دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مدیریت، علم و فناوری

پروژه ۷ درس داده کاوی

شناسایی ناهنجاری‌ها بر روی دیتاست KDDCUP99

نگارش
فاطمه سادات علیخانی

استاد درس
دکتر مهدی قطعی

فهرست

۱ مقدمه

در یک دیتاست اگر یک سری داده رفتارهای نرمال نسبت به بقیه نداشته باشند در واقع میتوان آنها را به عنوان کاندیدای داده های ناهنجاری مشخص کرد، تشخیص داده های ناهنجاری در پروژه‌های مختلف به ما کمک میکند تا داده‌های خود را بهتر تحلیل کنیم یا داده هایی که رفتار متفاوت دارند را شناسایی کنیم مانند تشخیص زمانی که به شبکه یک سیستم حمله میشود (که در این پروژه بررسی میشود) در این گزارش روشهای مختلف تشخیص ناهنجاری را مانند isolation forest و mahalonobis distance و Univariate outlier detection را بررسی میکنیم و در انتها آن ها را با هم مقایسه میکنیم.

۲ معرفی دیتاست

در این گزارش از دیتا ست KDDCUP99-(http) که در سایت odds وجود دارد استفاده شده است.

دیتاست اصلی در سایت UCI machine learning است و این دیتاست شامل طیف گسترده‌ای از نفوذهای شبیه سازی شده به یک محیط نظامی است و برای این ساخته شده است که بتوان به وسیله آن اتصالات خوب را از اتصالات بد مانند نفوذ یا حملات اینترنتی تشخیص داد.

این دیتاست شامل ۴۱ ستون و ۴ میلیون سطر است . همینطور در این دیتاست نوع حملات نیز مشخص شده است که یکی از چهار حالت زیر است :

- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow'' attacks;
- probing: surveillance and other probing, e.g., port scanning.

با توجه به تعداد بسیار زیاد سطرها در بیشتر مقالات فقط بر روی درصدی از داده‌ها بررسی انجام شده است.

در سایت odds ۴ ستون از دیتاست مورد نظر انتخاب شده که آنها ستون‌های (service, duration, src_bytes, dst_bytes) می‌باشند. که توضیح عملکرد هر ستون در جدول زیر آمده است :

feature name	description	type
duration	length (number of seconds) of the connection	continuous
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous

و در سایت odds آمده است که ستون‌های duration , src_bytes , dst_bytes را تغییر داده و آن‌ها را به صورت log(x+0.1) تبدیل کرده است.

همینطور نمونه هایی که مقدار service آنها برابر http است و مقدار log_in آنها مثبت است انتخاب شده است که مجموع این دیتاست به صورت کلی شامل ۳ ستون و ۵۶۷۴۹۷ سطر است.

همینطور یک مقدار y دارد که به وسیله آن مشخص میشود که اطلاعات یک ردیف مربوط به داده‌ی نا هنجاری است یا خیر.

قسمتی از دیتاست به صورت شکل زیر است :

```
data = {'duration' : arrays['X'][0],'src_bytes' : arrays['X'][1],'dst_bytes' : arrays['X'][2]}
df = pd.DataFrame(data)
df
```

	duration	src_bytes	dst_bytes
0	-2.302585	5.371103	10.716107
1	-2.302585	5.088213	8.418058
2	-2.302585	5.464255	7.113224
3	-2.302585	5.451468	7.616825
4	-2.302585	5.476882	6.186414
...	...	...	...
567493	-2.302585	5.357058	7.735477
567494	-2.302585	5.389528	5.464255
567495	-2.302585	5.384954	8.191401
567496	-2.302585	5.389528	7.118097
567497	-2.302585	5.389528	7.001337

