NSL-KDD Rapor

Siber Güvenlik için Veri Madenciliği

Harran Üniversitesi
Bilgisayar Mühendisliği

Kerem SÖYLEMEZ (200504060)
Hasan GÜRSES (180504082)

NSL-KDD Detaylı Bakış

NSL-KDD Genel Bakış

NSL-KDD, veri setinin sınırlamalarını ele almak için CIC (Kanada Siber Güvenlik Enstitüsü) tarafından yayımlanan KDD Cup'99 veri setinin geliştirilmiş bir versiyonudur.
Aynı zamanda web siteleri aracılığıyla araştırmacıların kullanımına açıktır.
NSL-KDD veri seti, KDD Cup'99 veri setinin sorunlarını ortadan kaldırmayı ve izinsiz giriş tespit araştırması için daha gerçekçi ve zorlu bir veri seti sağlamayı amaçlamaktadır.
Bu veri seti saldırı tespit sistemi araştırması alanında yaygın olarak kullanılmaktadır ve tespit oranları açısından KDD-99 veri setinden daha iyi performans göstermektedir. Veri kümesi aynı zamanda anormal veri tanıma gibi başka amaçlar için de kullanılabilir. NSL-KDD veri kümesi, farklı makine öğrenimi algoritmaları ve teknikleri ile kullanılabilmektedir.

Ne için Kullanılır?

NSL-KDD veri seti siber güvenlik alanında kullanılan bir veri setidir. Bunun içerisinde izinsiz giriş/sızma tespit (IDS) sistemleri değerlendirmede ve test amaçlarında kullanılır.

Tespit sistemlerini değerlendirmede makine öğrenmesi modellerden de yararlanılabilir. Veri seti barındırdıklarıyla kullanıcıya standartlaşmış bir veri seti sunduğundan onları kullanarak kendi tespit algoritmaların geliştirebilirler.

Burada birkaç kullanım alanını görebiliriz;

1. IDS Deneme/Test:
İzinsiz giriş tespit sistemlerini ve algoritmalarını test etmek için uygun ve standartlaşmış bir veri kümesi sağlar.

2. Algoritmalar:
İzinsiz giriş sistemlerinde geliştirilmiş ve geliştirilen algoritmalarının testine imkân sağlar. Eksikleri ve diğer algoritmalar ile kıyaslandığında avantaj ve dezavantajları bu şekilde ortaya çıkarır.

3. Eğitim ve Test:
Bu veri seti isimlendirilmiş birçok senaryo örneklerini içerisinde barındırır. BU kolaylıkla beraber makine öğrenme modelleriyle izinsiz giriş tespitleri için yazılımı train ve test etmek mümkün hale gelir. Bu örneklerde normal ağ trafiği dahil birçok saldırı senaryoları da barındırır.

4. Model Validation:
Geliştirilen veya mevcut bir izinsiz giriş sisteminin ne kadar verimli ve hassas çalıştığına saptanabilir. Bu sayede kullanılan veri kümesi içerisindeki hangi verinin atlandığı, hangi verinin kullanıldığı görülebilmekte ve gerçek ortamlar için veri elde edilebilmekte.

5. IDS Yaklaşımlarının Karşılaştırılması:
NSL-KDD yaygın olarak kullanılan bir standart haline geldiğinden birçok izinsiz giriş yöntemleri test edilebilir ve karşılaştırılabilmekte. Aynı zamanda tekniklerin güçlü ve zayıf yanları ortaya çıkartılabilmekte. Geliştirilen tekniğin limitleri belirlenebilir ve geliştirmeler yapılabilir. NSL-KDD eğitim ve geliştirme alanında geniş çaplı kullanımda bulunmakta. Bu yüzden veri setinin barındırdığı gerçek dünya senaryosunda yetersiz kalabilir.

Genel olarak NSL-KDD yaygın kullanılan izinsiz girişi tespit sistemlerinin çeşitli denemelerinin ve kıyaslamalarının yapılabildiği bir ortam sağlamakta.
Ama unutulmaması gerekir ki gerçek dünya ortamında veri seti içerisindeki veriler yetersiz kalabilir.

Neden NSL-KDD?

NSL-KDD deneme ve test setlerindeki kayıtların sayısı düşük düzeydedir.
Bu ise yüksek bir verimlilik sağlamaktadır. Bu avantaj, testte küçük bir kısmın rastgele seçilmesine gerek kalmadan deneylerin setin tamamı üzerinde gerçekleştirilmesini uygun maliyetli (verimli) hale getirir. KDD ile kıyaslanırsa NSL-KDD birçok avantajları vardır. Örneğin, test setlerinde tekrarlayan kayıt bulunmamaktadır.
Diğer avantajlar;

1. Gerçekçi Ağ Trafiği:
NSL-KDD gerçek bilgisayar ağı üzerinden geliştirildiğinden gerçekçi bir simülasyon sağlamakta. Bu da yeni, daha verimli, efektif ve güvenilir izinsiz giriş tespit modellerinin geliştirilmesini sağlar.

2. Saldırı Çeşitliliği:
Bu veri seti birçok saldırı çeşidini barındırmakta. Bu şekilde algoritmalar birçok alanda test edilebilir ve avantajlarına saptanabilir. Eksik kalınan yanlar da saptanır ve böylelikle daha emin adımlara geliştirme süreci işlenebilir. Bu çeşitlilik sayesinde sistemler çeşitli saldırıları seçebilir ve sınıflandırabilir.

3. Benchmarking:
NSL-KDD yaygın olarak kullanıldığından araştırmacılar ve geliştiriciler için izinsiz giriş tespit sistemlerinin karşılaştırılması ve farklı algoritmaların denenmesini sağlar.

4. Feature Azaltma:
Veri seti içerisinde feature seçimi ile istenilen özellikte ve sorgulamada işlem yapılabilmekte ve eğitim süreci bu şekilde sınırlandırılabilmekte. Bu şekilde sistemin veri arttırılabilmekte.

