Siber Güvenlik İçin Veri Madenciliği
Ödev1 Raporu

Hazırlayanlar: 200504016- Betül Yaşar

200504025- Muzaffer Birinci

Malware Traffic Analysis Knowledge Dataset 2019 (MTA-KDD'19) Analizi

Malware Traffic Analysis Knowledge Dataset 2019 (MTA-KDD'19), makine öğrenimi tabanlı zararlı yazılım trafiği analiz algoritmalarını eğitmek ve değerlendirmek için özel olarak hazırlanmış ve güncellenmiş edilmiş bir veri setidir.

1. Veri Setinin Oluşturulması:

- Veri seti, çevrimiçi olarak erişilebilen en büyük ağ trafiği yakalama veritabanlarından başlayarak oluşturulmuştur.
- Bu veri seti, geniş uygulamalı bir dizi özellik içerir ve ardından gürültüyü kaldırmak, eksik verileri ele almak ve boyutunu olabildiğince küçük tutmak için temizlenmiş ve önceden işlenmiştir.

2. Özellikleri:

- MTA-KDD'19, makine öğrenimi algoritmalarına özellikle hitap etmekle birlikte, herhangi bir belirli uygulama tarafından önyargılı değildir.
- Veri seti, malware analizine dair 34 özellik içermektedir. Bu özellikler, malware'lerin davranışlarına, etkileşimlerine ve diğer karakteristik özelliklerine geniş bir bakış sunar.

3. Otomatik Güncellenme:

- MTA-KDD'19'un oluşturulma süreci otomatik olarak çalışabilir. Bu, veri setini güncel tutmak için önemlidir. Bu otomatik süreç, veri setinin her zaman en son zararlı yazılım trafiği özelliklerini ve trendlerini yansıtmasını sağlar.

4. Uygulama Alanları:

- Bu veri seti, makine öğrenimi tabanlı zararlı yazılım trafiği analiz algoritmalarını eğitmek ve değerlendirmek için idealdir.
- Ayrıca, ağ güvenliği uzmanları ve araştırmacıları için de değerli bir kaynaktır.

Kod Analizi

Verinin Okunması:

```
# CSV dosyasını okuyalım
df = pd.read_csv("datasetMalware33featues.csv")
#Veriler doğru bir şekilde çekilmiş mi kontrol amacıyla ilk 5 sütunu çekelim
df.head()
```

- **pd.read_csv** fonksiyonu kullanılarak veri seti okundu ve **data** adında bir DataFrame'e yüklendi.
- **data.head()** fonksiyonu, veri setinin ilk 5 satırını yazdırmak için kullanıldı. Bu, veri setinin doğru bir şekilde yüklendiğini kontrol etmek için yapılır.

UDPoverIP	MaxLen	MinLen	...	NumPorts	FlowLEN	FlowLENrx	repeated_pkts_ratio	NumCon	NumIPdst	Start_flow	DeltaTimeFlow	HTTPpkts	label
-0.156991	-0.009901	-1.028609	...	-0.889691	0.479515	0.284428	0.862578	5.253913	5.142702	0.562400	-0.052248	1.487787	1.0
-0.156991	0.722669	0.628640	...	0.293799	1.301840	-0.036629	-0.908981	-0.190341	-0.194497	0.562400	-0.315281	1.786307	1.0
-0.156991	-0.172436	0.628640	...	-0.889691	-0.452523	-0.049479	-0.671509	-0.190341	-0.194497	0.562400	-1.197897	0.583908	1.0
-0.156991	0.722669	0.628640	...	0.823293	1.416370	-0.042111	-1.626711	-0.190341	-0.194497	0.562400	-0.097754	1.805515	1.0
6.836627	0.485384	0.628640	...	1.116241	0.868445	0.678529	0.902818	5.253913	5.142781	0.563213	0.885878	1.614688	1.0

- Veri setinin ilk 5 satırında, 34 sütun bulunmaktadır. Bunlardan 33'ü özellikleri temsil ederken, son sütun olan "Label" sütunu etiket bilgisini içerir.
- Bu adım, veri setinin başarılı bir şekilde yüklenip yüklenmediğini kontrol etmek için önemlidir. Eğer veri setinin bir bölümü hatalı veya eksik yüklendiyse, bu hemen fark edilir.