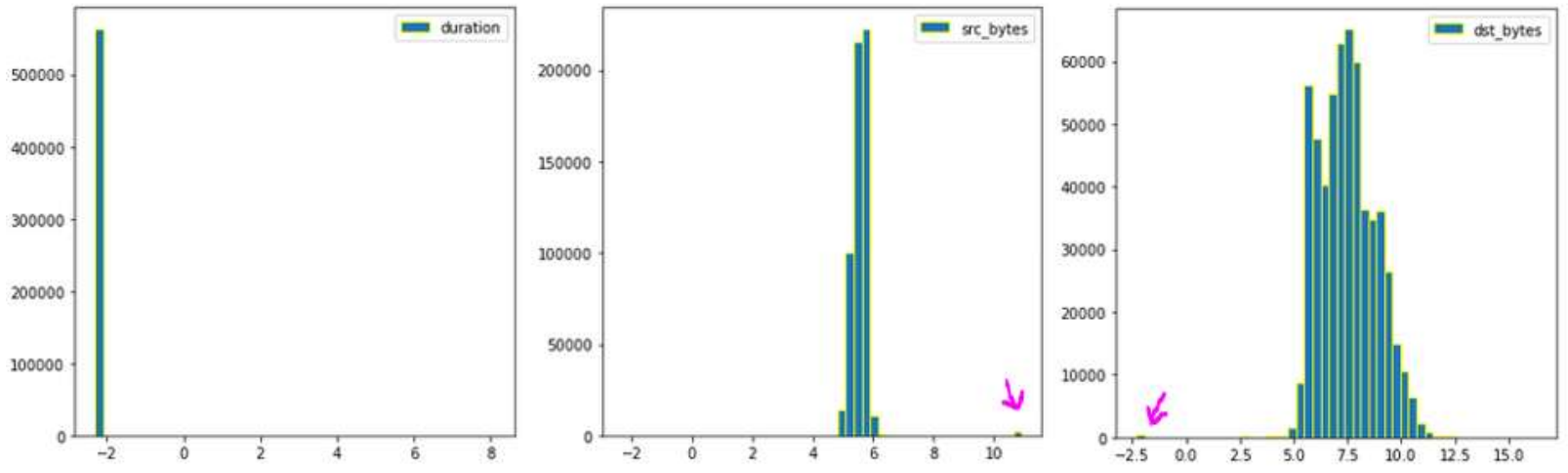
567498 rows × 3 columns

۳ مصور سازی

برای آشنا شدن با دیتاست، هسیتوگرام آن را کشیده و به صورت زیر است و مقدار bin size را برابر ۵۰ قرار داده

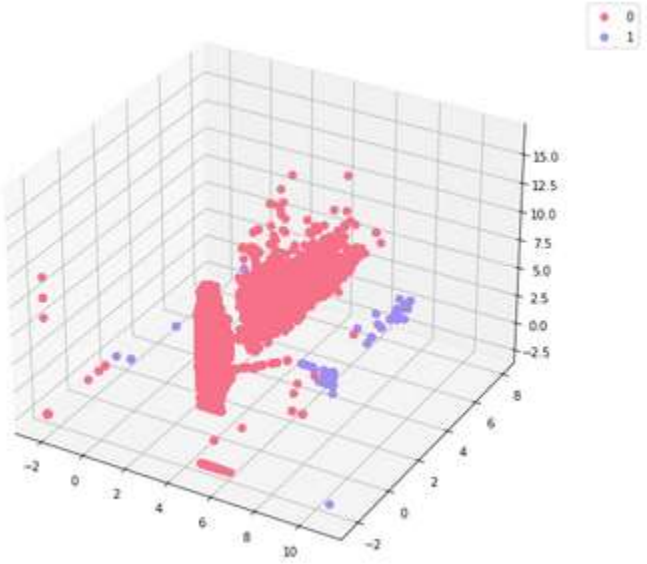
```
%matplotlib inline
plt.figure(figsize=(12,12))
columns = ['duration' , 'src_bytes','dst_bytes']
for i, c in enumerate(columns, 1):
    plt.subplot(1,3,i)
    plt.hist(df[c], bins=50, edgecolor="yellow" , label=c)

    plt.legend(loc='upper right')
```

همینطور که در شکل مشخص است برای ستون duration همه داده ها در یک قسمت خاص قرار دارد حتی با بیشتر کردن مقدار bin size نمودار تغیری نکرد، برای دو ستون دیگر قسمت علامت زده شده را که به طور کلی از بقیه دور است و شاید بتوان آنها را به عنوان داده outlier معرفی کنیم. اما این ها را بررسی نکرده و جلوتر روش‌های دیگری از شناسایی داده‌های پرت را بررسی می‌کنیم.