NSL-KDD Features (Özellikler)

NSL-KDD veri seti birçok ağ bağlantısını açıklayan özellikler barındırmakta.
Bu sayede izinsiz giriş tespit sistemleri eğitilebilmekte.
Örnekler;

1. Temel Özellikler:
Duration: Bağlantının süresi
Protocol_type: Bağlantı protokolü (tcp, udp, icmp).
Service: Hedefteki ağ hizmeti (http, ftp, telnet)
Flag: Bağlantının durumunu belirtir
(SF = normal,
S0 = bağlantı denemeleri,
REJ = izin verilmeyen bağlantılar)

2. Bağlantı Özellikleri:
Src_bytes: Kaynaktan hedefe kadar olan veri baytlarının sayısı
Dst_bytes: Hedeften kaynağa kadar olan veri baytlarının sayısı
Wrong_fragment: Yanlış fragment sayısı
Urgent: Acil paketler sayısı

3. Host-Bazlı Trafik Özellikleri:
Hot: Son iki saniye içinde aynı ana bilgisayara yapılan bağlantıların sayısı.
Num_failed_logins: Başarısız giriş işlemlerinin sayısı
Num_compromised: Risk altındaki koşulların sayısı.
Root_shell: 1 = if root shell is obtained, 0 = değilse.
Num_file_creations: The number of file creation operations.

4. Zaman-Bazlı Özellikler:
Count: Son iki saniye içinde geçerli bağlantıyla aynı ana bilgisayara yapılan bağlantıların sayısı.
Srv_count: Son iki saniye içinde geçerli bağlantıyla aynı hizmete yapılan bağlantıların sayısı.

5. İkili Özellikler:
Is_guest_login: 1 = misafir, 0 = değilse
Is_host_login: 1 = host, 0 = değilse
Is_guest_login: 1 = misafir, 0 = değilse

6. İsimlendirme:
Saldırı Türleri: DoS, Probe, R2L ve U2R

Bu özellikler makine öğrenmesi izinsiz giriş tespiti sistemlerinin eğitimi içindir.
İsimlendirmeler sayesinde bağlantı bir ağa mı bağlı yoksa saldırı altında mı belirtmekte.

Çalışmamızda Kullandığımız Yöntemler

Üzerinde çalıştığımız Lable bu şekilde listelenebilir.

"duration","protocol_type","service","flag","src_bytes",
"dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
"logged_in","num_compromised","root_shell","su_attempted","num_root",
"num_file_creations","num_shells","num_access_files","num_outbound_cmds",
"is_host_login","is_guest_login","count","srv_count","serror_rate",
"srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
"diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
"dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
"dst_host_rerror_rate","dst_host_srv_rerror_rate","label"

Alan olarak ise **flag**, **protocol_type**, **service** alanlarını (label) seçip onların üzerinde çalıştık,

Tüm verideki alanların hepsini gösterdik ve başlıklarına göre sayılarını listeledik.

1.Flag, Protocol_type, ve Service Alanları Üzerinde Çalışma
NSL-KDD veri kümesinde, analizimizi odakladığımız temel alanlar flag, protocol_type ve service'dir. Bu alanlar, ağ trafiğinin çeşitli özelliklerini temsil eder. Bu alanlar üzerinde detaylı bir inceleme yaparak, bu veri kümesinin güvenlik analitiği ve sınıflandırma problemleri için kullanılabilirliğini anlamaya çalıştık.

2. Kategorilerin İncelenmesi ve Grafiksel Temsili:
Seçtiğimiz flag, protocol_type ve service alanlarının kategorilerini inceledik. Her bir kategori içindeki örnek sayılarını belirleyerek, bu kategorilerin dağılımlarını görselleştirdik. Bu analiz, belirli ağ trafiği özelliklerinin veri kümesindeki dağılımını anlamamıza ve potansiyel anormal durumları tespit etmemize yardımcı oldu.

3. Tüm Veri Setinin Genel İncelenmesi ve Sayısal Analiz:
Ayrıca, NSL-KDD veri kümesinde bulunan tüm alanları gözden geçirip, başlıklarına göre sayılarını detaylı bir şekilde listeledik. Bu sayede, veri kümesinin genel yapısı hakkında kapsamlı bir anlayış elde ettik. Her bir özellik sütununun veri kümesindeki örnek sayılarına odaklanarak, hangi alanların daha yoğun veya seyrek olduğunu belirledik.

Bu analizler, NSL-KDD veri kümesini daha yakından tanımamıza ve üzerinde çalıştığımız özelliklerin veri setindeki önemini anlamamıza yardımcı oldu. Her bir kategori ve özellik sütunu üzerinde yapılan incelemeler, daha sonra uygulanacak olan makine öğrenimi modellerinin başarılı ve verimli bir şekilde eğitebilmemize katkı sağlayacaktır.

Veri Ön İşleme ve EDA İşlemlerimizin Kod ile Anlatımı ve Kullanım Nedenleri

Kullandığımız Kütüphaneler

1. pandas:
   Veri analizi için kullanılan güçlü bir Python kütüphanesidir. Veri analiz etmek için yaygın olarak kullanıldığından seçtik.

2. numpy:
Bilimsel hesaplamalar ve çok boyutlu diziler üzerinde çalışma için kullanılan bir kütüphanedir. Pandas gibi veri manipülasyonu ve analizi için sıkça kullanılır.

3. sys:
Sistemle işlemleri için kullanılan bir kütüphanedir.
Genellikle komut satırı argümanları ve çeşitli sistemle ilgili bilgilerle ilgilenir.

4. sklearn:
Makine öğrenimi için kullanılan bir kütüphanedir.
Sınıflandırma, regresyon, kümeleme, boyut azaltma ve model seçimi gibi çeşitli algoritmaları vardır.

5. io:
Input/Output işlemleri için kullanılan bir kütüphanedir. Veri okuma ve yazma işlemlerinde kullanılır.

6. random:
Rastgele sayı üretimi ve rastgele öğelerin seçimi işlemlerinde kullanılır.

7. seaborn:
İstatistiksel grafikler oluşturmak için kullanılır. İstatistiksel çıktılarımız için kullandık.

8. matplotlib.pyplot
Grafik oluşturma ve görselleştirme için kullanılır. İstettiklerimizi ve sonuçlarımızı çıktı halinde getirebilmek için kullandık

Kod ile Yaptığımız Veri Ön İşleme ve EDA İşlemleri

Verilerimizin Yüklenmesi

`train_url` ve `test_url` csv formatında train ve test NSL-KDD verilerimiz ekledik.

1. train_url: Eğitim veri setinin dosya konumunu temsil eder.

2. test_url: Test veri setinin dosya konumunu temsil eder.

