

**MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE**

**NATIONAL TECHNICAL UNIVERSITY OF UKRAINE**

**" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"**

**Artem Volokyta**

# **Software Security**

**Lab Work 1**

**kulubecioglu Mehmet**

**Class Number 12**

**IM-14 FIOT**

**Kyiv**

**IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE**

**2024**

# **Intrusion Detection System with Machine Learning**

## **Introduction**

This report covers the work done to develop an Intrusion Detection System (IDS) using the NSL-KDD dataset available on the Kaggle platform. In this process, machine learning methods were employed to analyze and classify the data.

### **Step 1: Finding the NSL-KDD Dataset and Notebook**

First, I logged into my Kaggle account and navigated to the NSL-KDD dataset page. I scrolled down the page to find the 'Most Votes' list, where I located the notebook titled 'Intrusion detection with ML&DL.' I clicked the 'Create Copy & Edit' button in the upper right corner of the notebook to create a copy of it in my Kaggle environment.

### **Step 2: Selecting a Subset of 10,000 Rows from the Dataset**

In the notebook, I proceeded to the 'exploring the dataset' section. Here, I needed to add a new code cell immediately below the `data_train.head()` code cell. To do this, I clicked between the cells and pressed the + Code button on the left. I wrote the following code in the new cell:

Kaggle Notebook interface showing the first cell of code. The code selects a subset of 10,000 rows from the dataset.

```
i = 11 # My class number is 12 so I will write it as 12-1= 11
data_train = data_train.iloc[i*10000:(i+1)*10000]
data_train.head()
data_train.info()
```

The output shows the first 15 rows of the selected data:

#	Column	Non-Null Count	Dtype
0	0	10000 non-null	int64
1	tcp	10000 non-null	object
2	ftp_data	10000 non-null	object
3	SF	10000 non-null	object
4	491	10000 non-null	int64
5	0.1	10000 non-null	int64
6	0.2	10000 non-null	int64
7	0.3	10000 non-null	int64
8	0.4	10000 non-null	int64
9	0.5	10000 non-null	int64
10	0.6	10000 non-null	int64
11	0.7	10000 non-null	int64
12	0.8	10000 non-null	int64
13	0.9	10000 non-null	int64
14	0.10	10000 non-null	int64
15	0.11	10000 non-null	int64

Kaggle Notebook interface showing the second cell of code. The code displays the first 42 rows of the selected data.

```
16 0.12 10000 non-null int64
17 0.13 10000 non-null int64
18 0.14 10000 non-null int64
19 0.15 10000 non-null int64
20 0.16 10000 non-null int64
21 0.17 10000 non-null int64
22 2 10000 non-null int64
23 2.1 10000 non-null int64
24 0.00 10000 non-null float64
25 0.00.1 10000 non-null float64
26 0.00.2 10000 non-null float64
27 0.00.3 10000 non-null float64
28 1.00 10000 non-null float64
29 0.00.4 10000 non-null float64
30 0.00.5 10000 non-null float64
31 150 10000 non-null int64
32 25 10000 non-null int64
33 0.17.1 10000 non-null float64
34 0.03 10000 non-null float64
35 0.17.2 10000 non-null float64
36 0.00.6 10000 non-null float64
37 0.00.7 10000 non-null float64
38 0.00.8 10000 non-null float64
39 0.05 10000 non-null float64
40 0.00.9 10000 non-null float64
41 normal 10000 non-null object
42 20 10000 non-null int64

dtypes: float64(15), int64(24), object(4)
memory usage: 3.3+ MB
```

This code allowed me to select a subset of 10,000 rows from the dataset. Next, I ran the `data\_train.head()` and `data\_train.info()` functions to verify that the data was selected correctly. I saved the outputs of these cells as screenshots.