شکل ۱ هیستوگرام

همینطور با توجه به اینکه این دیتاست سه بعدی است و داده‌هایی که outlier هستند مشخص هستند نمودار scatter plot آن ها به صورت زیر است که در آن نقطه‌های بنفش داده های پرت هستند و بقیه داده‌های اصلی ما هستند.



شکل ۲ _ نمودار پراکندگی داده های پرت

۴ الگوریتم‌های شناسایی ناهنجاری

۴.۱ تشخیص نقاط پرت با استفاده از یک متغیر (Detection Univariate outliers)

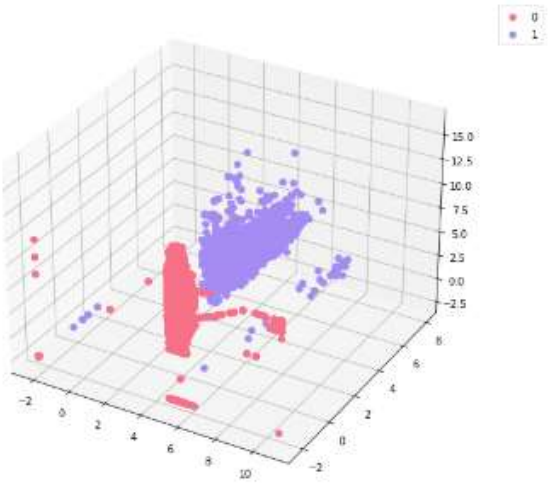
4.1.1 ستون duration

در این روش نقاط پرت را فقط بر اساس یک ستون بررسی می‌کنیم به این صورت مانند شکل زیر یک ستون را به فرم استاندارد (z-scor) در می‌آوریم :

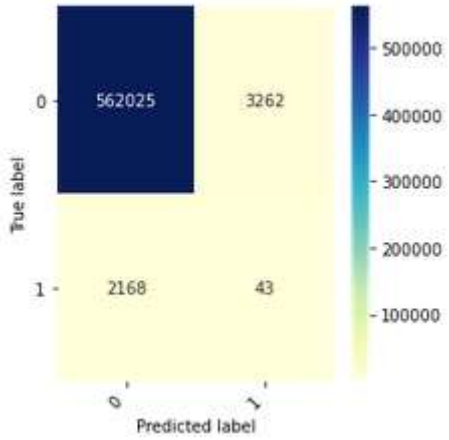
```
df2["normal_duration"] = (df["duration"] - df["duration"].mean())/df["duration"].std()
df2["normal_src_bytes"] = (df["src_bytes"] - df["src_bytes"].mean())/df["src_bytes"].std()
df2["normal_dst_bytes"] = (df["dst_bytes"] - df["dst_bytes"].mean())/df["dst_bytes"].std()
```

با این کار میانگین داده‌های هر ستون به صورت جداگانه برابر صفر میشود و انحراف از معیار آنها برابر یک می‌شود حال با توجه به اینکه ۹۸ درصد داده‌ها در قسمت mean ± 3* std قرار دارند میتوان داده‌هایی که در این بازه نیستند را به عنوان کاندیدای داده‌ی پرت معرفی کرد.

پس داده‌هایی که در هر ستون از ستون های نرمال مقدار قدر مطلق آن بیشتر از ۳ بود را میتوان کاندیدای داده‌ی پرت معرفی کرد.



برای ستون اول که duration است این کار را انجام می‌دهیم و شکل سه بعدی داده‌های پرت به صورت رو به رو میشود و همانطور که مشخص است بسیار متفاوت است نسبت به شکلی که در بخش مصور سازی برای نمایش داده های نا هنجاری و عادی دیدیم.



این مطلب که با نرمال سازی ستون duration و سپس بدست آوردن داده‌های پرت توسط آن، داده های پرت به درستی پیدا نمیشوند را از طریق ماتریس مقابل میتوان متوجه شد که از بین ۲۲۱۱ داده‌ی پرت فقط ۳۴ تا از آنها به درستی شناخته شده است و ۳۲۶۲ تا از داده های اصلی به عنوان داده اصلی شناخته شده‌اند.(۱: داده پرت ، ۰ : داده نرمال)

و به طور کلی مقدار دقت f1-score که به صورت زیر محاسبه می‌شود را بدست می آوریم این مقدار هم داده‌های ناهنجاری که اشتباه در نظر گرفته شده است را در نظر می‌گیرند و هم داده هایی که inlierهستند و به اشتباه به عنوان ناهنجاری شناخته میشوند.

```
F1-score = 2 * (precision * recall) / (precision + recall)
```