Veri Seti İçerisindeki Sütunların Belirtilmesi ve Verinin Değişkene Aktarılması

1. col_names Liste Tanımlama:
Bu kısım ile veri setindeki sütun adlarını içeren bir liste oluşturduk. Her bir özellik için bir sütun adı içermekte.
["duration", "protocol_type", "service", "flag", "src_bytes", "label" vb.]

2. df Veri Çerçevesi Oluşturma (Training Set):
Bu kısım ile eğitim veri setini temsil eden bir pandas veri çerçevesi oluşturduk
. `pd.read_csv` fonksiyonu kullanılarak CSV dosyası okunuyor ve belirtilen sütun adları (`col_names`) ile veri çerçevesi başlıkları atanıyor.

3. Eğitim ve Test Seti Boyutlarının Yazdırılması:
Bu kısım ile eğitim ve test veri setlerinin boyutlarını yazdırdık.
print('Dimensions of the Training set:',df.shape) ve print('Dimensions of the Test set:,df_test.shape)

Bu işlemleri veri setinin yapısal analizi ve temel istatistiksel bilgileri görmek için kullandık.

İstatistiksel Bilgiler

Bu işlemde describe() fonksiyonunu kullanarak temel istatistikleri edindik.

- **df.**describe()

Bu fonksiyonunu kullanarak **temel istatistik** bilgilerini ekrana yazdırdık. `describe()` ile sayısal sütunlara (örneğin, ortalama, standart sapma, minimum, maksimum, çeyreklikler) ait temel istatistik bilgilerini ekrana yazdırır.

Veri Türlerinin Kontrolü

info() fonksiyonunu kullanarak sütunlardaki veri türlerini ve eksik veri sayılarının kontrolünü sağladık.

- **df**.info()

Bu kısım ile bir pandas `info()` fonksiyonunu kullanarak veri çerçevesindeki her bir sütunun türünü, bellek kullanımını ve dolu olmayan değer sayısını gibi temel bilgileri ekrana yazdırırdık.

Çıktıların Oluşturulması (Görselleştirme)

Bilgi Çıktısı:
Her bir sütunun adı ve veri türü (`int64`, `float64`, `object`, vb.).
İçerik:
- Dolu Olmayan Değer Sayısı: Her sütundaki dolu olmayan değerlerin sayısı.
- Toplam Satır Sayısı: Veri çerçevesindeki toplam satır sayısı.
- Bellek Kullanımı: Veri çerçevesinin bellekte kapladığı alan.

Kullandığımız 3 Özellik

1. df['flag'].value_counts()` Komutu:
Bu kısımda "flag" sütunundaki benzersiz değerlerin sayısını sayılır ve her bir değerin kaç kez tekrarlandığı gösterilir.

2. df['protocol_type'].value_counts()` Komutu:
Bu kısımda "protocol_type" sütunundaki benzersiz değerlerin sayısını sayılır ve her bir değerin kaç kez tekrarlandığı gösterilir.

3. df['service'].value_counts()` Komutu:
Bu kısımda "service" sütunun daki benzersiz değerlerin sayısını sayılır ve her bir değerin kaç kez tekrarlandığı gösterilir.

Bu şekilde kategorik verilerin dağılımını inceleyebildik ve sütunlardaki değerlerin frekansları/dağılımı anlayabildik. Bu bilgiler ile elde ettiğimiz ise veri setinin içeriği hakkında genel bir bakış elde etmek ön işleme adımlarını belirlemek.

1. print('Label distribution Training set:')`
Bu kısımda ise **eğitim** veri setindeki etiket (label) sütunundaki benzersiz değerlerin sayısı sayılır ve her bir etiketin kaç kez tekrarlandığı.

2. `print('Label distribution Test set:')`
Bu kısımda da **test** veri setindeki etiket (label) sütunundaki benzersiz değerlerin sayısını sayılır ve her bir etiketin kaç kez tekrarlandığı.

Bu kısımı veri setindeki sınıf etiketlerinin dağılımını kontrol etmek için kullandık.

Veri Ön İşleme – Grafik

One-Hot-Encoding, tüm kategorik özellikleri ikili özelliklere dönüştürmek için kullandık.

Giriş özelliklerinin [0, n_values] aralığında değerler aldıkları varsayılmakta.
Bu yüzden her kategoriyi bir sayısal değere için özelliklerin öncelikle LabelEncoder ile dönüştürülmesi gerekti.

NSL-KDD veri kümesinin belirlediğimiz kategorik sütunlarının **(protocol_type, service, flag)** dağılımını inceledik

1. Kategorik Sütunlar ve Dağılımları:
Bu süreçte türü 'object' ise, sütunun dağılımını gösteren bir çubuk grafik oluşturulur.
for col_name in df.columns:
Kategorik sütunlardaki benzersiz kategorilerin sayısını ve dağılımlarını gösteren **çubuk grafikleri oluşturuldu.**

2. Service Sütunu İçin Özel İşlemler
'service' sütununun dağılımını detaylı inceleme sonrası çubuk grafik oluşturuldu.
Ek olarak, en yaygın kategorilerin sayısını ve dağılımını içeren bir **tablo** oluşturuldu.

3. Grafiklerin Oluşturulması:

seaborn ile grafikler oluşturuldu.
`sns.countplot(x=col_name, data=df)`
Kategorik sütunlardaki değerlerin sayısını gösteren **çubuk grafikleri oluşturur.**

4. Tablo:
Grafikler dışında 'service' sütununun en yaygın kategorilerini içeren bir **tablo oluşturuldu**

Kategoriler

Bu kısımda, test veri setindeki her bir kategorik sütundaki benzersiz kategorilerin sayısını inceleyen bir döngü ekledik.

1. Test Setindeki Kategorik Sütunlar ve Benzersiz Kategorilerin Sayısı:

Veri setindeki her sütunun veri tipini kontrol eder.
Veri tipi 'object' ise, o sütundaki benzersiz(farklı) kategorilerin sayısı ekrana yazdırır.

for col_name in df_test.columns

2. Bilgi Çıktısı:
Her bir kategorik sütun için benzersiz kategorilerin sayısı ekrana yazdırdık (grafik).

print("Feature '{col_name}' has {unique_cat} categories".format(col_name=col_name, unique_cat=unique_cat))`

LabelEncoder - Kategorik Özellikleri 2B (numpy)

Scikit-learn kütüphanesinin `**LabelEncoder**` ve `**OneHotEncoder**` sınıflarını kullanarak belirli kategorik sütunlardaki değerleri işleyen bir dizi işlemi gerçekleştirdik.

