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	16967
1	0.97	1.00	0.98	8228
accuracy			0.99	25195
macro avg	0.98	0.99	0.99	25195
weighted avg	0.99	0.99	0.99	25195

Точність моделі: 0.99

```
y_train.value_counts()
```

count

is_perl_attack	count
0	67792
1	32986

dtype: int64

```
[25] y_test.value_counts()
```

count

is_perl_attack	count
0	16967
1	8228

dtype: int64

На рисунку видно, що для класу "neptune" модель досягла високих показників точності, повноти та F1-метрики. Це підтверджує, що PNN ефективно працює з типами атак, для яких доступно достатньо даних.

5. Висновки

Результати експериментів показують, що:

- Ймовірнісна нейронна мережа (PNN) ефективно працює для задач класифікації за умови наявності достатньої кількості даних.
- Для атак типу "perl" недостатньо даних у датасеті NSL-KDD, тому фокус було змінено на більш поширений тип атак "neptune".
- Модель показала високі результати для атак типу "neptune" із загальною точністю 99%, підтверджуючи її ефективність.

У майбутньому слід зосередитися на:

- Збалансуванні даних у датасеті для рідкісних типів атак.
- Застосуванні методів збільшення даних (data augmentation) для класів з малою кількістю прикладів.
- Використанні ансамблевих моделей для покращення результатів класифікації.

6. Код роботи програми

```
1 import os
2 import requests
3 import pandas as pd
4 from tqdm import tqdm
5 from sklearn.model_selection import train_test_split
6 from sklearn.preprocessing import StandardScaler, OneHotEncoder
7 from sklearn.compose import ColumnTransformer
8
9 # Функція для завантаження файлу з баром прогресу
10 def download_file_with_progress(url, save_path):
11     response = requests.get(url, stream=True)
12     total_size = int(response.headers.get('content-length', 0))
13     with open(save_path, 'wb') as f, tqdm(
14         desc=f"Завантаження {os.path.basename(save_path)}",
15         total=total_size,
16         unit='B',
17         unit_scale=True,
18         unit_divisor=1024,
19     ) as bar:
20         for chunk in response.iter_content(1024):
21             f.write(chunk)
22             bar.update(len(chunk))
23     print(f"Файл завантажено: {save_path}")
24
25 # Завантаження та перевірка датасету
26 def download_and_prepare_nsl_kdd():
27     dataset_url = "https://github.com/defcom17/NSL_KDD/raw/master/KDDTrain+.txt"
28     save_dir = "nsl_kdd"
29     os.makedirs(save_dir, exist_ok=True)
30     train_file_path = os.path.join(save_dir, "KDDTrain+.txt")
31
32     # Перевірка, чи файл вже існує
33     if not os.path.exists(train_file_path):
34         print("Файл не знайдено, починаю завантаження...")
35         download_file_with_progress(dataset_url, train_file_path)
36     else:
```

```

37     print(f"Файл вже існує: {train_file_path}")
38
39     return train_file_path
40
41 # Завантаження, обробка та підготовка датасету
42 def load_and_prepare_data():
43     file_path = download_and_prepare_nsl_kdd()
44
45     # Опис колонок NSL-KDD, з урахуванням додаткових полів
46     columns = [
47         "duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
48         "land", "wrong_fragment", "urgent", "hot", "num_failed_logins",
49         "logged_in", "num_compromised", "root_shell", "su_attempted",
50         "num_root", "num_file_creations", "num_shells", "num_access_files",
51         "num_outbound_cmds", "is_host_login", "is_guest_login",
52         "count", "srv_count", "error_rate", "srv_error_rate",
53         "rerror_rate", "srv_rerror_rate", "same_srv_rate", "diff_srv_rate",
54         "srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",
55         "dst_host_same_srv_rate", "dst_host_diff_srv_rate",
56         "dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate",
57         "dst_host_serror_rate", "dst_host_srv_serror_rate",
58         "dst_host_rerror_rate", "dst_host_srv_rerror_rate", "label", "difficulty"
59     ]
60
61     # Завантаження датасету
62     df = pd.read_csv(file_path, names=columns)
63
64     # Попередня обробка: створення цільової змінної
65     df['is_perl_attack'] = df['label'].apply(
66         # lambda x: 1 if isinstance(x, str) and 'neptune' in x.lower() else 0
67         lambda x: 1 if isinstance(x, str) and 'neptune' in x.lower() else 0
68     )
69
70     # Видалення початкових міток та "difficulty"
71     df.drop(['label', 'difficulty'], axis=1, inplace=True)
72
73     # Поділ колонок на числові та категоріальні
74     categorical_columns = ['protocol_type', 'service', 'flag']

```

```

75     numeric_columns = [col for col in df.columns if col not in
76                        categorical_columns + ['is_perl_attack']]
77
78     # Поділ на вхідні дані та цільову змінну
79     X = df.drop('is_perl_attack', axis=1)
80     y = df['is_perl_attack']
81
82     # Побудова пайплайну для обробки даних
83     preprocessor = ColumnTransformer(
84         transformers=[
85             ('num', StandardScaler(), numeric_columns),
86             ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_columns)
87         ]
88     )
89
90     # Застосування пайплайну до даних
91     X = preprocessor.fit_transform(X)
92
93     # Поділ на навчальні та тестові дані
94     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
95                                                         random_state=42)
96
97     return X_train, X_test, y_train, y_test
98
99 X_train, X_test, y_train, y_test = load_and_prepare_data()
100 print(f"Розмір навчальної вибірки: {X_train.shape}")
101 print(f"Розмір тестової вибірки: {X_test.shape}")
102
103 from sklearn.naive_bayes import GaussianNB
104 from sklearn.metrics import classification_report, accuracy_score
105
106 def train_and_evaluate_pnn(X_train, X_test, y_train, y_test):
107     # Створення моделі PNN на( основі ймовірностей)
108     pnn_model = GaussianNB()
109
110     # Навчання моделі
111     pnn_model.fit(X_train, y_train)
112

```

```
113     # Прогнозування
114     y_pred = pnn_model.predict(X_test)
115
116     # Оцінка продуктивності моделі
117     print("Звіт про класифікацію:")
118     print(classification_report(y_test, y_pred))
119     print(f"Точність моделі: {accuracy_score(y_test, y_pred):.2f}")
120
121 train_and_evaluate_pnn(X_train, X_test, y_train, y_test)
122
123 y_train.value_counts()
124
125 y_test.value_counts()
```