- **410447 *Muhammet Enes EMİR***
- **400000 *Furkan BÜLBÜL***

- Bilgisayar Güvenliğinin Temelleri Dönem Ödevinin Raporu
  - Proje Dizin Yapısı
    - Dizin ve Dosya Açıklamaları
  - Versiyon Kontrol Sistemi
  - Sınıflar ve Fonksiyonlar
    - `dataset.py`
      1. `read_csv_file(path: str) -> pd.DataFrame`
      2. `read_columns_from_file(col_file_path: str) -> list`
      3. `set_col_names(dataset: pd.DataFrame, col_names: list) -> pd.DataFrame`
      4. `prepare_dataset(path_dataset: str, col_name_path: str)`
      5. `get_dataset` (Property)
    - `kmeans.py`
      1. init
      2. `map_dataset(dataset: pd.DataFrame, flag: str = "attack_flag")`
      3. `map_attack(attack)`
      4. `make_map_attack(dataset: pd.DataFrame, col_name: str = "attack_map")`
      5. `prepare_training(dataset, dataset_test, flag="attack_flag", col_name="attack_map")`
      6. `kmean_training` (Property)
    - `receive.py`
      1. `init`
      2. `preprocess_packet(packet: dict, cols: list[str])`
      3. `detect_anomaly(processed_data, indeks)`
      4. `start_server()`
    - `send.py`
      1. `init`
      2. `send_packet(packet)`
      3. `main`
    - `utils.py`
      1. `load_json_file(json_path)`
      2. `get_data_from_json(key, json_data, default=None)`
    - `main.py`
      1. `attacks` Listesi
      2. **2. `main()` Fonksiyonu
      3. `KMeanAlogriths` Sınıfı
  - Projeyi Çalıştırma

Bilgisayar Güvenliğinin Temelleri Dönem Ödevinin Raporu

Proje Dizin Yapısı

Aşağıda, projenizin dizin yapısını ve içerdiği dosyaları gösteren "tree" komutunun çıktısı bulunmaktadır:

```
.
├── kmeans-model
│   ├── kmeans_model.pkl
│   └── model1.pkl
├── nsl-kdd
│   ├── column_names.txt
│   ├── KDDTest-21.arff
│   ├── KDDTest-21.txt
│   ├── KDDTest+.arff
│   ├── KDDTest+.txt
│   ├── KDDTrain+_20Percent.arff
│   ├── KDDTrain+_20Percent.txt
│   ├── KDDTrain+.arff
│   └── KDDTrain+.txt
├── README.md
└── src
    ├── config.json
    ├── dataset.py
    ├── kmeans.py
    ├── main.py
    │   ├── dataset.cpython-312.pyc
    │   └── utils.cpython-312.pyc
    ├── receive.py
    ├── requirements.txt
    ├── send.py
    └── utils.py

5 directories, 22 files
```

Dizin ve Dosya Açıklamaları

- **kmeans-model**: KMeans modelinin kaydedildiği dosyalar.
  - `kmeans_model.pkl` : Eğitimli KMeans modelinin dosyası.
  - `model1.pkl` : Modelin başka bir versiyonunun kaydedildiği dosya.
- **nsl-kdd**: NSL-KDD veri kümesinin bulunduğu klasör.
  - `column_names.txt` : Kolon isimlerinin bulunduğu dosya.
  - `.arff` ve `.txt` dosyaları: Eğitim ve test veri kümesi dosyaları.
- **src**: Projenin ana kaynak dosyalarının bulunduğu klasör.

- `config.json` : Yapılandırma dosyası, parametreler ve dosya yolları içerir.
- `dataset.py` : Veri kümesinin işlenmesi ve hazırlanmasıyla ilgili fonksiyonları içerir.
- `kmeans.py` : KMeans algoritmasının uygulandığı dosya.
- `main.py` : Projenin ana çalıştırılabilir dosyası, model eğitimi ve kaydedilmesi işlemlerini yapar.
- `receive.py` ve `send.py` : Veri alma ve gönderme işlemleriyle ilgili dosyalar.
- `requirements.txt` : Projede kullanılan Python kütüphanelerinin listesi.
- `utils.py` : Yardımcı fonksiyonlar ve JSON dosyalarını yüklemek gibi işlemleri içerir.

Bu dizin yapısı ve dosyalar, projenin çeşitli bileşenlerini organize eder ve her bir dosya belirli bir işlevi yerine getirir.