Eksik Verilerin Doldurulması

```
# Eksik verileri (NaN değerlerini) sütun ortalamalarıyla dolduralım
df.fillna(df.mean(), inplace=True)

# Eksik veri kalmış mı kontrol edelim
missing_values = df.isnull().sum()
missing_values
```

Eksik değerler, veri bilimi ve makine öğrenimi projelerinde sıkça karşılaşılan bir sorundur. Eksik değerler, veri toplama sırasında oluşabileceği gibi, bazı özelliklerin belirli gözlemler için anlamsız olması nedeniyle de oluşabilir. Eksik değerlerin doğru bir şekilde ele alınmaması, analizin yanıltıcı veya hatalı sonuçlar vermesine neden olabilir.

Bu projede, eksik değerlerin genel dağılımı bozmamak için ilgili sütunun ortalamasıyla doldurulmasına karar verildi. Ortalama değer, eksik değerlerin yerini doldurmak için sıkça kullanılan bir yöntemdir çünkü bu yöntem, eksik değerlerin genel dağılıma minimal etki yapmasını sağlar.

Eksik değerlerin doldurulmasının ana nedenleri şunlardır:

- **Model Performansı:** Makine öğrenimi algoritmalarının çoğu eksik değerleri kabul etmez. Bu nedenle, eksik değerleri doldurmak modelin eğitilmesi için gereklidir.
- **Veri İntegrasyonu:** Eksik değerler, veri setinin diğer kısımlarıyla tutarlı bir şekilde entegre olmadan analiz yapmayı zorlaştırır.
- **Analiz Doğruluğu:** Eksik değerlerin doldurulmamış olması, istatistiksel testlerin ve makine öğrenimi model tahminlerinin doğruluğunu etkileyebilir.

Kontrol:
Veri setinde hala eksik değer olup olmadığını kontrol etmek için yukarıda verilen kodu kullanabiliriz

Eğer sonuç 0 ise, bu veri setinde hiç eksik değer kalmadığı anlamına gelir.

Bu adım, veri setinin eksik değerlerini doğru bir şekilde ele alarak analizin doğruluğunu ve güvenilirliğini artırmak için önemlidir.

Normalizasyon İşlemi

```
#Normalizasyon işlemi
scaler = MinMaxScaler()

# 'label' sütunu hariç tüm sütunları ölçeklendirelim
df_scaled = df.copy()
df_scaled[df_scaled.columns[:-1]] = scaler.fit_transform(df_scaled[df_scaled.columns[:-1]])

# Ölçeklendirilmiş verinin ilk beş satırını gösterelim
df_scaled.head()
```

	FinFlagDist	SynFlagDist	RstFlagDist	PshFlagDist	AckFlagDist	DNSoverIP	TCPoverIP	UDPoverIP	MaxLen	MinLen	...	NumPorts	FlowLEN	FlowLE
0	0.448414	0.613970	0.000000	0.503470	0.629803	0.0	1.000000e+00	0.0	0.178426	0.417328	...	0.000000	0.471063	0.618
1	0.000000	0.414872	0.000000	0.719140	0.639137	0.0	1.000000e+00	0.0	0.266543	0.640945	...	0.412948	0.661491	0.614
2	0.000000	0.286645	0.000000	0.222305	0.240782	0.0	1.000000e+00	0.0	0.158876	0.640945	...	0.000000	0.255227	0.614
3	0.448414	0.491773	0.000000	0.738527	0.667009	0.0	1.000000e+00	0.0	0.266543	0.640945	...	0.597702	0.688014	0.614
4	0.786739	0.788029	0.990069	0.561153	0.706934	1.0	5.273559e-14	1.0	0.238001	0.640945	...	0.699919	0.561128	0.622

5 rows × 34 columns

Veri setlerindeki özelliklerin değer aralıkları farklı olabilir. Örneğin, bir özellik 0 ile 1 arasında değerler alırken, başka bir özellik 0 ile 1000 arasında değerler alabilir. Bu tür özelliklerin farklı değer aralıklarına sahip olması, makine öğrenimi algoritmalarının performansını olumsuz etkileyebilir.

Bu nedenle, tüm özelliklerin aynı ölçeğe sahip olmasını sağlamak için normalizasyon işlemi yapılır. Bu projede, **MinMaxScaler** yöntemi kullanılarak normalizasyon gerçekleştirildi.