1. Kategorik Sütunların Belirlenmesi:
Bu kısım, işlem yapılacak olan kategorik sütunların isimlerini içeren bir liste oluşturur.
Biz, **'protocol_type'**, **'service'**, ve **'flag'** sütunlarını seçtik.

2. Eğitim ve Test Veri Çerçevelerindeki Kategorik Sütun Değerlerinin Seçilmesi:
Bu kısımda eğitim ve test veri çerçevelerinden **sadece seçilen kategorik sütunların** değerlerini içeren yeni veri çerçeveleri oluşturduk.
`df_categorical_values = df[categorical_columns]` ve `testdf_categorical_values = df_test[categorical_columns]`

3. Eğitim Veri Çerçevesindeki Kategorik Sütun Değerlerinin Görüntülenmesi:
Bu kısım, eğitim veri çerçevesindeki seçilen kategorik sütunların ilk beş satırını yazdırdık.
`df_categorical_values.head()`

4. LabelEncoder ve OneHotEncoder Kullanımı:

Bu kısımda `LabelEncoder` ve `OneHotEncoder` sınıflarını kullanarak kategorik sütunlardaki değerlerini işledik.
`**LabelEncoder**`, kategorik sütunlardaki değerleri sayısal formata dönüştürür,
`**OneHotEncoder**` her bir kategoriye ait yeni bir sütun ekler ve sütunlardaki değerleri ikili (binary) formata dönüştürür.

Seçtiğimiz Özelliklerimiz (sevice, protocol_type, flag)

1. Protocol Type İçin Benzersiz Değerlerin İşlenmesi:
Bu kısım ile 'protocol_type' sütunundaki benzersiz değerleri alıyoruz, sıralıyoruz ve her bir değeri 'Protocol_type_' ön ekini ekleyerek yeni bir liste oluşturuluyor.
`unique_protocol=sorted(df.protocol_type.unique())`, `string1 = 'Protocol_type_'`, `unique_protocol2=[string1 + x for x in unique_protocol]`

2. Service İçin Benzersiz Değerlerin İşlenmesi:
Bu kısım, 'service' sütunundaki benzersiz değerleri alıyoruz, sıralıyoruz ve her bir değeri 'service_' ön ekini ekleyerek yeni bir liste oluşturuluyor.
`unique_service=sorted(df.service.unique())`, `string2 = 'service_'`, `unique_service2=[string2 + x for x in unique_service]`

3. Flag İçin Benzersiz Değerlerin İşlenmesi:
Bu kısım, 'flag' sütunundaki benzersiz değerleri alıyoruz, sıralıyoruz ve her bir değeri 'flag_' ön ekini ekleyerek yeni bir liste oluşturuluyor.
`unique_flag=sorted(df.flag.unique())`, `string3 = 'flag_'`, `unique_flag2=[string3 + x for x in unique_flag]`

Oluşturduğumuz Bütün Listelerin Birleştirilmesi:
Bu kısım, 'protocol_type', 'service', ve 'flag' sütunlarındaki benzersiz değerleri içeren tüm listeleri birleştirerek final bir sütun adı listesi elde ediyoruz.

Test Seti İçin Service İçin Benzersiz Değerlerin İşlenmesi:
Bu kısım, test veri setindeki 'service' sütunundaki benzersiz değerleri alıyoruz, sıralıyoruz ve her bir değeri 'service_' ön ekini ekleyerek yeni bir liste oluşturuluyor.

Test Seti İçin Tüm Listelerin Birleştirilmesi:
Bu kısım, test veri setindeki 'protocol_type', 'service', ve 'flag' sütunlarındaki benzersiz değerleri içeren tüm listeleri birleştirerek final bir sütun adı listesi oluşturuluyor.

Bu kısımda elde ettiğimiz ise kategorik sütunlardaki benzersiz değerleri alarak bu değerleri sütun adlarına dönüştürmek. Böylelikle veri setinin boyutunu artırabilir ve bu sütunların bir makine öğrenimi modeline uygun hale getirilmesine sağlayabiliriz.

Kategoriler Sayısallaştırma

LabelEncoder() (kategorik özellikleri sayısal dönüşüm)

Bu kısımda, `LabelEncoder` kullanarak kategorik sütunlardaki değerleri sayısal forma dönüştürdük.

1. Eğitim Veri Çerçevesinde Label Encoding İşlemi:
Bu kısımda `LabelEncoder` sınıfını kullanarak eğitim veri çerçevesindeki kategorik sütunlardaki değerleri sayısal forma dönüştürüyoruz.
`fit_transform` fonksiyonu, her bir sütundaki benzersiz kategorik değerlere bir sayı atar.
`df_categorical_values_enc=df_categorical_values.apply(LabelEncoder().fit_transform)`

2. Eğitim Veri Çerçevesindeki İlk Beş Satırın Görüntülenmesi:
Bu kısımda eğitim veri çerçevesindeki hem kategorik sütunlardaki orijinal değerleri hem de bu değerlerin sayısal forma dönüştürülmüş halleri ekrana yazdırılıyor.
`print(df_categorical_values.head())` ve `print(df_categorical_values_enc.head())`

3. Test Veri Çerçevesinde Label Encoding İşlemi:
Bu kısımda eğitimde kullanılan `LabelEncoder`'ı test veri çerçevesindeki kategorik sütunlara uygulanıyor. Bu şekilde eğitim ve test veri çerçevelerindeki kategorik sütunlardaki değerler aynı dönüşümle sayısal forma getirilmiş oluyoruz.
`testdf_categorical_values_enc=testdf_categorical_values.apply(LabelEncoder().fit_transform)`

Bu işlemlerle, makine öğrenimi modellerine uygulanacak olan veri setlerindeki kategorik değerleri sayısal hale getirebiliriz.

One-Hot Encoder

`OneHotEncoder` ile eğitim ve test veri çerçevelerindeki sayısal değerlerini ikili (binary) formata dönüştürür.

