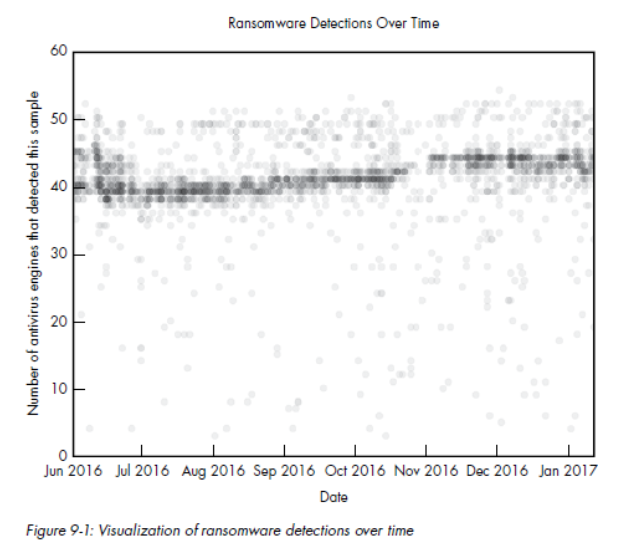
**9.Visualizing Malware Trends:Kötü Amaçlı Yazılım Trendlerini Görselleştirme:**

Bazen kötü amaçlı yazılım koleksiyonlarını analiz etmenin en iyi yolu onları görselleştirmektir. Güvenlik verilerini görselleştirmek, kötü amaçlı yazılımlardaki ve genel olarak tehdit ortamındaki eğilimleri hızlı bir şekilde tanımamızı sağlar. Bu görselleştirmeler genellikle görsel olmayan istatistiklerden çok daha sezgiseldir ve farklı izleyicilere içgörüleri iletmeye yardımcı olabilirler. Örneğin, bu bölümde görselleştirmenin bir veri kümesinde yaygın olan kötü amaçlı yazılım türlerini, kötü amaçlı yazılım veri kümelerindeki eğilimleri (örneğin, 2016'da bir eğilim olarak fidye yazılımının ortaya çıkışı) ve ticari kötü amaçlı yazılımları tespit etmede antivirüs sistemleri. Bu örnekler üzerinde çalışarak, Python veri analizi paketi pandalarının yanı sıra, seaborn ve matplotlib Python veri görselleştirme paketlerini kullanarak değerli içgörülere yol açabilecek kendi görselleştirmelerinizi nasıl oluşturacağınızı anlıyorsunuz. Pandalar paketi çoğunlukla verileri yüklemek ve işlemek için kullanılır ve veri görselleştirmenin kendisiyle pek ilgisi yoktur, ancak verileri görselleştirme için hazırlamak için çok kullanışlıdır.

\*Why Visualizing Malware Data Is Important:

Kötü amaçlı yazılım verilerini görselleştirmenin nasıl yardımcı olabileceğini görmek için iki örnek üzerinden geçelim. İlk görselleştirme şu soruyu ele alıyor: Antivirüs endüstrisinin fidye yazılımlarını algılama yeteneği artıyor mu? İkinci görselleştirme, bir yıl boyunca hangi kötü amaçlı yazılım türlerinin trend olduğunu soruyor. Şekil 9-1'de gösterilen ilk örneğe bakalım.



Bu fidye yazılımı görselleştirmesini, binlerce fidye yazılımı kötü amaçlı yazılım örneğinden toplanan verileri kullanarak oluşturdum. Bu veriler, her dosyada 57 ayrı antivirüs motoru çalıştırmanın sonuçlarını içerir. Her daire bir kötü amaçlı yazılım örneğini temsil eder. Y ekseni, tarandığında her kötü amaçlı yazılım örneğinin antivirüs motorlarından kaç tane algılama veya pozitif aldığını gösterir. Bu y ekseni 60'ta dururken, belirli bir tarama için maksimum sayının toplam tarayıcı sayısı olan 57 olduğunu unutmayın. X ekseni, her kötü amaçlı yazılım örneğinin ilk olarak kötü amaçlı yazılım analiz sitesi VirusTotal.com'da görüldüğü ve tarandığı zamanı gösterir. Bu arsada, virüsten koruma topluluğunun bu kötü amaçlı dosyaları tespit etme becerisinin Haziran 2016'da görece güçlü bir şekilde başlayıp Temmuz 2016'da düştüğünü ve ardından yılın geri kalanında istikrarlı bir şekilde arttığını görebiliriz. 2016'nın sonunda, fidye yazılımı dosyaları, virüsten koruma motorlarının ortalama yüzde 25'i tarafından hala gözden kaçırılıyordu, bu nedenle, güvenlik topluluğunun bu süre zarfında bu dosyaları tespit etmede biraz zayıf kaldığı sonucuna varabiliriz. Bu araştırmayı genişletmek için, hangi antivirüs motorlarının fidye yazılımını ne oranda algıladığını ve zaman içinde nasıl geliştiğini gösteren bir görselleştirme oluşturabilirsiniz. Ya da başka bir kötü amaçlı yazılım kategorisine (örneğin, Truva atları) bakabilirsiniz. Bu tür planlar, hangi virüsten koruma motorlarını satın alacağınıza veya hangi tür kötü amaçlı yazılımlara yönelik özel algılama çözümleri tasarlamak isteyebileceğinize karar vermede yararlıdır - belki de ticari bir virüsten koruma algılama sistemini desteklemek için (özel algılama sistemleri oluşturma hakkında daha fazla bilgi için bkz. Bölüm 8).

Şimdi, Şekil 9-1 için kullanılan veri kümesinin aynısı kullanılarak oluşturulan başka bir örnek görselleştirme olan Şekil 9-2'ye bakalım.