Normalizasyon işlemi sırasında "Label" etiketini dahil etmememizin birkaç önemli nedeni bulunmaktadır:

- **Farklı Doğa:** "Label" etiketi, kategorik bir değişkeni (genellikle sınıflandırma için hedef değişkeni) temsil ederken, diğer özellikler sürekli değişkenlerdir. Bu iki tür değişkenin doğası farklıdır. Normalizasyon, genellikle sürekli değişkenlerde değerlerin ölçeğini ayarlamak için kullanılır. Kategorik bir değişkenin bu sürece dahil edilmesi, bu değişkenin anlamını ve kullanımını bozar.
- **Veri Bozulması:** "Label" etiketini normalizasyon işlemine dahil edersek, bu etiketlerin orijinal değerleri kaybolabilir. Özellikle MinMaxScaler gibi yöntemlerle yapılan normalizasyonda, 0 ve 1 arasında yeni değerlere dönüşebilirler. Bu, etiketlerin anlamlarını kaybetmelerine ve model eğitimi sırasında yanıltıcı olmalarına neden olabilir.

- **Model Eğitimi:** Makine öğrenimi modeli eğitimi sırasında, etiketler (hedef değişken) genellikle ayrı bir değişken olarak ele alınır ve özellik mühendisliği veya ölçekleme işlemlerinden muaf tutulur. Bu, modelin doğru bir şekilde eğitilmesi için gereklidir.

Sonuç olarak, normalizasyon işlemi sırasında "Label" etiketini dahil etmemek, veri setinin doğru bir şekilde işlenmesini ve makine öğrenimi modelinin etkili bir şekilde eğitilmesini sağlar. Bu yaklaşım, veri bilimi projelerinde yaygın bir uygulamadır.

İstatistiksel Analiz

```
# Veri setinin istatistiksel özetini alalım
statistical_summary = df_scaled.describe(percentiles=[.25, .5, .75])

# İstenen istatistiksel bilgileri gösterelim
statistical_summary.transpose()[['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max']]
```

Veri setinin özet istatistiklerini elde etmek için **describe** fonksiyonunu kullandık. Bu fonksiyon, veri setindeki her sütun için temel istatistiksel değerleri sağlar.

- **count**: Tüm öznitelikler için aynı sayıda veri noktası var, yani hiç eksik veri yok. Bu, analiz yapılırken kolaylık sağlar.
- **mean**: Ortalama değerler, her bir öznitelik için verilerin merkezi eğilimini gösterir. Örneğin, "PshFlagDist" için ortalama değer düşükken, "TCPOverIP" için ortalama değer nispeten daha yüksektir.
- **std**: Standart sapma değerleri, verilerin ne kadar yayıldığını gösterir. Örneğin, "PshFlagDist" için standart sapma değeri düşükken, "UDPOverIP" için bu değer yüksektir. Bu, "UDPOverIP" değerlerinin daha dağılmış olduğunu gösterir.
- **min ve max**: En düşük ve en yüksek değerler öznitelikler için veri aralığını gösterir. Birçok öznitelik için minimum değer 0 iken, maksimum değer 1'dir. Bu, bazı özniteliklerin normalleştirilmiş olabileceğini gösterir.
- **%25, %50 ve %75**: Bu değerler, öznitelikler için yüzdelik dilimleri (çeyreklikler) gösterir. Örneğin, "AvgLen" özniteliği için medyan (%50) değeri 0.2759'dur. Bu, verilerin yarısının bu değerin altında ve yarısının üstünde olduğunu gösterir.

Özetle, bu istatistikler veri setinin genel dağılımı, merkezi eğilimi ve yayılımı hakkında önemli bilgiler sağlar. Bu bilgiler, model eğitimi ya da diğer analizler için verinin önceden işlenmesi aşamasında rehberlik edebilir. Özellikle standart sapma, medyan gibi değerler, verinin ne kadar homojen ya da heterojen olduğunu gösterir ve bu da model seçimi veya özellik mühendisliği için önemli olabilir.

PCA (Ana Bileşenler Analizi)

PCA, çok boyutlu verilerin boyutunu azaltmak için kullanılan bir yöntemdir. Bu, özellikle büyük veri setlerinde, veri setinin boyutunu azaltarak hesaplama süresini kısaltmak ve görselleştirmeyi kolaylaştırmak için yapılır.