1. One-Hot Encoding İşlemi:
Bu kısımda `OneHotEncoder` sınıfını kullanarak eğitim veri çerçevesindeki sayısal forma dönüştürülmüş kategorik değerleri ikili (binary) forma dönüştürüyoruz.
`enc = OneHotEncoder(categories='auto')`, `df_categorical_values_encenc = enc.fit_transform(df_categorical_values_enc)`

2. One-Hot Encoding Sonucunun DataFrame'e Dönüştürülmesi:
Bu kısımda One-Hot Encoding işlemi edilen seyrek matrisi DataFrame'e dönüştürür.
`toarray()` fonksiyonu, seyrek matrisi yoğun (dense) bir matrise çevrilir.
`df_cat_data = pd.DataFrame(df_categorical_values_encenc.toarray(),columns=dumcols)`

3. Test Veri Seti İçin One-Hot Encoding İşlemi ve DataFrame'e Dönüştürülmesi:

Bu kısımda One-Hot Encoding işlemini test veri çerçevesindeki sayısal forma dönüştürülmüş kategorik değerlere uygular. Sonrasında ise elde ettiğimiz seyrek matrisi bir DataFrame'e dönüştürür.
`testdf_categorical_values_encenc = enc.fit_transform(testdf_categorical_values_enc)`, `testdf_cat_data = pd.DataFrame(testdf_categorical_values_encenc.toarray(),columns=testdumcols)`

4. DataFrame'in İlk Beş Satırının Görüntülenmesi:
Bu kısım, oluşturulan DataFrame'in ilk beş satırını ekrana yazdırır ve sonucun doğruluğu kontrol edilir.
`df_cat_data.head()`

Bu kısımda elde ettiğimiz, kategorik değerlerin One-Hot Encoding yöntemiyle binary hale dönüştürülmesi.

Test setteki eksik sütunlar eklendi

Bu kısımda amacımız ise eğitim ve test veri setleri arasındaki **'service'** sütunundaki benzersiz değerler arasındaki farkı bulmak ve bu elde edilen farklılık setini oluşturmak.

Eğitim ve Test Veri Setlerindeki 'service' Sütunundaki Değerlerin Listeye Dönüştürülmesi:
Bu kısımda eğitim ve test veri setlerindeki **'service'** sütunundaki değerleri birer listede toplanır.
`trainservice=df['service'].tolist()`, `testservice= df_test['service'].tolist()`

Eğitim ve Test Veri Setleri Arasındaki Farkın Bulunması:
Bu kısımda eğitim ve test veri setleri arasındaki **'service'** sütunundaki benzersiz değerlerin farkı buluyoruz.
`difference=list(set(trainservice) - set(testservice))`

Farklılık Setinin 'service_' Ön Eki ile Etiketlenmesi
Bu kısımda bulunan farklılık setindeki her bir değere **'service**_' ön ekini ekleniyor ve yeni bir liste elde ediliyor.
`string = 'service_'`, `difference=[string + x for x in difference]`

Farklılık Seti:
Bu kısım, oluşturulan farklılık setini ekrana yazdırır.
Eğitim veri setinde bulunan fakat test veri setinde bulunmayan **'service'** değerlerini gösterir.
`difference`

Bu kısımda amaçladığımız, eğitim ve test veri setleri arasındaki kategorik değerlerin farklılıklarını incelemek.
Bu şekilde modelin doğru bir şekilde genelleme yapabilmesi ve veri setinin tutarlılığını kontrol ettik.

Veri Setimizde Eksik veya Farklı Veriler

Bu aşamada test veri setindeki **'service'** sütunundaki benzersiz değerlerini ve eğitim veri setindeki 'service' sütunundaki benzersiz değerler arasındaki farkı ele aldık. Bu farklılık setinde bulunan her bir değeri yeni bir sütun olarak ekleyip ve bu sütunların değerlerini sıfırladık.

1. Aşama - Farklılık Setindeki Her Bir Değer İçin Yeni Sütun Eklenmesi:
Bu kısım farklılık setinde bulunan her **'service'** değeri için test veri setine yeni bir sütun ekler ve sütunun değerlerini sıfırlar (null).
`for col in difference: testdf_cat_data[col] = 0`

2. Aşama - Eğitim ve Test Veri Setlerinin Boyutlarının Görüntülenmesi:
Eğitim ve test veri setlerinin boyutları yazdırılır. Bu şekilde işlemin gerçekleşip gerçekleşmediğini kontrol ettik.
`print(df_cat_data.shape)`, `print(testdf_cat_data.shape)`

One-Hot Encoding ve Gereksiz Veri Kaldırma

Eğitim Veri Seti İçin İşlemler - Sütunların Birleştirilmesi
Bu kısım ile eğitim veri setindeki kategorik sütunların One-Hot Encoding çıktısıyla genişletilmiş olan `df_cat_data` veri çerçevesini ana veri çerçevesi `df` ile birleştirilir.

Gereksiz Sütunların Kaldırılması:
Seçtiğimiz değerlerden **'flag'**, **'protocol_type'**, ve **'service'** sütunları, yeni oluşturulan kategorik sütunlar ile birleştirildiğinde gereksiz olmuş oluyorlar ve şekilde bu sütunların veri çerçevesinden kaldırdık.
newdf.drop('flag', axis=1, inplace=True)`, `newdf.drop('protocol_type', axis=1, inplace=True)`, `newdf.drop('service', axis=1, inplace=True)`

Bu şekilde, kategorik sütunlardaki değerlerin One-Hot Encoding sonuçlarıyla genişletilmesini ve gereksiz sütunların kaldırılmasını sağladık (veri ön işleme adımları).
Bu adımları atmamızın nedeni ise bir ön hazırlıktır. Modelin eğitimi ve genelleme yeteneklerini iyileştirmeye yöneliktir.

Sınıflandırma Uygunluk

Veri seti her atak kategorisi için ayrı veri setlerine ayrıldı.
0=Normal, 1=DoS, 2=Probe, 3=R2L, 4=U2R.

DoS :

Probe :

R2L :

U2R :

Bu şekilde, 'label' sütununu yeni bir şekilde etiketlenerek, sınıflandırma modeli için uygun bir hale getirdik.

Filtreleme

Bu kısımda ise belirli etiketlere sahip olan satırları filtrelenir ve ayrı ayrı veri çerçevesi oluşturur.