Şekil 9-2: Zaman içinde aile bazında kötü amaçlı yazılım tespitlerinin görselleştirilmesi

Şekil 9-2, en yaygın 20 kötü amaçlı yazılım ailesini ve bunların 150 günlük bir süre içinde birbirlerine göre ne sıklıkta meydana geldiğini göstermektedir. Konu bazı önemli bilgileri ortaya koyuyor: En popüler kötü amaçlı yazılım ailesi Allaple.A, 150 gün boyunca tutarlı bir şekilde ortaya çıkarken, Nemucod.FG gibi diğer kötü amaçlı yazılım aileleri daha kısa süreler boyunca yaygındı ve sonra sessizliğe büründü. Kendi iş yerinizin ağında tespit edilen kötü amaçlı yazılımlar kullanılarak oluşturulan bunun gibi bir olay örgüsü, zaman içinde kuruluşunuza yönelik saldırılara ne tür kötü amaçlı yazılımların dahil olduğunu gösteren yararlı eğilimleri ortaya çıkarabilir. Bunun gibi bir karşılaştırma figürü oluşturulmadan, bu kötü amaçlı yazılım türlerinin zaman içindeki göreceli zirvelerini ve hacimlerini anlamak ve karşılaştırmak zor ve zaman alıcı olacaktır.

Bu iki örnek, kötü amaçlı yazılım görselleştirmenin ne kadar yararlı olabileceğini göstermektedir. Bu bölümün geri kalanı, kendi görselleştirmelerinizi nasıl oluşturacağınızı gösterir. Bu bölümde kullanılan örnek veri setini tartışarak başlıyoruz ve ardından verileri analiz etmek için pandalar paketini kullanıyoruz. Son olarak, verileri görselleştirmek için matplotlib ve seaborn paketlerini kullanıyoruz

\*Understanding Our Malware Dataset:Kötü Amaçlı Yazılım Veri Kümemizi Anlamak

Kullandığımız veri kümesi, bir kötü amaçlı yazılım algılama toplama hizmeti olan VirusTotal tarafından toplanan 37.000 benzersiz kötü amaçlı yazılım ikili dosyasını açıklayan veriler içerir. Her ikili dosya dört alanla etiketlenir: ikili dosyayı kötü amaçlı olarak işaretleyen antivirüs motorlarının sayısı (57'den) (buna her örnekle ilişkili pozitif sayısı diyorum), her ikilinin boyutu, ikilinin türü (bitcoin madencisi , keylogger, fidye yazılımı, truva atı veya solucan) ve ikili dosyanın ilk görüldüğü tarih. Göreceğiz ki, her ikili program için bu oldukça sınırlı miktarda meta veriyle bile, verileri veri kümesine ilişkin önemli içgörüleri ortaya çıkaracak şekilde analiz edip görselleştirebileceğimizi göreceğiz.

-Loading Data into pandas:

Popüler Python veri analizi kitaplığı pandaları, verileri DataFrames adı verilen analiz nesnelerine yüklemeyi kolaylaştırır ve ardından yeniden paketlenen bu verileri dilimlemek, dönüştürmek ve analiz etmek için yöntemler sağlar. Verilerimizi yüklemek ve analiz etmek ve kolay görselleştirme için hazırlamak için pandalar kullanıyoruz. Python yorumlayıcısına bazı örnek verileri tanımlamak ve yüklemek için Liste 9-1'i kullanalım

In [135]: import pandas

In [136]: example\_data = [(1){'column1': 1, 'column2': 2},

...: {'column1': 10, 'column2': 32},

...: {'column1': 3, 'column2': 58}]

In [137]: (2)pandas.DataFrame(example\_data)

Out[137]:

column1 column2

0     1     2

1     10     32

2     3     58

Listing 9-1: Loading data into pandas directly

Burada example\_data adını verdiğimiz bazı verileri Python sözlükleri listesi olarak tanımlıyoruz (1). Bu dikt listesini oluşturduktan sonra, karşılık gelen pandalar DataFrame'i almak için bunu DataFrame yapıcısına (2) iletiriz. Bu işaretlerin her biri, sonuçta elde edilen DataFrame'de bir satır haline gelir. Diktlerdeki anahtarlar (sütun1 ve sütun2) sütun haline gelir. Bu, verileri doğrudan pandalara yüklemenin bir yoludur. Ayrıca harici CSV dosyalarından da veri yükleyebilirsiniz. Bu bölümün veri kümesini yüklemek için Liste 9-2'deki kodu kullanalım (sanal makinede veya bu kitapla birlikte verilen veri ve kod arşivinde mevcuttur).

import pandas

malware = pandas.read\_csv("malware\_data.csv")

Listing 9-2: Loading data into pandas from an external CSV file

Malware\_data.csv dosyasını içe aktardığınızda, ortaya çıkan kötü amaçlı nesne şuna benzer şekilde görünmelidir:

positives     size     type     fs\_bucket

0     45                 251592 trojan     2017-01-05 00:00:00

1     32                 227048 trojan     2016-06-30 00:00:00

2     53                 682593 worm 2016-07-30 00:00:00

3     39                     774568  trojan 2016-06-29 00:00:00

4     29                     571904 trojan 2016-12-24 00:00:00

5     31                     582352 trojan 2016-09-23 00:00:00

6     50                     2031661 worm 2017-01-04 00:00:00

Artık kötü amaçlı yazılım veri kümemizden oluşan bir Pandalar DataFrame'imiz var. Dört sütunu vardır: pozitifler (bu örnek için 57 antivirüs motorundan antivirüs tespitlerinin sayısı), boyut (kötü amaçlı yazılım ikili programının diskte kapladığı bayt sayısı), tür (Truva atı, solucan gibi kötü amaçlı yazılım türü) ve benzeri) ve fs\_bucket (bu kötü amaçlı yazılımın ilk görüldüğü tarih).

-Working with a pandas DataFrame:Pandalar DataFrame ile çalışma

Verilerimiz artık bir pandanın DataFrame'inde olduğuna göre, Listing 9-3'te gösterildiği gibi define () yöntemini çağırarak verilerimize nasıl erişilip değiştirileceğine bakalım.