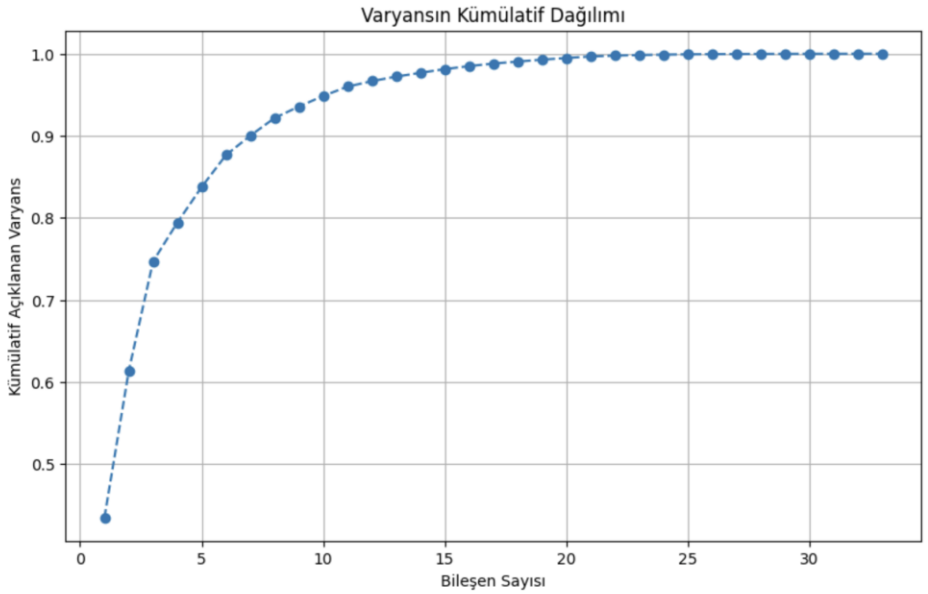
Bu projede, PCA'nın ilk adımı olarak optimal bileşen sayısını belirlemek için PCA uygulandı. Optimal bileşen sayısını belirlemek, PCA'nın etkili bir şekilde uygulanabilmesi için kritiktir. Eğer çok fazla bileşen seçilirse, boyut indirgeme etkili olmayabilir. Eğer çok az bileşen seçilirse, verinin önemli bilgileri kaybolabilir.

```
# 'label' sütunu hariç veriyi PCA için hazırlayalım
X = df_scaled.drop('label', axis=1)

# Tüm bileşenler için PCA uygulayalım
pca = PCA().fit(X)

# Kümülatif varyansı hesaplayalım
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
```

```
# Kümülatif varyans grafiğini çizelim
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_variance)+1), cumulative_variance, marker='o', linestyle='--')
plt.title("Varyansın Kümülatif Dağılımı")
plt.xlabel("Bileşen Sayısı")
plt.ylabel("Kümülatif Açıklanan Varyans")
plt.grid(True)
plt.show()
```




```
# Veriyi %95 varyansı koruyacak şekilde bileşen sayısıyla PCA uygulayalım
pca_95 = PCA(n_components=0.95)
X_pca = pca_95.fit_transform(X)
# Yeni bileşen sayısını gösterelim
num_components = X_pca.shape[1]
num_components
```