1. Filtreleme Kriterleri:
Bu kısım, her sınıf için filtreleme kriterlerini belirler.
DoS sınıfı için '0' ve '1' etiketlerine sahip satırları içerir.

```
to_drop_DoS = [0,1]
to_drop_Probe = [0,2]
to_drop_R2L = [0,3]
to_drop_U2R = [0,4]
```

2. Filtreleme İşlemi:
Bu kısım, belirli etiketlere sahip satırları filtreler ve dört ayrı veri çerçevesi oluşturur.

```
DoS_df=newdf[newdf['label'].isin(to_drop_DoS)]
Probe_df=newdf[newdf['label'].isin(to_drop_Probe)]
R2L_df=newdf[newdf['label'].isin(to_drop_R2L)]
U2R_df=newdf[newdf['label'].isin(to_drop_U2R)]
```

3. Filtrelenmiş Veri Çerçevelerinin Boyutlarının Görüntülenmesi:
Bu kısım, filtrelenmiş veri çerçevelerinin boyutlarını ekrana yazdırlır. Böylelikle her bir sınıfa ait örnek sayısını kontrol edilebilir.

```
print('Train:')
print('Dimensions of DoS:' ,DoS_df.shape)
print('Dimensions of Probe:' ,Probe_df.shape)
print('Dimensions of R2L:' ,R2L_df.shape)
print('Dimensions of U2R:' ,U2R_df.shape)
print()
print('Test:')
print('Dimensions of DoS:' ,DoS_df_test.shape)
print('Dimensions of Probe:' ,Probe_df_test.shape)
print('Dimensions of R2L:' ,R2L_df_test.shape)
```

Özellik Ölçeklendirme

Bu kısımda veri çerçevelerini özellikler (X) ve hedef değişkenler (Y) olarak ayırdık.
Aynı zamanda her bir sınıf için eğitim ve test setleri oluşturduk.

Eğitim Seti İçin Veri Ayrımı (DoS Sınıfı):
Özellikler (X): DoS sınıfının özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): DoS sınıfının etiketlerini içeren veri çerçevesi oluşturduk.

```
X_DoS = DoS_df.drop('label',axis=1)
Y_DoS = DoS_df.label
```

Eğitim Seti İçin Veri Ayrımı (Probe Sınıfı):

Özellikler (X): Probe sınıfının özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): Probe sınıfının etiketlerini içeren veri çerçevesi oluşturduk.

```
X_Probe = Probe_df.drop('label',axis=1)
Y_Probe = Probe_df.label
```

Eğitim Seti İçin Veri Ayrımı (R2L Sınıfı):
Özellikler (X): R2L sınıfının özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): R2L sınıfının etiketlerini içeren veri çerçevesi oluşturduk.

```
X_R2L = R2L_df.drop('label',axis=1)
Y_R2L = R2L_df.label
```

4. Eğitim Seti İçin Veri Ayrımı (U2R Sınıfı):
Özellikler (X): U2R sınıfının özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): U2R sınıfının etiketlerini içeren veri çerçevesi oluşturduk.

```
X_U2R = U2R_df.drop('label',axis=1)
Y_U2R = U2R_df.label
```

5. Test Seti İçin Veri Ayrımı (DoS Sınıfı):
Özellikler (X): DoS sınıfının test verilerinin özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y):
DoS sınıfının test verilerinin etiketlerini içeren veri çerçevesi oluşturduk.

```
X_DoS_test = DoS_df_test.drop('label',axis=1)
Y_DoS_test = DoS_df_test.label
```

6. Test Seti İçin Veri Ayrımı (Probe Sınıfı):
Özellikler (X): Probe sınıfının test verilerinin özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): Probe sınıfının test verilerinin etiketlerini içeren veri çerçevesi oluşturduk.

```
X_Probe_test = Probe_df_test.drop('label',axis=1)
Y_Probe_test = Probe_df_test.label
```

7. Test Seti İçin Veri Ayrımı (R2L Sınıfı):
Özellikler (X): R2L sınıfının test verilerinin özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): R2L sınıfının test verilerinin etiketlerini içeren veri çerçevesi oluşturduk.

```
X_R2L_test = R2L_df_test.drop('label',axis=1)
Y_R2L_test = R2L_df_test.label
```

Test Seti İçin Veri Ayrımı (U2R Sınıfı):
Özellikler (X): U2R sınıfının test verilerinin özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): U2R sınıfının test verilerinin etiketlerini içeren veri çerçevesi oluşturduk.

```
X_U2R_test = U2R_df_test.drop('label',axis=1)
Y_U2R_test = U2R_df_test.label
```

Bu tür bir veri ayrımı işlemi ile özellik matrisini (X) ve hedef değişkeni (Y) olarak ayırdık ve hedefimiz makine öğrenimi modellerinin eğitilmesi ve değerlendirilmesi için veri setini hazırlamak.

Sınıflar için Test ve Eğitim Setleri

Bu kısımda ise veri çerçevelerini özellikler (X) ve hedef değişkenler (Y) olarak ayırdık, her bir sınıf için eğitim ve test setleri oluşturduk.

1. Eğitim Seti İçin Veri Ayrımı (DoS Sınıfı):
Özellikler (X): DoS sınıfının özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): DoS sınıfının etiketlerini içeren veri çerçevesi oluşturduk.

```
X_DoS = DoS_df.drop('label',axis=1)
Y_DoS = DoS_df.label
```

2. Eğitim Seti İçin Veri Ayrımı (Probe Sınıfı):
Özellikler (X): Probe sınıfının özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): Probe sınıfının etiketlerini içeren veri çerçevesi oluşturduk.

```
X_Probe = Probe_df.drop('label',axis=1)
Y_Probe = Probe_df.label
```

3. Eğitim Seti İçin Veri Ayrımı (R2L Sınıfı):
Özellikler (X): R2L sınıfının özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): R2L sınıfının etiketlerini içeren veri çerçevesi oluşturduk.

```
X_R2L = R2L_df.drop('label',axis=1)
Y_R2L = R2L_df.label
```

4. Eğitim Seti İçin Veri Ayrımı (U2R Sınıfı):
Özellikler (X): U2R sınıfının özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): U2R sınıfının etiketlerini içeren veri çerçevesi oluşturduk.

```
X_U2R = U2R_df.drop('label',axis=1)
Y_U2R = U2R_df.label
```

5. Test Seti İçin Veri Ayrımı (DoS Sınıfı):
Özellikler (X): DoS sınıfının test verilerinin özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): DoS sınıfının test verilerinin etiketlerini içeren veri çerçevesi oluşturduk.

```
X_DoS_test = DoS_df_test.drop('label',axis=1)
Y_DoS_test = DoS_df_test.label
```