Versiyon Kontrol Sistemi

Verisyon kontrol sistemi olarak git kullanılmıştır.

```
ff73832 Merge branch 'refactor'
4d840e4 Fix some bug
1574a05 Update some prints
4534ba6 Refactor server class
ddf0247 Refactor send class
ea24711 Fixed some bugs
29a670c Remove code folder and move to src directory
c41d0ad Add requierement file for pip packages
d4948e6 Write send and receive for real time testing
243c499 Train kmean model and save it
d772c40 Update gitignore file
c87c90e Fix some bug
d4ff17a Write kmeans class
d5f74ad Add new paths to config file
1de0070 Create main and read dataset
4350c03 Fix some bug for dataset reading process
dd78607 Read path from json file
ec2eef6 Add a class for reading Pandas DataFrame files
cf68862 Train kmean model
0d3e668 Updated .gitignore files
1a4d8cd Updated files path
e889a09 Updated .gitignore files
b264aa5 Updated files path
5db242e Warnings eliminated
0b3fd26 Initial commit
```

Sınıflar ve Fonksiyonlar

Bu başlık altında her bir kod dosyasının içeriği, ilgili fonksiyonları ile birlikte ayrıntılı olarak açıklanmıştır.

`dataset.py`

`Dataset` sınıfı, veri setlerini yüklemek, sütun isimlerini belirlemek ve veri ön işleme işlemlerini kolaylaştırmak amacıyla tasarlanmıştır. Bu sınıf, NSL-KDD veri seti gibi yapılandırılmış dosyaları okuyarak Python ortamına yükler ve gerekli sütun isimlerini ekler. Böylece veri seti, analiz ve modelleme işlemleri için hazır hale getirilir.

1. `read_csv_file(path: str) -> pd.DataFrame`

Bu fonksiyon, belirtilen dosya yolundan bir CSV dosyasını okur ve bir `pandas.DataFrame` nesnesi olarak döndürür. Eğer dosya CSV formatında değilse veya bir hata oluşursa, program sonlandırılır.

Kullanım Örneği:

```
dataset = Dataset.read_csv_file("data/nsl_kdd.csv")
print(dataset.head())
```

Çıktı Örneği:

```
   duration  protocol_type  service  ...  num_outbound_cmds  is_host_login  is_guest_login
0         0  tcp           http     ...                  0              0               0
1         0  udp           private  ...                  0              0               0
```

2. `read_columns_from_file(col_file_path: str) -> list`

Belirtilen dosyadan sütun isimlerini okur ve bir liste olarak döndürür. Eğer dosya okunamazsa veya hata oluşursa, `None` döndürülür.

Kullanım Örneği:

```
column_names = Dataset.read_columns_from_file("data/column_names.txt")
print(column_names)
```

Çıktı Örneği:

```
['duration', 'protocol_type', 'service', 'flag', 'src_bytes', ...]
```

3. `set_col_names(dataset: pd.DataFrame, col_names: list) -> pd.DataFrame`

Bir veri setine sütun isimlerini atar ve yeni sütun isimleriyle birlikte veri setini döndürür. Eğer sütun isimleri `None` ise program sonlandırılır.

Kullanım Örneği:

```
dataset = Dataset.read_csv_file("data/nsl_kdd.csv")
column_names = Dataset.read_columns_from_file("data/column_names.txt")
dataset_with_columns = Dataset.set_col_names(dataset, column_names)
print(dataset_with_columns.columns)
```

Çıktı Örneği:

```
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes', ...], dtype='object')
```

4. `prepare_dataset(path_dataset: str, col_name_path: str)`

Veri setini yükler, sütun isimlerini okur ve sütun isimlerini veri setine ekleyerek işlem görmeye hazır bir veri seti döndürür.

Kullanım Örneği:

```
prepared_dataset = Dataset.prepare_dataset("data/nsl_kdd.csv", "data/column_names.txt")
print(prepared_dataset.head())
```

Çıktı Örneği:

```
   duration  protocol_type  service  ...  num_outbound_cmds  is_host_login  is_guest_login
0         0  tcp           http     ...                  0              0               0
1         0  udp           private  ...                  0              0               0
```

5. `get_dataset` (Property)

Sınıfın içinde tanımlı olan `dataset` özelliğini döndürmek için bir getter fonksiyonudur. Ancak, mevcut kodda bu özellik tanımlı değildir ve bu nedenle bu metodun işlevselliği eksiktir.