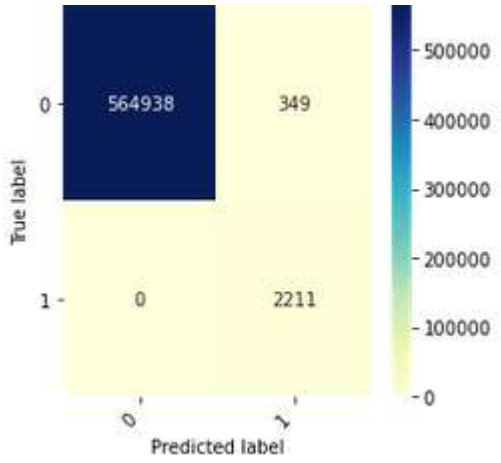
مقدار دقتی که برای ستون اول به دست آمده برابر یک صدم است که بسیار مقدار کمی است.

```
f1_score(arrays['y'][0], y_n_duration)
```

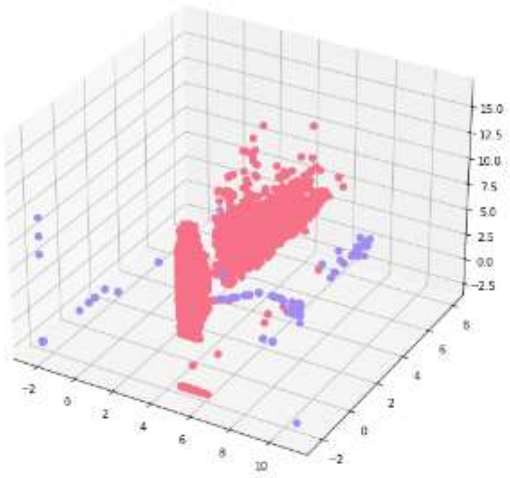
```
0.01559100797679478
```

۴.۱.۲ ستون src_bytes

بعد از آن سراغ ستون دوم می‌رویم و آنهایی که مقدار قدرمطلق نرمال شده‌ی ستون دوم آنها از ۳ بیشتر است را به عنوان داده پرت در نظر می‌گیریم



ماتریس مقابل این مطلب را نشان می‌دهد که بدست آوردن داده‌های پرت از این طریق بسیار بهتر از روش قبل است و در ان از مجموع ۲۲۱۱ داده پرت همگی به درستی تشخیص داده شده اند و داده ایی نبوده که ناهنجاری بوده باشد و به عنوان داده inlier شناخته شود این نکته بسیار مهمی است که همگی داده های ناهنجاری به درستی شناخته شوند ولی ۳۴۹ داده inlier به عنوان داده پرت شناخته شده اند.



شکل سه بعدی داده‌های پرت به صورت رو به رو میشود.

این شکل شبیه شکل ۲ است و میتوان حدس زد که داده‌های پرت موجود در این ستون به طور کلی مشابه داده‌های پرت به طور کلی است.

و مقدار دقت f1-score که در قسمت قبل محاسبه آن توضیح داده شد به صورت زیر است و همانطور که مشخص است این مقدار بسیار مناسب است:

```
f1_score(arrays['y'][0], y_src_bytes)
```

```
0.9268497170404527
```

۴.۱.۳ ستون dst_bytes

همین کار را برای ستون سوم انجام می‌دهیم و آنهایی که مقدار قدرمطلق نرمال شده‌ی ستون سوم آنها از ۳ بیشتر است را به عنوان داده پرت در نظر می‌گیریم و مقدار f1-score آنها به صورت زیر میشود و این مقدار کمتر از دو مقدار مشخص شده قبل است و بهترین مقدار را داده‌های پرت ستون src_bytes داشته‌اند.

```
f1_score(arrays['y'][0], y_col2n)
```

```
0.0017528483786152496
```

4.2 تشخیص نقاط پرت با استفاده از mahalanobis distances

۴.۲.۱ توضیح روش

در این روش به جای اینکه از یک ستون برای تشخیص نا هنجاری استفاده کنیم از همه ستون‌ها استفاده می‌کنیم این روش به صورت زیر محاسبه میشود :

$$\text{mahalanobis distances} = (x - \text{mean})^{T} (\text{covariance})^{-1} (x - \text{mean})$$

همینطور که مشخص است در این روش از همه ستون ها استفاده میشود.