In [51]: malware.describe()

Out[51]:

positives         size

count     37511.000000     3.751100e+04

mean     39.446536         1.300639e+06

std         15.039759         3.006031e+06

min         3.000000         3.370000e+02

25%         32.000000     1.653960e+05

50%         45.000000     4.828160e+05

75%         51.000000     1.290056e+06

max         57.000000     1.294244e+08

Listing 9-3: Calling the describe() method

Liste 9-3'te gösterildiği gibi, define () yöntemini çağırmak DataFrame'imiz hakkında bazı yararlı istatistikleri gösterir. İlk satır, sayım, boş olmayan pozitif satırların toplam sayısını ve boş olmayan satırların toplam sayısını sayar. İkinci satır, ortalama veya örnek başına ortalama pozitif sayısını ve kötü amaçlı yazılım örneklerinin ortalama boyutunu verir. Daha sonra hem pozitifler hem de boyut için standart sapma ve veri kümesindeki tüm örneklerde her bir sütunun minimum değeri gelir. Son olarak, sütunların her biri için yüzdelik değerleri ve sütunlar için maksimum değeri görüyoruz

Kötü amaçlı yazılım DataFrame'deki pozitif öğeler sütunu gibi sütunlardan birinin verilerini almak istediğimizi varsayalım (örneğin, her dosyanın sahip olduğu ortalama algılama sayısını görüntülemek veya pozitiflerin dağıtımını gösteren bir histogram çizmek için) veri kümesi). Bunu yapmak için, Listing 9-4'te gösterildiği gibi, pozitifler sütununu bir sayı listesi olarak döndüren kötü amaçlı yazılım ['pozitifler'] yazıyoruz.

In [3]: malware['positives']

Out[3]:

0     45

1     32

2     53

3     39

4     29

5     31

6     50

7     40

8     20

9     40

--snip--

Listing 9-4: Returning the positives column

Bir sütunu aldıktan sonra, istatistiklerini doğrudan hesaplayabiliriz. Örneğin, kötü amaçlı yazılım malware['pozitif'] ortalama mean () sütunun ortalamasını hesaplar, kötü amaçlı yazılım ['pozitifler'] max () maksimum değeri hesaplar, kötü amaçlı yazılım ['pozitifler'] min () minimum değeri hesaplar ve kötü amaçlı yazılım [ 'pozitifler'] std () standart sapmayı hesaplar. 9-5 listesi, her birinin örneklerini gösterir.

In [7]: malware['positives'].mean()

Out[7]: 39.446535682866362

In [8]: malware['positives'].max()

Out[8]: 57

In [9]: malware['positives'].min()

Out[9]: 3

In [10]: malware['positives'].std()

Out[10]: 15.039759380778822

Listing 9-5: Calculating the mean, maximum, and minimum values and the standard

deviation:Liste 9-5: Ortalama, maksimum ve minimum değerleri ve standart sapmayı hesaplama

Daha ayrıntılı analiz yapmak için verileri de bölebiliriz. Örneğin, Liste 9-6 truva atı, bitcoin ve solucan kötü amaçlı yazılım türleri için ortalama pozitifleri hesaplar.

In [67]: malware[malware['type'] == 'trojan']['positives'].mean()

Out[67]: 33.43822473365119

In [68]: malware[malware['type'] == 'bitcoin']['positives'].mean()

Out[68]: 35.857142857142854

In [69]: malware[malware['type'] == 'worm']['positives'].mean()

Out[69]: 49.90857904874796

Listing 9-6: Calculating the average detection rates of different malwares:Liste 9-6: Farklı kötü amaçlı yazılımların ortalama tespit oranlarının hesaplanması

İlk olarak, aşağıdaki gösterimi kullanarak türün truva atına ayarlandığı DataFrame satırlarını seçiyoruz: kötü amaçlı yazılım [kötü amaçlı yazılım ['tür'] == 'truva atı'] malware[malware['type'] == 'trojan']. Elde edilen verilerin pozitifler sütununu seçmek ve ortalamayı hesaplamak için bu ifadeyi şu şekilde genişletiyoruz: kötü amaçlı yazılım [kötü amaçlı yazılım ['tür'] == 'truva atı' ['pozitifler'] ortalama () malware[malware['type'] == 'trojan' ['positives'] mean(). 9-6 listesi ilginç bir sonuç veriyor, bu da solucanların bitcoin madenciliği ve Truva atı kötü amaçlı yazılımlarından daha sık tespit edilmesidir.

Ortalama olarak 49.9> 35.8 ve 33.4 olduğu için, kötü niyetli solucan örnekleri (49.9), kötü niyetli bitcoin ve truva atı örneklerinden (35.8, 33.4) daha fazla satıcı tarafından tespit edilir.

-Filtering Data Using Conditions:Koşulları Kullanarak Verileri Filtreleme

Diğer koşulları da kullanarak verilerin bir alt kümesini seçebiliriz. Örneğin, verileri filtrelemek için kötü amaçlı yazılım dosya boyutu gibi sayısal veriler üzerinde "büyüktür" ve "küçüktür" stil koşullarını kullanabilir ve ardından ortaya çıkan alt kümelerdeki istatistikleri hesaplayabiliriz. Virüsten koruma motorlarının etkinliğinin dosya boyutuyla ilgili olup olmadığını öğrenmekle ilgileniyorsak bu yararlı olabilir. Bunu Liste 9-7'deki kodu kullanarak kontrol edebiliriz.

In [84]: malware[malware['size'] > 1000000]['positives'].mean()

Out[84]: 33.507073192162373

In [85]: malware[malware['size'] > 2000000]['positives'].mean()

Out[85]: 32.761442050415432

In [86]: malware[malware['size'] > 3000000]['positives'].mean()

Out[86]: 27.20672682526661

In [87]: malware[malware['size'] > 4000000]['positives'].mean()

Out[87]: 25.652548725637182

In [88]: malware[malware['size'] > 5000000]['positives'].mean()

Out[88]: 24.411069317571197

Listing 9-7: Filtering the results by malware file size

Önceki koddaki ilk satırı alın: ilk olarak, DataFrame'imizi yalnızca bir milyonun üzerinde boyuta sahip örneklere göre alt gruplara ayırıyoruz (malware[malware['size'] > 1000000]).. Sonra pozitifler sütununu alıp ortalamayı hesaplıyoruz

(['positives'].mean())mean = ortalama yaklaşık 33,5. Bunu giderek artan dosya boyutları için yaptığımızda, her grup için ortalama algılama sayısının azaldığını görüyoruz. Bu, kötü amaçlı yazılım dosya boyutu ile bu kötü amaçlı yazılım örneklerini tespit eden ortalama antivirüs motoru sayısı arasında gerçekten bir ilişki olduğunu keşfettik ve bu ilginç ve daha fazla araştırmayı hak ediyor. Bunu matplotlib ve seaborn kullanarak görsel olarak inceleyeceğiz.