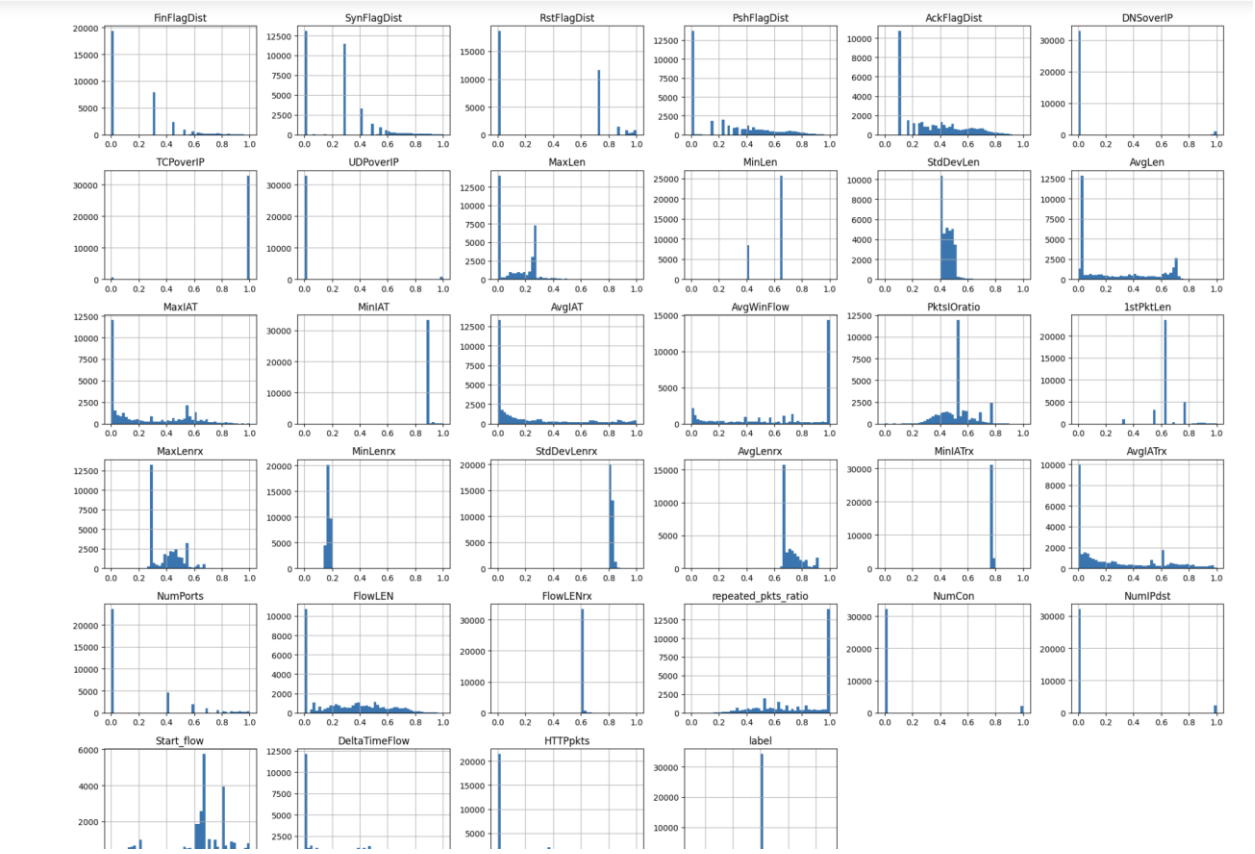
```
11
```

Optimal bileşen sayısını belirlemek için, PCA'nın açıkladığı varyansın kumulatif dağılımını inceledik. Bu, her bir bileşenin veri setindeki toplam varyansın ne kadarını açıkladığını gösterir. Genel olarak, toplam varyansın %95'ini veya %99'unu açıklayan bileşen sayısını seçmek yaygın bir uygulamadır.

Bu projede, grafikteki dirsek noktasını bulmak için kumulatif dağılım grafiği kullanıldı. Dirsek noktası, eklenen her yeni bileşenin açıkladığı varyansın azaldığı noktadır. Bu nokta, optimal bileşen sayısını belirlemek için bir kriter olarak kullanılabilir.

Bu adım, veri setinin boyutunu etkili bir şekilde azaltarak, makine öğrenimi modelinin daha hızlı ve daha doğru bir şekilde eğitilmesini sağlamak için önemlidir.