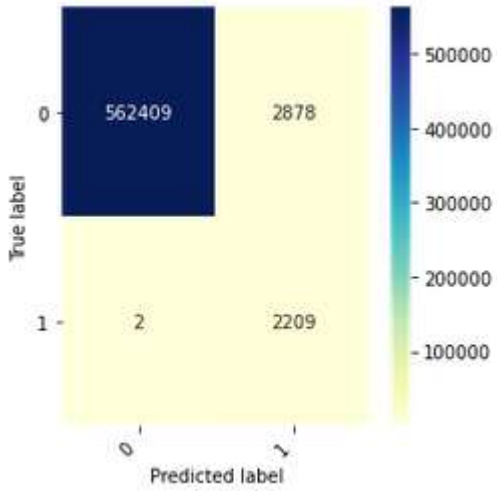
۴.۲.۲ پیاده سازی

برای پیاده سازی این روش از کتابخانه scipy استفاده کرده و مقدار این فاصله را حساب میکنیم و برای بدست آورن داده های پرت باید برای این فاصله ها یک threshold در نظر بگیریم از Chi-square distribution با درجه آزادی ۳ استفاده میکنیم و مقدار زیر بدست می آید. نمونه هایی که مقدار فاصله آنها از این مقدار بیشتر شد را به عنوان کاندید برای داده پرت در نظر می‌گیریم

```
chi2.ppf(0.96, df=3 )
```

```
8.606947404377836
```

۴.۲.۳ دقت الگوریتم



داده ها به صورت ماتریس رو به رو دسته بندی میشوند این ماتریس نشان میدهد که اکثر داده‌های پرت به درستی شناخته شده اند و فقط ۲ تا از آنها به عنوان داده inlier شناخته شده اما از بین داده های inlier نزدیک به ۲۰۰۰ تا از آنها به عنوان داده پرت شناخته شده اند

که این کمی باعث ایجاد مشکل میشود اما به طور کلی اگر هدف شناسایی داده های نا هنجاری و مقابله با آن باشد این روش تقریبا به خوبی عمل میکند.

همینطور مقدار f1-score محاسبه داده‌ی پرت در این الگوریتم به صورت زیر است :

```
f1_score(arrays['y'][0], mahalanobis_distances_out)
```

```
0.6053713346122226
```

این مقدار به طور کلی مقدار قابل قبولی است و از دو مقدار برای ستون اول و سوم بسیار بیشتر است.

۴.۳ شناسایی داده‌ی پرت به روش isolation forest

۴.۳.۱ توضیح الگوریتم

Isolation forest یک روش شناسایی داده پرت است که در آن از تعدادی درخت تصمیم(Decision Tree) استفاده شده است. در این روش یک ویژگی یا ستون به صورت تصادفی انتخاب میشود سپس یک مقدار بین بازه آن ستون مشخص میشود و یک خط با توجه به آن مقدار کشیده میشود و داده‌ها را به دو دسته تقسیم میکند. و این تقسیم بندی آنقدر تکرار میشود تا یک نمونه به صورت تک در یک محدوده قرار گیرد آن نمونه داده پرت شناخته میشود، واضح است داده هایی که نزدیک به هم هستند دیرتر در یک محدوده جدا از بقیه داده ها قرار میگیرند و داده های پرت و ناهنجاری که مشابه داده های دیگر نیستند زودتر از بقیه جدا میشوند و در درختی که تشکیل میشود در ارتفاع بالاتری به عنوان نود برگ شناخته می‌شود و به ریشه درخت نزدیک تر است.