\*Using matplotlib to Visualize Data:Verileri Görselleştirmek için matplotlib kullanma

Python veri görselleştirmesi için gidilecek kitaplık matplotlib'dir; aslında, diğer çoğu Python görselleştirme kitaplığı, esasen matplotlib etrafında kullanışlı sarmalayıcılardır. Matplotlib'i pandalarla kullanmak kolaydır: grafiğini çizmek istediğimiz verileri elde etmek, dilimlemek ve parçalara ayırmak için pandaları kullanırız ve onu grafiklendirmek için matplotlib kullanırız. Amaçlarımız için en kullanışlı matplotlib işlevi plot işlevidir. Şekil 9-3, çizim işlevinin neler yapabileceğini gösterir.

Şekil 9-3: Kötü amaçlı yazılım örneklerinin boyutları ve antivirüs tespitlerinin sayısı grafiği

Burada, kötü amaçlı yazılım veri setimizin pozitiflerini ve boyut özelliklerini çiziyorum. Bir önceki bölümde pandalarla ilgili tartışmamızın da işaret ettiği gibi ilginç bir sonuç ortaya çıkıyor. Küçük dosyaların ve çok büyük dosyaların, bu dosyaları tarayan 57 antivirüs motorunun çoğu tarafından nadiren tespit edildiğini göstermektedir. Ancak, orta büyüklükteki dosyalar (yaklaşık 104,5–107) çoğu motor tarafından algılanır. Bunun nedeni, küçük dosyaların motorların kötü amaçlı olduklarını belirlemelerine izin verecek kadar yeterli bilgi içermemesi ve büyük dosyaların çok yavaş taranması, birçok antivirüs sisteminin onları tararken punt yapmasına neden olabilir.

\*Plotting the Relationship Between Malware Size and Vendor Detections:Kötü Amaçlı Yazılım Boyutu ve Satıcı Algılamaları Arasındaki İlişkiyi Çizme

Liste 9-8'deki kodu kullanarak Şekil 9-3'te gösterilen grafiği nasıl yapacağımızı inceleyelim.

(1) import pandas

from matplotlib import pyplot

malware = (2)pandas.read\_csv("malware\_data.csv")

pyplot.plot((3)malware['size'], (4)malware['positives'],

(5)'bo', (6)alpha=0.01)

pyplot.xscale((7)"log")

(8) pyplot.ylim([0,57])

pyplot.xlabel("File size in bytes (log base-10)")

pyplot.ylabel("Number of detections")

pyplot.title("Number of Antivirus Detections Versus File Size")

(9) pyplot.show()

Listing 9-8: Visualizing data using the plot() function

Gördüğünüz gibi, bu grafiği oluşturmak çok fazla kod gerektirmiyor. Her bir satırın ne yaptığını inceleyelim. İlk olarak, pandalar ve matplotlib kitaplığının pyplot modülü dahil olmak üzere gerekli kitaplıkları içe aktarıyoruz (1). Daha sonra, daha önce öğrendiğiniz gibi, kötü amaçlı yazılım veri kümemizi pandalar DataFrame'e yükleyen read\_csv işlevini (2) çağırırız.

Daha sonra plot () işlevini çağırıyoruz. İşleve yönelik ilk argüman, kötü amaçlı yazılım boyutu verileridir (3) ve sonraki argüman, kötü amaçlı yazılım pozitif verileri (4) veya her kötü amaçlı yazılım örneği için pozitif tespitlerin sayısıdır. Bu argümanlar, matplotlib'in çizeceği verileri tanımlar, ilk argüman x ekseni üzerinde gösterilecek verileri ve ikincisi y ekseninde gösterilecek verileri temsil eder. Bir sonraki argüman olan 'bo' (5), matplotlib'e veriyi temsil etmek için hangi renk ve şeklin kullanılacağını söyler.

Son olarak, alfa veya dairelerin şeffaflığını 0,1 (6) olarak ayarladık, böylece daireler birbiriyle tamamen örtüşse bile verilerin grafiğin farklı bölgelerinde ne kadar yoğun olduğunu görebiliriz.

Not:B in bo maviyi, o ise çemberi temsil etmektedir, yani matplotlib'e verilerimizi temsil etmesi için mavi daireler çizmesini söylüyoruz. Deneyebileceğiniz diğer renkler yeşil (g), kırmızı (r), camgöbeği (c), macenta (m), sarı (y), siyah (k) ve beyazdır (w). Deneyebileceğiniz diğer şekiller, bir nokta (.), Veri noktası başına tek bir piksel (,), bir kare (ler) ve bir beşgendir (p). Tüm ayrıntılar için, http: // matplotlib.org adresindeki matplotlib belgelerine bakın.

Plot () fonksiyonunu çağırdıktan sonra, x ekseninin ölçeğini logaritmik (7) olacak şekilde ayarladık. Bu, kötü amaçlı yazılım boyutu verilerini 10'un katları cinsinden görüntüleyeceğimiz ve çok küçük ve çok büyük dosyalar arasındaki ilişkileri görmeyi kolaylaştıracağımız anlamına gelir.

Verilerimizi grafiğe döktüğümüze göre, eksenlerimizi etiketler ve arsamıza başlık koyarız. X ekseni, kötü amaçlı yazılım dosyasının boyutunu ("Bayt cinsinden dosya boyutu (log base-10)"), y ekseni ise algılama sayısını ("Algılama sayısı") temsil eder. Analiz ettiğimiz 57 antivirüs motoru olduğundan, y ekseni ölçeğini (8) 'de 0 ila 57 aralığına ayarladık. Son olarak, grafiği görüntülemek için show () işlevini (9) çağırıyoruz. Grafiği bunun yerine bir resim olarak kaydetmek istersek, bu çağrıyı pyplot.savefig ("myplot.png") ile değiştirebiliriz.