## Resetting the Indices

To reset the row indices, I added a code cell and executed the following code:

Intrusion Detection System with ML&... Draft saved

File Edit View Run Settings Add-ons Help

+ | ✂ | 📄 | 📄 | ▶ ▶▶ Run All | Code ▾

● Draft Session (1h:4m) | HDD | CPU | RAM | 🔌 | ↺ | ⋮

26	0.00.2	0 non-null	float64
27	0.00.3	0 non-null	float64
28	1.00	0 non-null	float64
29	0.00.4	0 non-null	float64
30	0.00.5	0 non-null	float64
31	150	0 non-null	int64
32	25	0 non-null	int64
33	0.17.1	0 non-null	float64
34	0.03	0 non-null	float64
35	0.17.2	0 non-null	float64
36	0.00.6	0 non-null	float64
37	0.00.7	0 non-null	float64
38	0.00.8	0 non-null	float64
39	0.05	0 non-null	float64
40	0.00.9	0 non-null	float64
41	normal	0 non-null	object
42	20	0 non-null	int64

dtypes: float64(15), int64(24), object(4)  
memory usage: 132.0+ bytes

+ Code + Markdown

▶

```
data_train = data_train.reset_index(drop=True)
```

+ Code + Markdown

↑ ↓ 🗑

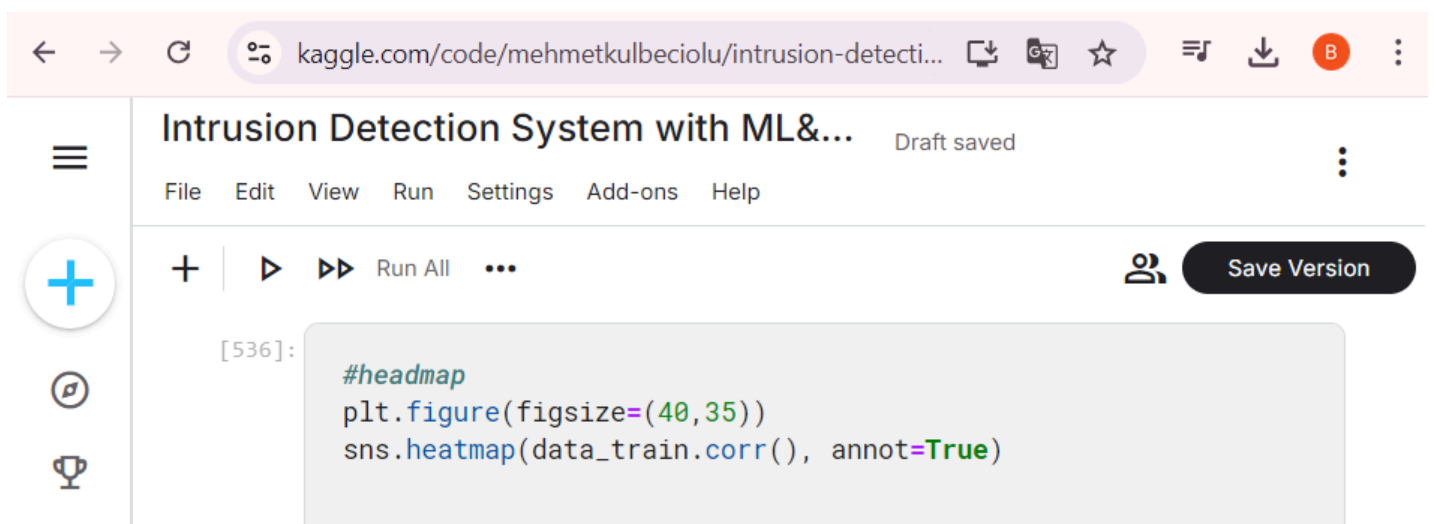
After this operation, I ran the `data_train.head()` cell again to confirm that the indices now started from zero.