Kullanım Örneği:

```
# Eğer `self.dataset` tanımlanmış olsaydı:
dataset_instance = Dataset()
print(dataset_instance.get_dataset)
```

Çıktı Örneği:

```
<Mevcut veri seti>
```

`kmeans.py`

`KMeanAlogriths` sınıfı, K-Means algoritmasını kullanarak NSL-KDD veri seti üzerinde anomali tespiti yapmak amacıyla geliştirilmiştir. Bu sınıf, veri kümesini işlemeye, saldırı türlerini etiketlemeye, modeli eğitmeye ve sonuçları

kaydetmeye olanak tanır. Ayrıca, sınıf K-Means algoritmasını hem ikili (normal/saldırı) hem de çok sınıflı (saldırı türleri) olarak uygular.

1. init

Sınıfın kurucu metodudur. Gerekli parametreleri alarak sınıfın özelliklerini başlatır:

- `dataset` : Eğitim veri kümesi.
- `dataset_test` : Test veri kümesi.
- `attacks` : Saldırı türlerini içeren liste.
- `test_size` : Eğitim ve doğrulama veri kümesi oranı.
- `n_clusters` : K-Means algoritması için küme sayısı.
- `path_model_save` : Eğitilen modelin kaydedileceği yol.
- `numeric_features` : Kullanılacak sayısal özellikler.

Kullanım Örneği:

```
kmeans = KMeanAlogriths(dataset=train_data, dataset_test=test_data)
```

2. `map_dataset(dataset: pd.DataFrame, flag: str = "attack_flag")`

Veri kümesindeki saldırı etiketlerini (normal/saldırı) ikili değerlere (`0: normal` , `1: saldırı`) dönüştürür ve yeni bir sütun ekler.

Kullanım Örneği:

```
mapped_data = kmeans.map_dataset(dataset=train_data)
print(mapped_data.head())
```

Çıktı Örneği:

```
   duration  src_bytes  dst_bytes  attack  attack_flag
0         0        491          0  normal            0
1         0          0          0  attack            1
```

3. `map_attack(attack)`

Bir saldırı türünü, `attacks` listesine göre sayısal bir değere eşler. Eğer saldırı listede yoksa `0` döner.

Kullanım Örneği:

```
attack_type = kmeans.map_attack("neptune")
print(attack_type)
```

Çıktı Örneği:

```
1
```

4. `make_map_attack(dataset: pd.DataFrame, col_name: str = "attack_map")`

Veri kümesindeki saldırı türlerini sayısal değerlere eşleyerek yeni bir sütun ekler.

Kullanım Örneği:

```
mapped_attacks = kmeans.make_map_attack(dataset=train_data)
print(mapped_attacks.head())
```

Çıktı Örneği:

```
   duration  src_bytes  dst_bytes  attack  attack_map
0         0        491          0  normal           0
1         0          0          0  neptune          1
```

5. `prepare_training(dataset, dataset_test, flag="attack_flag", col_name="attack_map")`

Eğitim ve test veri kümelerini saldırı türlerine göre etiketler ve sadece sayısal özellikleri döndürür.

Kullanım Örneği:

```
numeric_features, train_data, test_data = kmeans.prepare_training(train_data, test_data)
print(numeric_features.head())
```

Çıktı Örneği:

```
   duration  src_bytes  dst_bytes
0         0        491          0
1         0          0          0
```

6. `kmean_training` (Property)

K-Means modelini eğitir ve doğrulama veri kümesindeki doğruluk oranını hesaplar. Eğitilen modeli belirtilen dosya yoluna kaydeder.

Kullanım Örneği:

```
kmeans.kmean_training
```

Çıktı Örneği:

```
0.85  # Çok sınıflı doğruluk
0.90  # İkili doğruluk
```

`receive.py`

`ReceivePacket` sınıfı, TCP üzerinden alınan veriyi işlemek ve K-Means modelini kullanarak anomali tespiti yapmak için tasarlanmıştır. Bu sınıf, bir TCP sunucusu başlatarak veri paketlerini alır, veriyi işler ve K-Means modelini kullanarak bu verinin anomali olup olmadığını kontrol eder.

1. `init`

Sınıfın kurucu metodudur. Aşağıdaki parametreleri alır:

- `kmean_model` : K-Means modelini yükler.
- `cols` : Verinin hangi sütunlarının kullanılacağını belirler.
- `address` : Sunucunun dinleyeceği IP adresi.
- `port` : Sunucunun dinleyeceği port.