Şimdi bir ilk örneği inceledik, şimdi başka bir örnek yapalım.

\*Plotting Ransomware Detection Rates:Fidye Yazılımı Algılama Oranlarını Çizme:

Bu sefer, bu bölümün başında gösterdiğim fidye yazılımı tespit planı olan Şekil 9-1'i yeniden oluşturmayı deneyelim. 9-9 listesi, zaman içindeki fidye yazılımı tespitlerimizi gösteren tüm kodu gösterir.

import dateutil

import pandas

from matplotlib import pyplot

malware = pandas.read\_csv("malware\_data.csv")

malware['fs\_date'] = [dateutil.parser.parse(d) for d in malware['fs\_bucket']]

ransomware = malware[malware['type'] == 'ransomware']

pyplot.plot(ransomware['fs\_date'], ransomware['positives'], 'ro', alpha=0.05)

pyplot.title("Ransomware Detections Over Time")

pyplot.xlabel("Date")

pyplot.ylabel("Number of antivirus engine detections")

pyplot.show()

Listing 9-9: Plotting ransomware detection rates over time:Liste 9-9: Zaman içindeki fidye yazılımı algılama oranlarını çizme

Liste 9-9'daki kodun bir kısmı şimdiye kadar açıkladığım şeyden aşina olmalı ve bazıları olmayacak. Şimdi kodu satır satır gözden geçirelim:

import dateutil

Yararlı Python paketi tarihi, birçok farklı formattaki tarihleri kolayca ayrıştırmanızı sağlar. Dateutil'i içe aktarıyoruz çünkü tarihleri görselleştirebilmemiz için ayrıştırıyor olacağız.

import pandas

from matplotlib import pyplot

Ayrıca pandaların yanı sıra matplotlib kütüphanesinin pyplot modülünü de içe aktarıyoruz.

malware = pandas.read\_csv("malware\_data.csv")

malware['fs\_date'] = [dateutil.parser.parse(d) for d in malware['fs\_bucket']]

ransomware = malware[malware['type'] == 'ransomware']

Bu satırlar veri kümemizi okur ve sadece fidye yazılımı örnekleri içeren fidye yazılımı adı verilen filtrelenmiş bir veri kümesi oluşturur, çünkü burada çizmek istediğimiz veri türü budur.

pyplot.plot(ransomware['fs\_date'], ransomware['positives'], 'ro', alpha=0.05)

pyplot.title("Ransomware Detections Over Time")

pyplot.xlabel("Date")

pyplot.ylabel("Number of antivirus engine detections")

pyplot.show()

Bu beş satır kod, Liste 9-8'deki kodu yansıtır: veriyi çizer, arsa başlığını koyar, x ve y eksenlerini etiketler ve ardından her şeyi ekrana dönüştürürler (bkz. Şekil 9-4). Yine, bu grafiği diske kaydetmek istersek, pyplot.show () çağrısını pyplot.savefig ("myplot.png") ile değiştirebiliriz.

Şekil 9-4: Zaman içinde fidye yazılımı tespitlerinin görselleştirilmesi

Plot () işlevini kullanarak bir grafik daha deneyelim.

\*Plotting Ransomware and Worm Detection Rates:Fidye Yazılımı ve Solucan Algılama Oranlarını Çizme

Bu sefer, zaman içindeki fidye yazılımı algılamalarını çizmek yerine, solucan algılamalarını da aynı grafikte çizelim. Şekil 9-5'te netleşen şey, antivirüs endüstrisinin solucanları tespit etmede (daha eski bir kötü amaçlı yazılım trendi) fidye yazılımdan (daha yeni bir kötü amaçlı yazılım trendi) daha iyi olduğudur.

Bu grafikte, zaman içinde (x ekseni) kaç antivirüs motorunun kötü amaçlı yazılım örnekleri (y ekseni) tespit ettiğini görüyoruz. Her bir kırmızı nokta bir tür = "fidye yazılımı" type="ransomware" kötü amaçlı yazılım örneğini temsil ederken, her mavi nokta bir tür = "solucan"  type="worm" örneğini temsil eder. Ortalama olarak, fidye yazılımı örneklerinden daha fazla motorun solucan örneklerini algıladığını görebiliriz. Bununla birlikte, her iki örneği de tespit eden motorların sayısı zaman içinde yavaş yavaş artmaktadır.

Liste 9-10, bu arsa plot  yapmak için kodu gösterir

import dateutil

import pandas

from matplotlib import pyplot

malware = pandas.read\_csv("malware\_data.csv")

malware['fs\_date'] = [dateutil.parser.parse(d) for d in malware['fs\_bucket']]

ransomware = malware[malware['type'] == 'ransomware']

worms = malware[malware['type'] == 'worm']

pyplot.plot(ransomware['fs\_date'], ransomware['positives'],

'ro', label="Ransomware", markersize=3, alpha=0.05)

pyplot.plot(worms['fs\_date'], worms['positives'],

'bo', label="Worm", markersize=3, alpha=0.05)

pyplot.legend(framealpha=1, markerscale=3.0)

pyplot.xlabel("Date")

pyplot.ylabel("Number of detections")

pyplot.ylim([0, 57])

pyplot.title("Ransomware and Worm Vendor Detections Over Time")

pyplot.show()

Liste 9-10: Zaman içindeki fidye yazılımı ve solucan algılama oranlarının grafiğini çizme

9-10 Listesinin ilk kısmına bakarak kodun üzerinden geçelim:

import dateutil

import pandas

from matplotlib import pyplot

malware = pandas.read\_csv("malware\_data.csv")

malware['fs\_date'] = [dateutil.parser.parse(d) for d in malware['fs\_bucket']]

ransomware = malware[malware['type'] == 'ransomware']

(1) worms = malware[malware['type'] == "worm"]