6. Test Seti İçin Veri Ayrımı (Probe Sınıfı):
Özellikler (X): Probe sınıfının test verilerinin özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): Probe sınıfının test verilerinin etiketlerini içeren veri çerçevesi oluşturduk.

```
X_Probe_test = Probe_df_test.drop('label',axis=1)
Y_Probe_test = Probe_df_test.label
```

7. Test Seti İçin Veri Ayrımı (R2L Sınıfı):

Özellikler (X): R2L sınıfının test verilerinin özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): R2L sınıfının test verilerinin etiketlerini içeren veri çerçevesi oluşturduk.

```
X_R2L_test = R2L_df_test.drop('label',axis=1)
Y_R2L_test = R2L_df_test.label
```

8. Test Seti İçin Veri Ayrımı (U2R Sınıfı):
Özellikler (X): U2R sınıfının test verilerinin özelliklerini içeren veri çerçevesinden 'label' sütunu çıkardık.
Hedef Değişken (Y): U2R sınıfının test verilerinin etiketlerini içeren veri çerçevesi oluşturduk.

```
X_U2R_test = U2R_df_test.drop('label',axis=1)
Y_U2R_test = U2R_df_test.label
```

Bu tür bir veri ayrımı işlemi ile hedefimiz, makine öğrenimi modellerinin eğitilmesi ve değerlendirilmesi için veri setini hazırlamak (ön işlem).

Sınıf Kullanımı - Standartlaştırma

StandardScaler sınıfını kullanarak veri setinin standartlaştırılmasını sağladık.
Standartlaştırma işlemi ile her bir özellik sütununu ortalaması 0, standart sapması ise 1 olacak şekilde dönüştürdük ve böylelikle özelliklerin birbirleriyle aynı ölçeğe sahip olmasını elde ettik.

1. DoS Sınıfı İçin Standartlaştırma (`scaler1`):
`StandardScaler` ile DoS sınıfına ait eğitim setinin standartlaştırılması.

```
scaler1 = preprocessing.StandardScaler().fit(X_DoS)
X_DoS = scaler1.transform(X_DoS)
```

2. Probe Sınıfı İçin Standartlaştırma (`scaler2`):
Bu kısım, `StandardScaler` ile Probe sınıfına ait eğitim setinin standartlaştırılması.

```
scaler2 = preprocessing.StandardScaler().fit(X_Probe)
X_Probe = scaler2.transform(X_Probe)
```

3.R2L Sınıfı İçin Standartlaştırma (`scaler3`):
Bu kısım, `StandardScaler` ile R2L sınıfına ait eğitim setinin standartlaştırılması.

```
scaler3 = preprocessing.StandardScaler().fit(X_R2L)
X_R2L = scaler3.transform(X_R2L)
```

4. U2R Sınıfı İçin Standartlaştırma (`scaler4`):
`StandardScaler` ile U2R sınıfına ait eğitim setinin standartlaştırılması

```
scaler4 = preprocessing.StandardScaler().fit(X_U2R)
X_U2R = scaler4.transform(X_U2R)
```

5. DoS Sınıfı İçin Test Verilerinin Standartlaştırılması (`scaler5`):
`StandardScaler` ile DoS sınıfına ait test setinin standartlaştırılması.

```
scaler5 = preprocessing.StandardScaler().fit(X_DoS_test)
X_DoS_test = scaler5.transform(X_DoS_test)
```

6. Probe Sınıfı İçin Test Verilerinin Standartlaştırılması (`scaler6`):

Bu kısım, `StandardScaler` kullanarak Probe sınıfına ait test setinin standartlaştırılması.

```
scaler6 = preprocessing.StandardScaler().fit(X_Probe_test)
X_Probe_test = scaler6.transform(X_Probe_test)
```

7. R2L Sınıfı İçin Test Verilerinin Standartlaştırılması (`scaler7`):

Bu kısım, `StandardScaler` kullanarak R2L sınıfına ait test setinin standartlaştırılması.

```
scaler7 = preprocessing.StandardScaler().fit(X_R2L_test)
X_R2L_test = scaler7.transform(X_R2L_test)
```

8. U2R Sınıfı İçin Test Verilerinin Standartlaştırılması (`scaler8`):

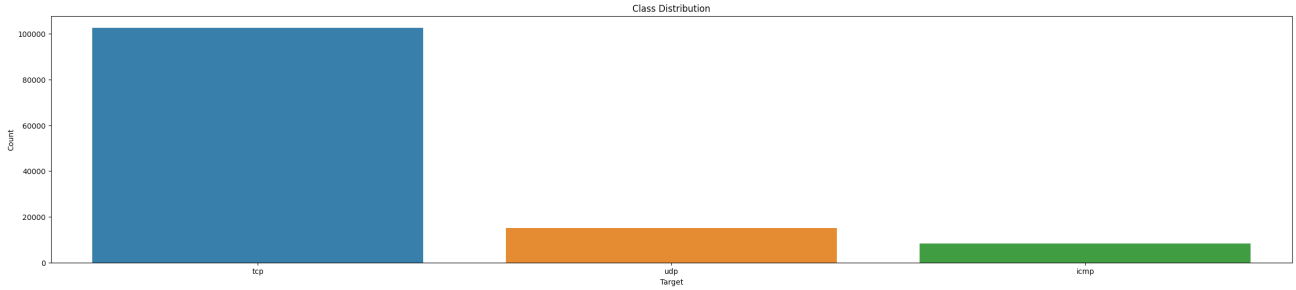
Bu kısım, `StandardScaler` kullanarak U2R sınıfına ait test setinin standartlaştırılması

```
scaler8 = preprocessing.StandardScaler().fit(X_U2R_test)
X_U2R_test = scaler8.transform(X_U2R_test)
```

Bu standartlaştırma işlemleri, ile hedefimiz özelliklerin farklı ölçeklere sahip olma durumunu düzeltilmek ki bu da birçok makine öğrenimi algoritması için önemlidir (ileriye dönük). Standartlaştırma ile elde ettiğimiz ise, modellerin daha iyi performans göstermesi ve daha iyi genelleme yapması.