Kullanım Örneği:

```
receive_packet = ReceivePacket(kmean_model=kmeans_model, cols=["duration", "src_bytes", "dst_bytes"])
```

2. `preprocess_packet(packet: dict, cols: list[str])`

Veri paketini işleyerek K-Means modeline uygun hale getirir. Bu işlem, veri kümesindeki eksik sütunları doldurmayı, kategorik verileri sayısal verilere dönüştürmeyi ve veri çerçevesini modelin gereksinimlerine uygun hale getirmeyi içerir.

Kullanım Örneği:

```
processed_data = receive_packet.preprocess_packet(packet, ["duration", "src_bytes", "dst_bytes"])
```

3. `detect_anomaly(processed_data, indeks)`

Verilen işlenmiş veriyi K-Means modeline besler ve anomali olup olmadığını kontrol eder. Modelin tahminlerine göre, verinin kümeler arasındaki mesafeyi hesaplar. Eğer mesafe belirli bir eşikten büyükse, veriyi anomali olarak işaretler.

Kullanım Örneği:

```
receive_packet.detect_anomaly(processed_data, indeks=1)
```

Çıktı Örneği:

```
Detect Anomali : 1: 2905, 50, 2000, 1500
Normal         : 2: 1200, 45, 1900, 1000
```

4. `start_server()`

Bir TCP sunucusu başlatır, bağlantıları dinler ve her gelen bağlantı için veriyi alır, işler ve anomali tespiti yapar. Bu fonksiyon sürekli çalışır ve gelen verileri sürekli olarak işler.

Kullanım Örneği:

```
receive_packet.start_server()
```

Sunucu, belirtilen IP adresi ve port üzerinden gelen verileri bekler.

`send.py`

`SendPacket` sınıfı, veriyi belirtilen bir IP adresi ve port üzerinden TCP ile sunucuya gönderen bir yapıdır. Bu sınıf, veriyi JSON formatına dönüştürüp sunucuya iletmek için kullanılır.

1. `init`

Sınıfın kurucu metodudur. Aşağıdaki parametreleri alır:

- `address` : Verinin gönderileceği IP adresi.
- `port` : Verinin gönderileceği port.

Kullanım Örneği:

```
send_packet = SendPacket(address="192.168.1.100", port=9999)
```

2. `send_packet(packet)`

Bu fonksiyon, verilen veri paketini TCP üzerinden belirtilen adrese ve porta gönderir.

1. İlk olarak bir TCP istemcisi (`socket`) oluşturur.
2. Veri paketini JSON formatına dönüştürür.
3. JSON verisini TCP istemcisine gönderir.
4. Veriyi gönderdikten sonra soketi kapatır.

Kullanım Örneği:

```
send_packet.send_packet(packet)
```

Çıktı Örneği:

```
Sending Packet 0: {'duration': 120, 'src_bytes': 5000, 'dst_bytes': 1500}
```

3. `main`

Ana fonksiyon, komut satırı argümanları alır ve veriyi göndermek için `SendPacket` sınıfını kullanır. Aşağıdaki adımları takip eder:

1. **Argümanlar**: Kullanıcıdan gönderilecek sütunları (`sending_cols`), kaç adet veri gönderileceğini (`send_count`), adres ve port bilgilerini alır.
2. **Veri Yükleme**: `Utils` sınıfı kullanılarak JSON yapılandırma dosyasından veri yüklenir.
3. **Veri Hazırlama**: `Dataset` sınıfı ile belirtilen test veri kümesi ve sütun isimleri ile dataset hazırlanır.
4. **Veri Gönderme**: `send_packet` fonksiyonu ile her bir veri satırı belirtilen sunucuya gönderilir.

Kullanım Örneği:

```
python send.py --sending_cols duration src_bytes dst_bytes --send_count 10000 --address 127.0.0.1 --pc
```

`utils.py`

`Utils` sınıfı, JSON dosyalarını yüklemek ve bu dosyalardan verileri almak için yardımcı işlevler sağlar. Bu sınıf, yapılandırma dosyalarından veri çekmek ve bu verileri kullanarak programın diğer bölümlerinde veri sağlamak için kullanılır.

1. `load_json_file(json_path)`

Bu fonksiyon, belirtilen dosya yolundaki JSON dosyasını yükler ve içeriğini Python veri yapısına (genellikle bir sözlük) dönüştürür.

- **Girdi**: `json_path` (JSON dosyasının yolu).
- **Çıktı**: JSON dosyasının içeriği (Python veri yapısı olarak).
- **Hata Durumunda**:
  - Dosya bulunamazsa: `FileNotFoundError` hatası.
  - Dosya JSON formatında değilse: `ValueError` hatası.
  - JSON dosyası geçerli değilse: `ValueError` hatası.