--snip--

Kod, önceki örneğe benzer. Şimdiye kadarki fark, verilerimizin (1) solucan filtreli sürümünü, fidye yazılımı filtrelenmiş verileri oluşturduğumuz aynı yöntemi kullanarak oluşturmamızdır. Şimdi kodun geri kalanına bir göz atalım:

--snip--

(1) pyplot.plot(ransomware['fs\_date'], ransomware['positives'],

' ro', label="Ransomware", markersize=3, alpha=0.05)

(2) pyplot.plot(worms['fs\_bucket'], worms['positives'],

'bo', label="Worm", markersize=3, alpha=0.05)

(3) pyplot.legend(framealpha=1, markerscale=3.0)

pyplot.xlabel("Date")

pyplot.ylabel("Number of detections")

pyplot.ylim([0,57])

pyplot.title("Ransomware and Worm Vendor Detections Over Time")

pyplot.show()

pyplot.gcf().clf()

Bu kod ile Liste 9-9 arasındaki temel fark, plot () işlevini iki kez çağırmamızdır: bir kez ro seçiciyi kullanan fidye yazılımı verileri için

(1) kırmızı daireler oluşturmak ve bir kez daha solucan verileri üzerinde bo

solucan verileri için mavi daireler oluşturmak için seçici (2). Üçüncü bir veri kümesini çizmek istersek, bunu da yapabiliriz. Ayrıca, Liste 9-9'dan farklı olarak, burada (3) 'te, mavi işaretlerin solucan zararlı yazılımını ve kırmızı işaretlerin fidye yazılımını temsil ettiğini gösteren figürümüz için bir açıklama oluşturuyoruz. Framealpha parametresi, göstergenin arka planının ne kadar yarı saydam olduğunu belirler (1'e ayarlayarak, tamamen opak hale getiririz) ve parametre işaretleyici ölçeği, göstergedeki işaretleyicilerin boyutunu ölçekler (bu durumda, üç faktör ile) .

Bu bölümde, matplotlib'de bazı basit olayların nasıl yapıldığını öğrendiniz. Ancak dürüst olalım, muhteşem değiller. Bir sonraki bölümde, grafiklerimize daha profesyonel bir görünüm vermemize ve daha karmaşık görselleştirmeleri hızlı bir şekilde uygulamamıza yardımcı olacak başka bir çizim kitaplığı kullanacağız.

\*Using seaborn to Visualize Data:Verileri Görselleştirmek için seaborn'u kullanma:

Artık pandalar ve matplotlib'i tartıştığımıza göre, aslında matplotlib üzerine inşa edilmiş ancak daha ince bir konteynere sarılmış bir görselleştirme kütüphanesi olan seaborn'a geçelim. Grafiklerimizi biçimlendirmek için yerleşik temaların yanı sıra daha karmaşık analizler gerçekleştirirken zaman kazandıran önceden hazırlanmış üst düzey işlevler içerir. Bu özellikler, sofistike, güzel grafikler oluşturmayı basit ve kolay hale getirir. Seaborn'u keşfetmek için, veri kümemizde her bir kötü amaçlı yazılım türü için kaç örnek bulunduğunu gösteren bir çubuk grafik oluşturarak başlayalım (bkz. Şekil 9-6)

Şekil 9-6: Bu bölümün veri kümesindeki farklı kötü amaçlı yazılım türlerinin çubuk grafik çizimi

9-11 listesi, bu grafiği yapmak için gereken kodu gösterir.

import pandas

from matplotlib import pyplot

import seaborn

(1)malware = pandas.read\_csv("malware\_data.csv")

(2) seaborn.countplot(x='type', data=malware)

(3) pyplot.show()

9-11 Listesi: Türe göre kötü amaçlı yazılım sayılarının bir çubuk grafiğini oluşturma:Listing 9-11: Creating a bar chart of malware counts by type

Bu kodda, verilerimizi önce pandas.read\_csv (1) aracılığıyla okuruz ve ardından, DataFrame'imizde (2) tür sütununda bir barplot oluşturmak için seaborn’un countplot işlevini kullanırız. Son olarak, (3) 'te pyplot'un show () yöntemini çağırarak arsa görünmesini sağlıyoruz. Seaborn'un matplotlib'i kapladığını hatırlayın, bu da matplotlib'den denizornu figürlerimizi göstermesini istememiz gerektiği anlamına geliyor. Şimdi daha karmaşık bir örnek grafiğe geçelim.

\*Plotting the Distribution of Antivirus Detections:Antivirüs Algılamalarının Dağılımını Çizme

Aşağıdaki taslağın temeli şu şekildedir: çoğu antivirüs motoru tarafından kötü amaçlı yazılımın yüzde kaçının gözden kaçırıldığını ve çoğu motor tarafından tespit edilen yüzde kaçının olduğunu anlamak için veri kümemizdeki kötü amaçlı yazılım örneklerinde antivirüs tespitlerinin dağılımını (sıklığını) anlamak istediğimizi varsayalım .

Bu bilgiler bize ticari antivirüs endüstrisinin etkinliği hakkında bir fikir veriyor. Bunu, Şekil 9-7'de gösterildiği gibi, her bir algılama sayısı için bu sayıda algılamaya sahip kötü amaçlı yazılım örneklerinin oranını gösteren bir çubuk grafik (histogram) çizerek yapabiliriz.

Şekil 9-7: Antivirüs tespitlerinin dağıtımının görselleştirilmesi (pozitifler)

Bu şeklin x ekseni, toplam 57 antivirüs motorunun kaç tanesinin onları tespit ettiğine göre sıralanmış kötü amaçlı yazılım örnekleri kategorilerini temsil eder. Bir örnek 57 motorun 50'si tarafından zararlı olarak tespit edilmişse 50'ye yerleştirilir, 57 motordan sadece 10'u tarafından tespit edilmişse 10 kategorisine girer. Her çubuğun yüksekliği, o kategoride kaç toplam örnek olduğu ile orantılıdır.

Grafik, 57 antivirüs motorumuzun çoğu tarafından birçok kötü amaçlı yazılım örneğinin tespit edildiğini (grafiğin en sağ üst bölgesindeki frekanslardaki büyük artışla gösterilmiştir), ancak aynı zamanda önemli bir azınlığın da az sayıda örnek tarafından tespit edildiğini açıkça ortaya koymaktadır. motorların (grafiğin en solunda gösterilmiştir).