### Step 3: Visualizing Data (Creating a Heatmap)

To analyze the data, I needed to use the Seaborn library. I imported Seaborn at the beginning of the notebook:

```
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
import seaborn as sns
from sklearn.preprocessing import RobustScaler
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn import svm
```

Then, I added a new cell to create a heatmap to visualize the correlations between the variables in the dataset, writing the following code:



# Intrusion Detection System with ML&...

Draft saved

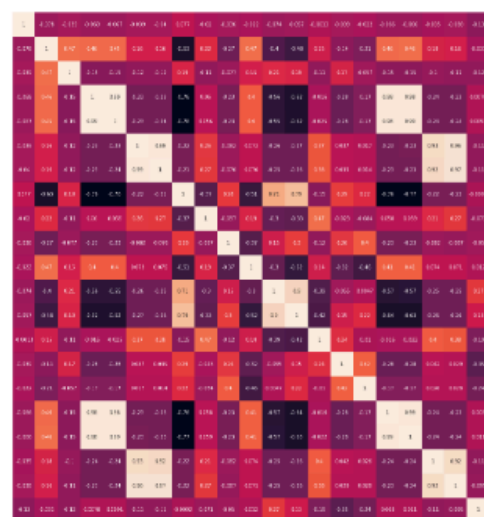
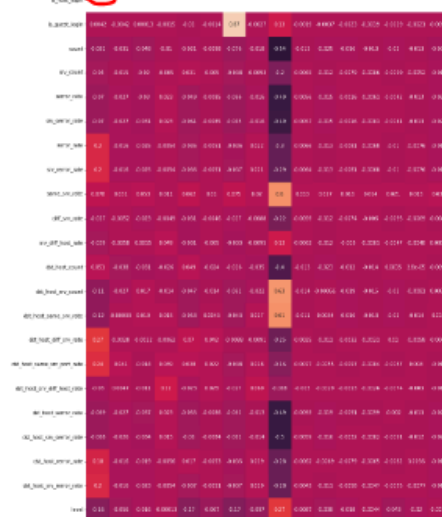
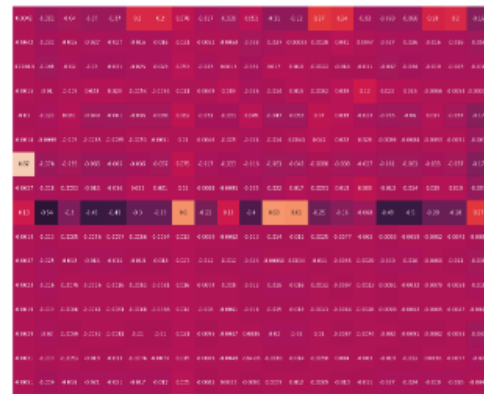
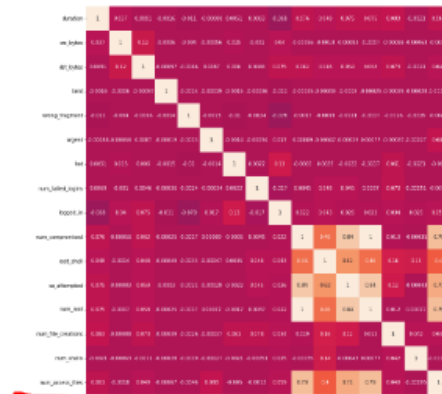
File Edit View Run Settings Add-ons Help

+ ▶ ▶▶ Run All ...



Save Version

[536... <AxesSubplot: >



This graph displayed the correlation between the columns in the dataset. Notably, the 'num\_outbound\_cmds' column appeared empty. If this column did not show any correlation, I decided to remove it from the dataset:

```
[537]: data_train['num_outbound_cmds'].describe()
```

```
[537... count      10000.0  
mean         0.0  
std          0.0  
min          0.0  
25%          0.0  
50%          0.0  
75%          0.0  
max          0.0  
Name: num_outbound_cmds, dtype: float64
```

```
[538]: data_train.drop('num_outbound_cmds', axis=1, inplace=True)
```