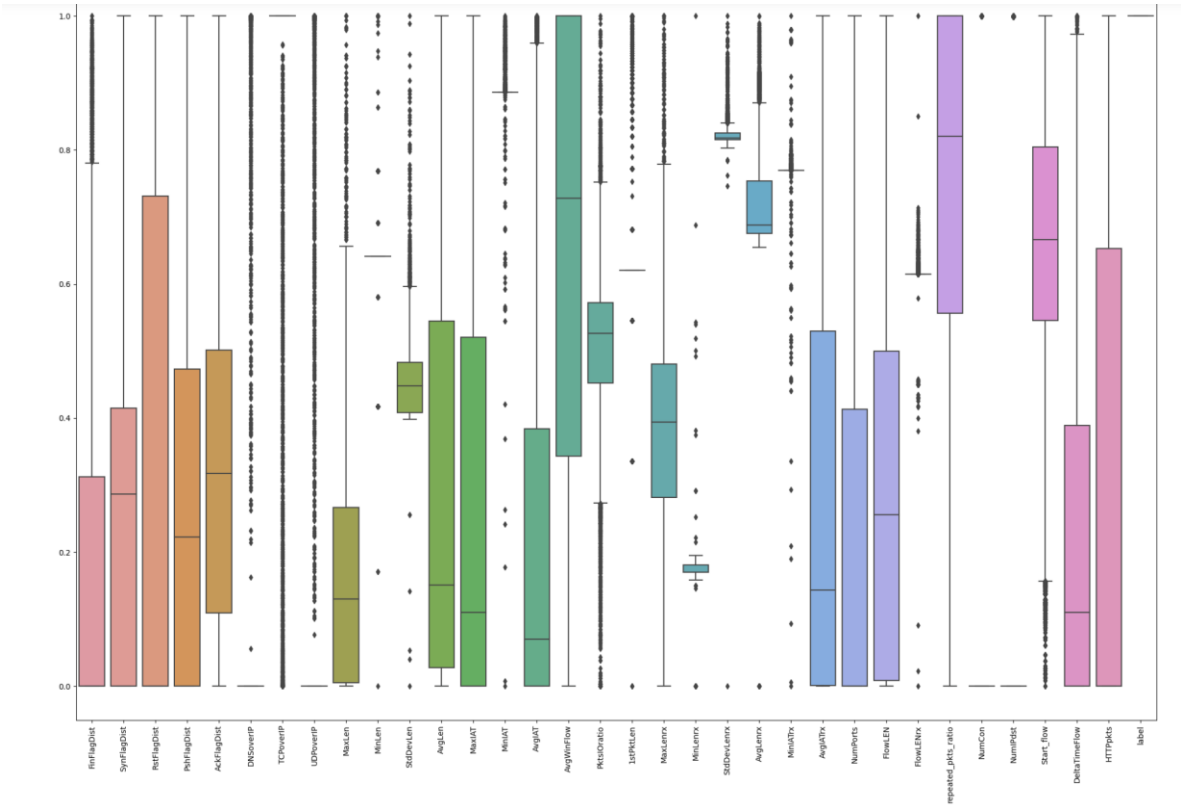
Veri Setinin Histogram Grafiklerinin Yorumlanması



- **FinFlagDist**: Çoğunlukla 0 değerine sahip, bu da çoğu örneğin Fin bayrağını kullanmadığını gösteriyor. Ancak 1 değeri de var, bu da bazı örneklerin Fin bayrağını kullandığını gösteriyor.
- **SynFlagDist**: İki ana zirve (0 ve 1) ile ikili bir dağılım gösteriyor. Bu, SYN bayrağının ya aktif ya da pasif olduğunu gösterir.
- **RstFlagDist**: Çoğunluğunun 0 değerine sahip olması, RST bayrağının çoğunlukla kullanılmadığını gösteriyor.
- **PshFlagDist**: Çoğunlukla 0 değeri etrafında yoğunlaşma var. Bu, PSH bayrağının nadiren kullanıldığına işaret ediyor.
- **AckFlagDist**: İki belirgin zirve var, bu da ACK bayrağının ya aktif ya da pasif olduğunu gösteriyor.
- **DNSoverIP**: Yoğunluk, 0 değeri etrafında. Bu, DNS sorgularının sıkça IP üzerinden yapılmadığını gösteriyor.
- **TCPoverIP** ve **UDPoverIP**: İkisi de benzer ikili dağılımlar gösteriyor. Bu, TCP ve UDP protokollerinin ya kullanıldığı ya da kullanılmadığına işaret ediyor.
- **MaxLen** ve **MinLen**: Maksimum ve minimum uzunluklar için dağılımlar, birkaç örneğin belirgin bir uzunluğa sahip olduğunu gösteriyor.
- **StdDevLen**, **AvgLen**, **MaxLenX**, **MinLenX**, ve **AvgLenX**: Bu özellikler, paket uzunlukları ile ilgili. Çoğunlukla belirli değerler etrafında yoğunlaşmaları, bu değerlerin sıkça rastlandığını gösteriyor.