Bu veri kümesini oluşturmak için kullandığım metodoloji nedeniyle beşten az motor tarafından tespit edilen örnekleri göstermiyoruz: Kötü amaçlı yazılımı beş veya daha fazla antivirüs motorunun algıladığı örnekler olarak tanımlıyorum. Yalnızca 5–30 tespit alan önemli sayıda örnek içeren bu grafiklendirilmiş sonuç, kötü amaçlı yazılım tespitinde motorlar arasında hala önemli anlaşmazlıklar olduğunu göstermektedir. 57 motordan 10'u tarafından kötü amaçlı yazılım olarak tespit edilen bir örnek, ya 47 motorun bunu tespit edemediğini ya da 10'unun bir hata yaptığını ve zararsız bir dosyada yanlış pozitif verdiğini gösteriyor. İkinci olasılık çok düşüktür, çünkü antivirüs satıcılarının ürünleri çok düşük yanlış pozitif oranlarına sahiptir: Çoğu motorun bu örnekleri kaçırması çok daha muhtemeldir.

Bu grafiği oluşturmak, Liste 9-12'de gösterildiği gibi sadece birkaç satırlık çizim kodu gerektirir.

import pandas

import seaborn

from matplotlib import pyplot

malware = pandas.read\_csv("malware\_data.csv")

(1) axis = seaborn.distplot(malware['positives'])

(2) axis.set(xlabel="Number of engines detecting each sample (out of 57)",

ylabel="Amount of samples in the dataset",

title="Commercial Antivirus Detections for Malware")

pyplot.show()

Listing 9-12: Plotting distribution of positives:Liste 9-12: Pozitiflerin dağılımını çizme

Seaborn paketi, dağıtım grafikleri (histogramlar) oluşturmak için yerleşik bir işleve sahiptir ve bu nedenle tek yaptığımız, dağıtmak işlevini görüntülemek istediğimiz verileri, kötü amaçlı yazılım ['pozitifler'] (1) geçirmek oldu. Daha sonra arsa başlığını, x ekseni etiketini ve y ekseni etiketini yapılandırmak için seaborn tarafından döndürülen eksen nesnesini kullanırız (2). Şimdi iki değişkenli bir deniz kenarı grafiğini deneyelim: kötü amaçlı yazılım için pozitif tespitlerin sayısı (beş veya daha fazla tespit içeren dosyalar) ve bunların dosya boyutları. Bu komployu daha önce Şekil 9-3'te matplotlib ile oluşturduk, ancak seaborn'un ortak plot fonksiyonunu kullanarak daha çekici ve bilgilendirici bir sonuç elde edebiliriz. Şekil 9-8'de gösterilen ortaya çıkan olay örgüsü zengin bir şekilde bilgilendiricidir, ancak ilk başta anlamak biraz çaba gerektirir, bu yüzden hadi üzerinden geçelim.

Bu çizim, Şekil 9-7'de yaptığımız histograma benzer, ancak tek bir değişkenin çubuk yükseklikleri aracılığıyla dağılımını göstermek yerine, bu grafik iki değişkenin dağılımını gösterir (bir kötü amaçlı yazılım dosyasının boyutu, x ekseninde) ve y eksenindeki algılama sayısı) renk yoğunluğu yoluyla. Bölge ne kadar karanlıksa, o bölgede o kadar fazla veri bulunur. Örneğin, dosyaların en yaygın olarak yaklaşık 105,5 boyutunda ve yaklaşık 53 pozitif değerde olduğunu görebiliriz. Ana grafiklerin üst ve sağındaki alt noktalar, boyut ve tespit verilerinin frekanslarının düzleştirilmiş bir versiyonunu gösterir. tespitlerin dağılımını (önceki grafikte gördüğümüz gibi) ve dosya boyutlarını ortaya çıkarın.

Şekil 9-8: Kötü amaçlı dosya boyutlarının pozitif tespitlere karşı dağılımının görselleştirilmesi

Merkez arsa en ilginç olanıdır çünkü boyut ve pozitifler arasındaki ilişkiyi gösterir. Matplotlib ile Şekil 9-3'te olduğu gibi ayrı veri noktalarını göstermek yerine, genel eğilimi çok daha net bir şekilde gösterir. Bu, çok büyük kötü amaçlı yazılım dosyalarının (boyut 106 ve üzeri) antivirüs motorları tarafından daha az sıklıkla tespit edildiğini gösteriyor ve bu da bize bu tür kötü amaçlı yazılımları tespit etmede uzmanlaşmış bir çözüm oluşturmak isteyebileceğimizi söylüyor.

Bu arsayı oluşturmak için, Liste 9-13'te gösterildiği gibi, seaborn'a bir planlama çağrısı yapılması yeterlidir.

import pandas

import seaborn

import numpy

from matplotlib import pyplot

malware = pandas.read\_csv("malware\_data.csv")

(1) axis=seaborn.jointplot(x=numpy.log10(malware['size']),

y=malware['positives'],

kind="kde")

(2) axis.set\_axis\_labels("Bytes in malware file (log base-10)",

"Number of engines detecting malware (out of 57)")

pyplot.show()

9-13 Listesi: Kötü amaçlı dosya boyutlarının ve pozitif tespitlerin dağılımını çizme

Burada, DataFrame'imizdeki (1) pozitiflerin ve boyut sütunlarının ortak bir dağılım grafiğini oluşturmak için seaborn'un ortak plan işlevini kullanıyoruz. Ayrıca, biraz kafa karıştırıcı bir şekilde, seaborn'un eklem planı işlevi için eksenlerimizi etiketlemek için Liste 9-11'den farklı bir işlev çağırmalıyız: ilk argümanı x ekseni etiketi ve ikinci argümanı olan set\_axis\_labels () işlevi (2) y ekseni etiketidir.