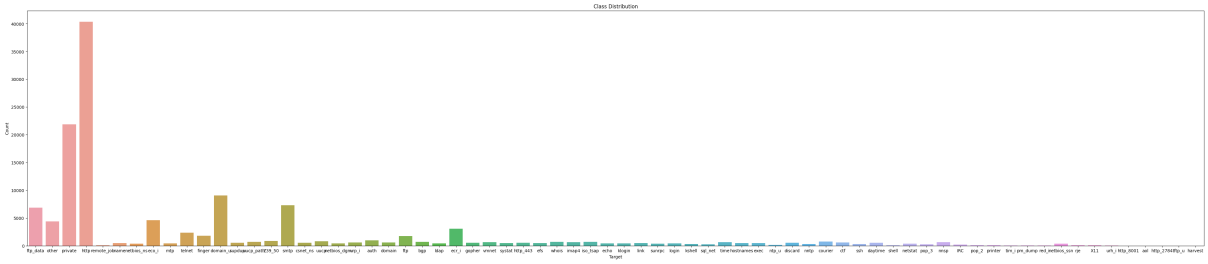
Görselleştirmeler ve İstatistikler Çıktılar:

Bu grafik protocol_type özelliğinin 3 kategorisi göstermekte.



-**tcp** kategorisi 100000 (yüksek değere),
-**udp** 18000,
-**icmp** ise 15000.

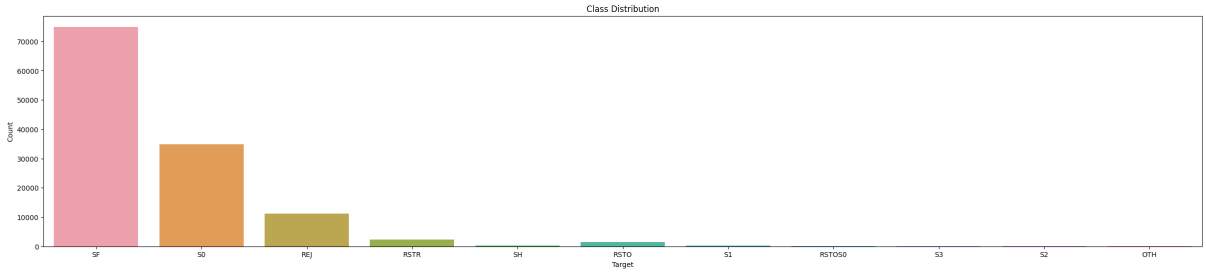
Bu Grafikte özellik 'service' 70 kategorisini görebiliyoruz



40000 ile http

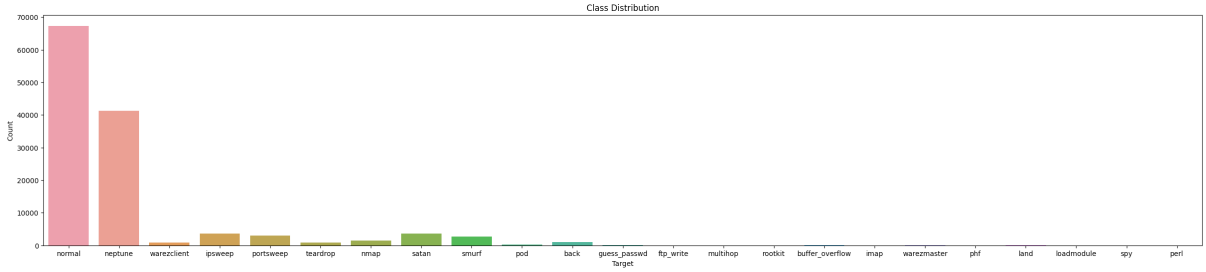
22000 ile private ve
6000 ile ftp-data gelmekte (en yüksek değerler olarak).

Bu grafikte özellik 'flag' 11 kategorisini görebiliyoruz



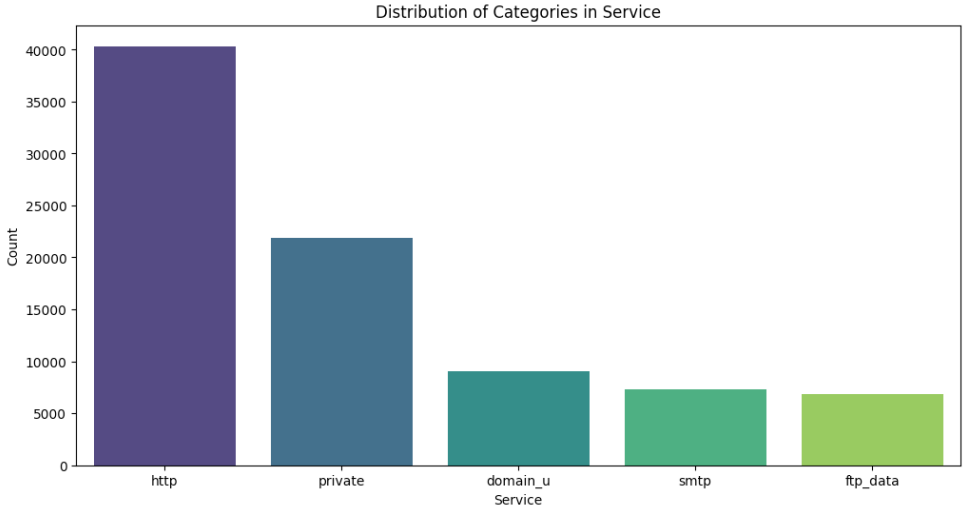
SF flag olarak 70000 üstü ile en yüksek değere sahip.
SO ise 30000 üstü ve REJ 10000 üstü ile SF'yi takip etmekte.

Bu grafikte özellik 'label' 23 kategorisini görebiliyoruz.



En yüksek özellik olarak normal lable'ı görebiliyoruz.
70000'ne yakın bir sayılma ile en yüksek değere sahip.
Ondan sonra gelen neptune label ise 40000 sayılma ile ikinci olarak gözükmekte.

Bu grafikte ise özellik service label içerisindeki kategorilerin sayılma değerlerini görebiliyoruz.



Çubuk grafikte ise http'nin 40000 ile en yüksek olduğunu görebiliriz.

Sonradan ise 22000 ile private ve 10000 ile domain_u gelmekte.

Distribution of categories in service:
service
http 40338
private 21853
domain_u 9043
smtp 7313
ftp_data 6860
Name: count, dtype: int64