- **MaxIAT**, **MinIAT**, **AvgIAT**, **AvgIATX**, **MinIATX**: Bu özellikler, paketler arası zaman (Inter-Arrival Time) ile ilgili. Dağılımlar, belirli aralıkların daha yaygın olduğunu gösteriyor.
- **AvgWinFlow**, **PktsIORatio**, **1stPktLen**: Bu histogramlar, ağ trafiği ve paket boyutlarına dair bilgi sağlar. Belirli değerler etrafındaki yoğunlaşmalar, bu değerlerin sıkça rastlandığını gösteriyor.
- **NumPorts**, **FlowLEN**, **FlowLENX**, **repeated_pkts_ratio**: Bu özellikler, ağ trafiğinin özelliklerine dair bilgiler sağlar. Özellikle NumPorts'un çoğunlukla 0 değeri etrafında yoğunlaştığını görüyoruz.
- **NumCon** ve **NumIPdst**: Bağlantı sayısı ve hedef IP adresi sayısı ile ilgili. İkisi de belirli değerler etrafında yoğunlaşıyor.
- **Start_flow**, **DeltaTimeFlow**, **HTTPpkts**: Bu özellikler, ağ trafiği başlangıç zamanları, zaman farkları ve HTTP paket sayısıyla ilgili. Yoğunlaşmalar, belirli değerlerin daha sık rastlandığını gösteriyor.
- **label**: Muhtemelen hedef değişken. Çoğunluğu 0 değeri etrafında, bu da bir sınıflandırma görevi için dengesiz bir veri seti olabileceğini gösterir.

Veri Setinin Mum Grafiğinin Yorumlanması



- **FinFlagDist**: Değerlerin büyük bir kısmı alt ve üst sınırlarda yoğunlaşmış. Bu, bu özelliğin kategorik veya ikili olabileceğini gösteriyor.
- **SynFlagDist**: Benzer şekilde, bu özellikte de değerler alt ve üst sınırlarda toplanmış durumda. Muhtemelen ikili bir özellik.
- **RstFlagDist**: Değerlerin çoğu alt sınıra yakın. Ancak 1 değerine doğru bir yoğunlaşma da var. Bu, bu özelliğin bazı durumlarda aktif hale geldiğini gösterebilir.
- **PshFlagDist**: Bu özellik için değerlerin dağılımı geniş. Bunun anlamı, bu özelliğin çeşitli durumları temsil edebileceğidir.
- **AckFlagDist**: Değerler oldukça dar bir aralıkta yoğunlaşmış. Bu özelliğin genellikle belirli bir değer etrafında toplandığını gösteriyor.
- **DNSoverIP** ve **TCPoverIP**: İkisi de benzer yapıda. Özellikle DNSoverIP'de değerlerin çoğu sıfırın etrafında.
- **UDPoverIP**: Değerlerin bir kısmı sıfırın etrafında toplanmış, ancak 1 değerine doğru da bir yoğunluk var.
- **MaxLen**: Bu özellikte değerlerin geniş bir dağılıma sahip olduğunu görebiliriz. Bu, ağ trafiğindeki maksimum uzunluğun farklı durumları kapsadığını gösterebilir.
- **MinLen**: Değerler daha dar bir aralıkta. Bu, trafiğin minimum uzunluğunun genellikle belirli bir aralıkta olduğunu gösteriyor.
- **StdDevLen** ve **AvgLen**: Her iki özellik için de geniş bir dağılım gözlemleniyor. Bu, ağ trafiği uzunluğunda önemli bir varyans olduğunu gösteriyor.
- **MaxIAT**, **MinIAT**, **AvgIAT**: Bu özellikler arasında geniş bir dağılım olduğunu görebiliriz. Özellikle MaxIAT'ta değerlerin büyük bir kısmı üst sınırda toplanmış.
- **AvgWinFlow**: Değerler dar bir aralıkta toplanmış. Bu özellik belirli bir durumu yansıtabilir.
- **PktsIORatio**: Bu özellik için değerler 0 ve 1 arasında dağılmış durumda.
- **1stPktLen** ve **NumPorts**: Geniş bir dağılım var. Bu özellikler farklı ağ trafiği durumlarını yansıtabilir.
- **FlowLEN**, **FlowLENX**: Geniş bir dağılım ve çok sayıda aykırı değer var.
- **repeated_pkts_ratio**: Çoğunlukla düşük değerler etrafında yoğunlaşma var, fakat üst sınırda da değerler mevcut.
- **NumCon**: Geniş bir dağılım mevcut.
- **Start_flow**, **DeltaTimeFlow**, **HTTPpkts**: Bu üç özellik için de geniş bir dağılım ve çok sayıda aykırı değer gözlemleniyor.
- **label**: Muhtemelen ikili bir hedef değişken. İki ana kategoriye sahip olabilir.

KAYNAKÇA:

Veri Seti: https://www.kaggle.com/datasets/mathurinache/mtakdd19/data

Github: https://github.com/Betullyr/Veri-Madenciligi