\*Creating a Violin Plot:Keman Grafiği Oluşturma

Bu bölümde araştırdığımız son olay örgüsü türü, seaborn keman grafiğidir violin plot.Bu taslak, belirli bir değişkenin çeşitli kötü amaçlı yazılım türleri arasındaki dağılımını zarif bir şekilde keşfetmemizi sağlar. Örneğin, veri kümemizde kötü amaçlı yazılım türüne göre dosya boyutlarının dağılımını görmek istediğimizi varsayalım. Bu durumda Şekil 9-9 gibi bir arsa oluşturabiliriz.

Şekil 9-9: Kötü amaçlı yazılım türüne göre dosya boyutlarının görselleştirilmesi

Bu grafiğin y ekseninde, 10'un üsleri olarak temsil edilen dosya boyutları vardır. X ekseninde her kötü amaçlı yazılım türünü sıralarız. Gördüğünüz gibi, her bir dosya türünü temsil eden çubukların kalınlığı, farklı boyut seviyelerinde değişiklik gösterir ve bu, söz konusu kötü amaçlı yazılım türü için verinin ne kadarının o boyutta olduğunu gösterir. Örneğin, önemli sayıda çok büyük fidye yazılımı dosyası olduğunu ve bu solucanların daha küçük dosya boyutlarına sahip olma eğiliminde olduğunu görebilirsiniz - muhtemelen solucanlar bir ağda hızla yayılmayı hedefledikleri için solucan yazarları bu nedenle dosya boyutlarını küçültme eğilimindedir. Bu kalıpları bilmek, potansiyel olarak bilinmeyen dosyaları daha iyi sınıflandırmamıza yardımcı olabilir (daha büyük bir dosyanın fidye yazılımı olma olasılığı daha yüksektir ve bir solucan olma olasılığı daha düşüktür) veya belirli bir türü hedefleyen savunma aracında hangi dosya boyutlarına odaklanmamız gerektiğini bize öğretebilir. kötü amaçlı yazılım.

Keman grafiğini yapmak, Liste 9-14'te gösterildiği gibi bir çizim çağrısı gerektirir.

import pandas

import seaborn

from matplotlib import pyplot

malware = pandas.read\_csv("malware\_data.csv")

(1) axis = seaborn.violinplot(x=malware['type'], y=malware['size'])

(2) axis.set(xlabel="Malware type", ylabel="File size in bytes (log base-10)",

title="File Sizes by Malware Type", yscale="log")

(3) pyplot.show()

Listing 9-14: Creating a violin plot

Liste 9-14'te önce keman grafiğini (1) oluşturuyoruz. Daha sonra seaborn'a eksen etiketlerini ve başlığı ayarlamasını ve y eksenini log-scale (2) olarak ayarlamasını söylüyoruz. Son olarak, arsa görünür hale getiriyoruz (3). Şekil 9-10'da gösterildiği gibi, her kötü amaçlı yazılım türü için pozitiflerin sayısını gösteren benzer bir grafik de oluşturabiliriz.

Şekil 9-10: Kötü amaçlı yazılım türü başına virüsten koruma pozitiflerinin (algılamalarının) sayısının görselleştirilmesi

Şekil 9-9 ve Şekil 9-10 arasındaki tek fark, y eksenindeki dosya boyutuna bakmak yerine, her dosyanın aldığı pozitiflerin sayısına bakmamızdır. Sonuçlar bazı ilginç eğilimleri gösteriyor. Örneğin, fidye yazılımı neredeyse her zaman 30'dan fazla tarayıcı tarafından algılanır. Bunun aksine, bitcoin, trojan ve keylogger kötü amaçlı yazılım türleri, zamanın önemli bir kısmında 30'dan az tarayıcı tarafından algılanır, bu da bu türlerin çoğunun güvenlik endüstrisinin savunmasını aştığı anlamına gelir (tarayıcıları tespit edemeyen yüklenen bu dosyalar büyük olasılıkla bu örneklerden etkilenir). Liste 9-15, Şekil 9-10'da gösterilen grafiğin nasıl oluşturulacağını gösterir.

import pandas

import seaborn

from matplotlib import pyplot

malware = pandas.read\_csv("malware\_data.csv")

axis = seaborn.violinplot(x=malware['type'], y=malware['positives'])

axis.set(xlabel="Malware type", ylabel="Number of vendor detections",

title="Number of Detections by Malware Type")

pyplot.show()

Listing 9-15: Visualizing antivirus detections per malware type:Liste 9-15: Kötü amaçlı yazılım türüne göre virüsten koruma algılamalarını görselleştirme

Bu koddaki ve önceki koddaki tek fark, keman grafiği işlevine farklı veriler (malware['positives'] yerine  malware['size'])geçirmemiz, eksenleri farklı şekilde etiketlememiz, başlığı farklı şekilde ayarlamamız ve atlamamızdır. y ekseni ölçeğini log-10 olarak ayarlama.

-> ÖZET:

Bu bölümde, kötü amaçlı yazılım verilerinin görselleştirilmesinin, trend olan tehditler ve güvenlik araçlarının etkinliği hakkında makroskopik içgörüler elde etmenizi nasıl sağladığını öğrendiniz. Pandalar, matplotlib ve seaborn'u kendi görselleştirmelerinizi oluşturmak ve örnek veri kümeleri hakkında fikir edinmek için kullandınız.

Ayrıca pandalarda yararlı istatistikleri göstermek için define () gibi yöntemleri nasıl kullanacağınızı ve veri kümenizin alt kümelerini nasıl çıkaracağınızı öğrendiniz. Daha sonra, antivirüs tespitlerindeki gelişmeleri değerlendirmek, trend olan kötü amaçlı yazılım türlerini analiz etmek ve diğer daha geniş soruları yanıtlamak için kendi görselleştirmelerinizi oluşturmak için bu veri alt kümelerini kullandınız.

Bunlar, sahip olduğunuz güvenlik verilerini yeni araçların ve tekniklerin geliştirilmesine bilgi verebilecek eyleme geçirilebilir istihbarata dönüştüren güçlü araçlardır. Umarım veri görselleştirmeleri hakkında daha fazla bilgi edinir ve bunları kötü amaçlı yazılım ve güvenlik analizi iş akışınıza dahil edersiniz.