Kullanım Örneği:

```
utils = Utils()
config_data = utils.load_json_file("config.json")
```

Çıktı Örneği:

```
{
    "paths": {
        "dataset_test": "path/to/test/data",
        "columns": "path/to/columns/file"
    }
}
```

2. `get_data_from_json(key, json_data, default=None)`

Bu fonksiyon, JSON verisinde belirtilen anahtarın değerini döndürür. Anahtarlar, nokta (.) ile ayrılmış şekilde belirtilen bir dizini takip eder. Eğer anahtar bulunamazsa, varsayılan bir değer döndürülebilir.

- **Girdi**:
  - `key` : Anahtarın yolu (örneğin, `"paths.dataset_test"`).
  - `json_data` : JSON verisi (genellikle `load_json_file` fonksiyonuyla yüklenmiş veri).
  - `default` : Anahtar bulunamazsa döndürülecek varsayılan değer (isteğe bağlı).
- **Çıktı**: Anahtarın değeri.
- **Hata Durumunda**: Anahtar bulunamazsa, varsayılan değer yoksa `KeyError` hatası fırlatılır.

Kullanım Örneği:

```
dataset_test_path = utils.get_data_from_json("paths.dataset_test", config_data)
```

Çıktı Örneği:

```
"path/to/test/data"
```

`main.py`

Bu dosya, veri kümesi üzerinde bir KMeans kümeleme algoritması çalıştırarak model eğitmek ve kaydetmek için bir ana fonksiyon içerir. `main.py` dosyasının temel amacı, yapılandırma dosyasından gerekli parametreleri alıp, veri kümesini hazırlayıp, KMeans modelini eğitmek ve kaydetmektir.

1. `attacks` Listesi

Bu liste, farklı saldırı türlerinin gruplandığı bir yapıdır. Her bir grup, belirli bir saldırı türüne ait adları içerir. Bu liste, KMeans algoritmasında etiket olarak kullanılacak saldırı türlerini temsil eder.

**2. `main()` Fonksiyonu

Ana fonksiyon, uygulamanın çalıştırılması için gereken temel işlemleri içerir. Şu adımları takip eder:

1. `_utils = Utils()` : `Utils` sınıfını başlatır.
2. **Yapılandırma Dosyasını Yükler**: `config.json` dosyasındaki verileri yükler.
   - `path_dataset` : Eğitim veri kümesinin yolu.
   - `path_dataset_test` : Test veri kümesinin yolu.
   - `path_column_names` : Kolon isimlerinin bulunduğu dosyanın yolu.
   - `path_kmeans_save` : KMeans modelinin kaydedileceği dizinin yolu.
3. **Veri Kümesini Yükler**:
   - `dataset` : Eğitim verisi.
   - `dataset_test` : Test verisi.
4. **KMeans Modeli İçin Parametreler Tanımlanır**:
   - `dataset` : Eğitim verisi.

- `dataset_test` : Test verisi.
- `attacks` : Saldırı türlerinin listesi.
- `test_size` : Eğitim ve test verisi oranı (burada `%60 eğitim,` %40 test).
- `n_clusters` : KMeans algoritmasındaki küme sayısı (2 küme).
- `path_model_save` : Modelin kaydedileceği dosya yolu.
- `numeric_features` : KMeans algoritmasında kullanılacak sayısal özellikler (örneğin, `duration` , `src_bytes` , `dst_bytes`).

5. **KMeans Modeli Eğitilir ve Kaydedilir**:
   - `kmean.kmean_training` : KMeans modelinin eğitilmesi.

3. `KMeanAlogriths` Sınıfı

Bu sınıf, KMeans algoritmasını eğitmek için kullanılan sınıftır. `main.py` dosyasında bu sınıfın örneği oluşturulmuş ve `kmean_training` fonksiyonu çağrılmıştır.

Projeyi Çalıştırma

İlk olarak main.py çalıştırılır:

```
python3 main.py
```

Modelin test işlemleri için send ve receive çalıştırılır. İlk receive başlatılmalıdır.

```
python3 receive.py --address 0.0.0.0 --port 8888 --cols duration src_bytes dst_bytes --model ../kmeans
```

```
python3 send.py --sending_cols duration src_bytes dst_bytes --send_count 10000 --address 127.0.0.1 --p
```