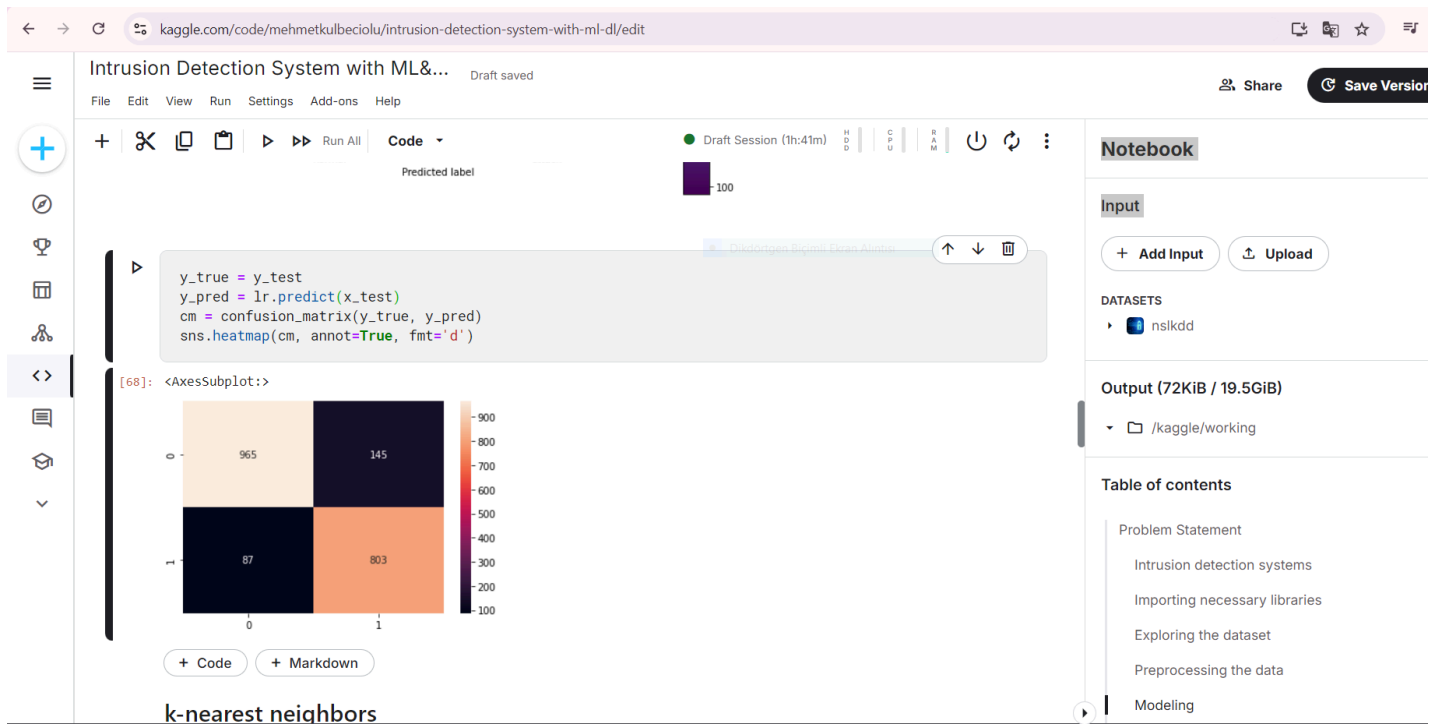
## Step 4: Working with Machine Learning Models



In the subsequent sections of the notebook, I began executing machine learning models. I started with the "logical regression" section. I ran the prepared cells one by one to obtain the results of various models, examining the performance of different algorithms.

In this notebook, the types of attacks in the dataset were transformed into a binary classification problem. This means there were two main categories: attacks and normal behaviors. The model outputs were classified as 0 (no attack) or 1 (attack).