۴.۳.۲ پیاده سازی

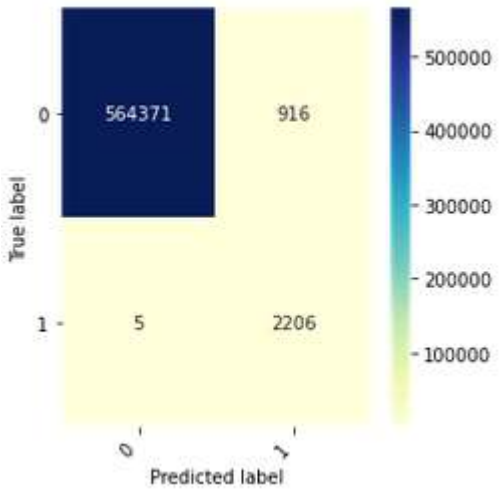
این روش را با استفاده از کتابخانه sklearn به صورت زیر پیاده سازی میکنیم:

```
clf = IsolationForest(contamination=.004)
```

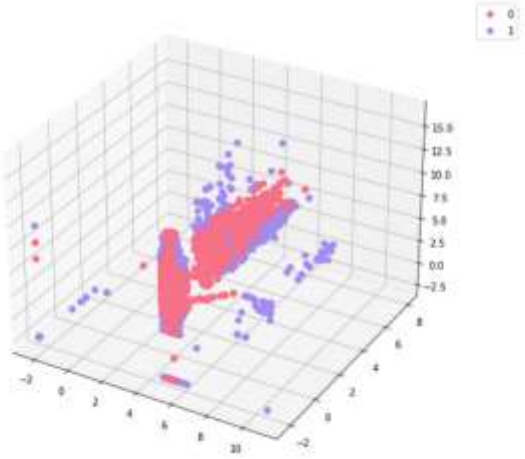
که در آن مقدار contamination درصد ناهنجاری ای است که باید الگوریتم در دیتاست پیدا کند و همانطور که در ابتدا توضیح داده شد مقدار ۴هزارم داده ها ناهنجاری هستند.

داده ها را به دو دسته train , test با نسبت ۷۰ و ۳۰ تقسیم بندی کرده و مدل را بر روی داده های train ، fit میکنیم و داده تست را با دقت ۰,۷۵ پیشبینی میکند.

۴.۳.۳ دقت الگوریتم



همنیطور ماتریس تشخیص داده‌های ناهنجاری به طور کلی به صورت رو به رو است همانطور که در این شکل میبینیم فقط ۶ داده ناهنجاری به درستی تشخیص داده نشده و بقیه آنها به طور کامل شناخته شده اند این الگوریتم برای زمانی که شناسایی داده های ناهنجاری و حملات شبکه ایی بسیار حساس است میتواند مفید باشد زیرا اکثر داده های ناهنجاری را به درستی شناخته است اما نزدیک به ۹۰۰ داده عادی را هم به عنوان داده نا هنجاری تشخیص داده است.



و شکل سه بعدی از پراکندگی نقاط ناهنجاری و inlier برای این الگوریتم به صورت زیر میباشد که تقریبا مشابه شکل-۲ که همان شکل اصلی داده هاست میباشد.

و نتیجه f1-score برای این داده‌ها به صورت زیر میباشد این مقدار از نتیجه بقیه مقادیر در الگوریتم های دیگر بسیار بهتر میباشد.

```
f1_score(arrays['y'][0], isolation_forest_out)
```

```
0.8273017063566472
```

5 نتیجه گیری و مقایسه روش ها

همینطور که در نمودار جمع بندی زیر میبینیم استفاده از روشهای تشخیص داده پرت تنها با یک ستون همیشه نمیتواند روش مناسبی باشد و بهتر از روشهایی که از چند ستون استفاده کرد از بین این روشها روش isolation forest نتیجه بسیار خوبی داشت و برای این دیتاست مناسب بود. و همانطور که دیدیم این روش اکثر داده‌های ناهنجاری را تشخیص داد این برای مواقعی که میخواهیم از رخداد حملات در شبکه های اینترنتی جلوگیری کنیم بسیار مهم است و میتوانیم جلوی اکثر این حملات را بگیریم.

روش	F1-score
نرمال سازی ستون duration	۰.۰۱۵
نرمال سازی ستون src_bytes	۰.۹۲۶
نرمال سازی ستون dst_bytes	۰.۰۰۱۷
Mahalanobis distance	۰.۶۰۵
Isolation forest	۰.۸۲۷

6 منابع و مراجع

(http (KDDCUP99) dataset, 2021)

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008.

(Multivariate Outlier Detection in Python, 2021)

(Univariate Outlier Detection in Python, 2021)

(Mahalanobis Distance, 2019)