I visualized the results by creating a confusion matrix in one of the cells. This matrix displayed the distribution of correct and incorrect predictions across different classes. My sample code was as follows:



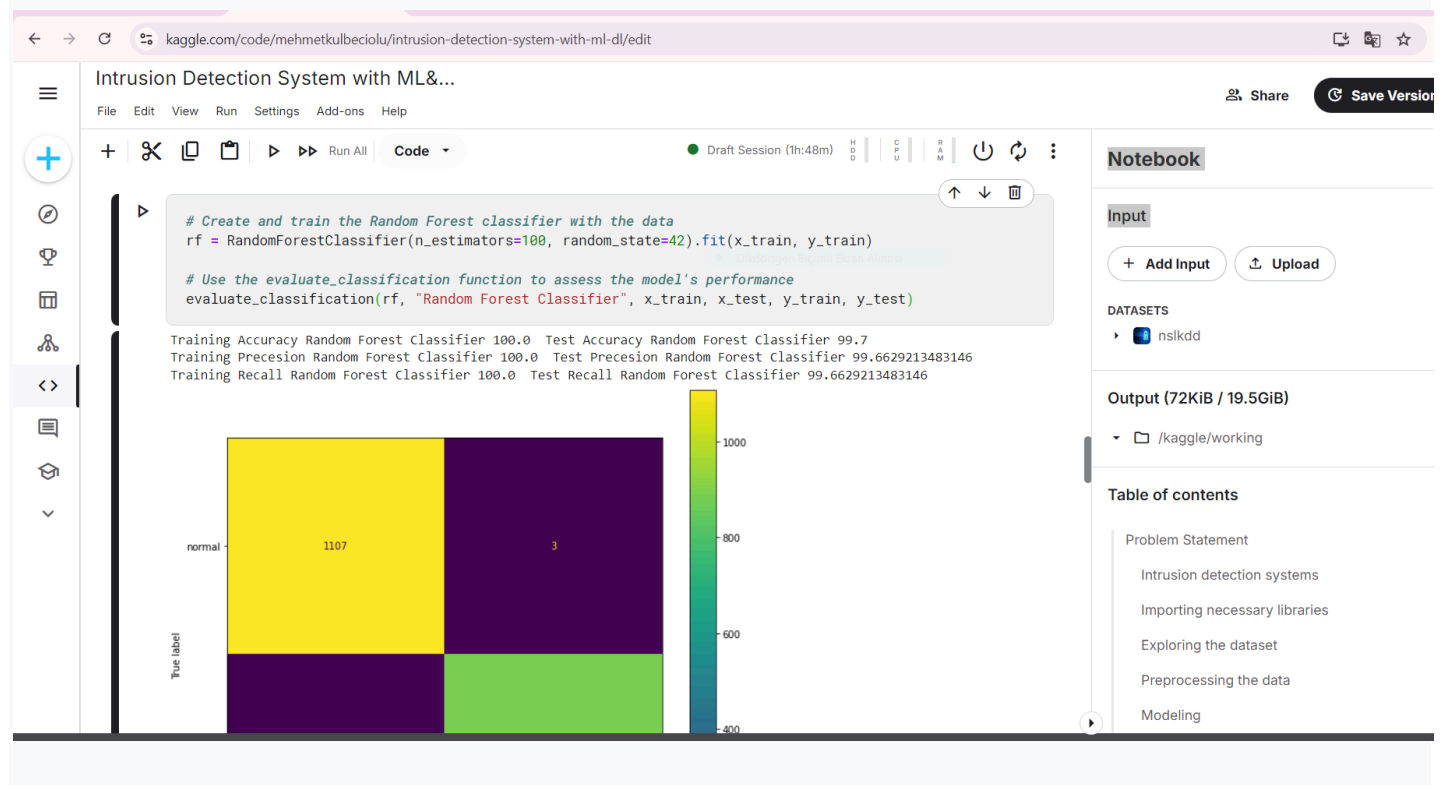
The matrix had four sections:

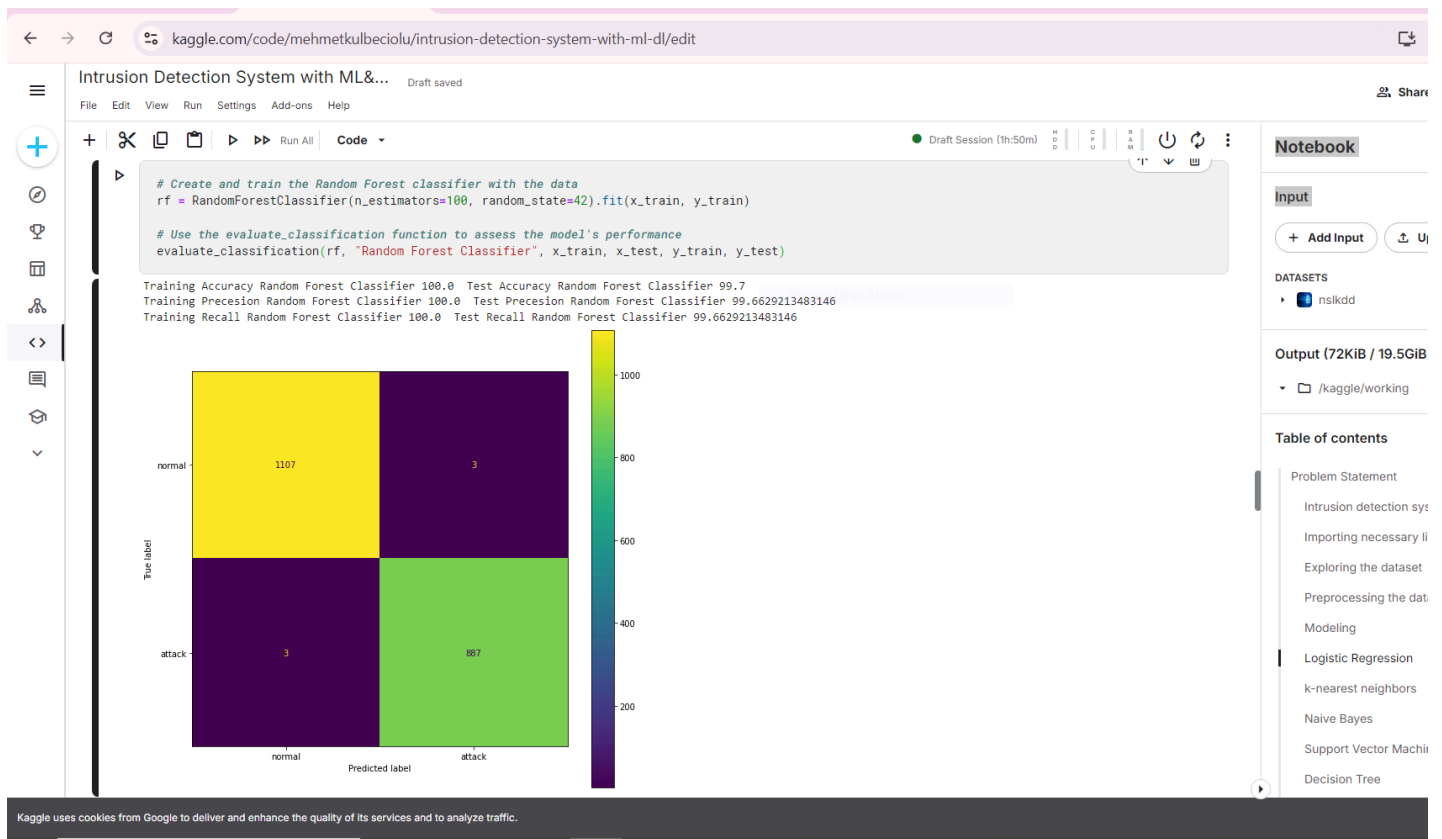
- Top Left: True Negatives (correctly predicted non-attack cases)

- Bottom Right: True Positives (correctly predicted attacks)
- Bottom Left: False Negatives (attacks incorrectly predicted as normal)
- Top Right: False Positives (normals incorrectly predicted as attacks)

## Step 5: Trying Additional Models (Optional)

I also tried out other models from the sklearn library in the notebook. I examined the results of these models and compared how different algorithms performed. For example, I used the Random Forest algorithm as follows:





## Conclusion

This laboratory work provided significant experience in developing an intrusion detection system using machine learning techniques with the NSL-KDD dataset. The results obtained will contribute to the development of a more reliable IDS by comparing the performance of different algorithms.