COMP90073

Name: Katsuhide I

ID: 1132300

Task 1

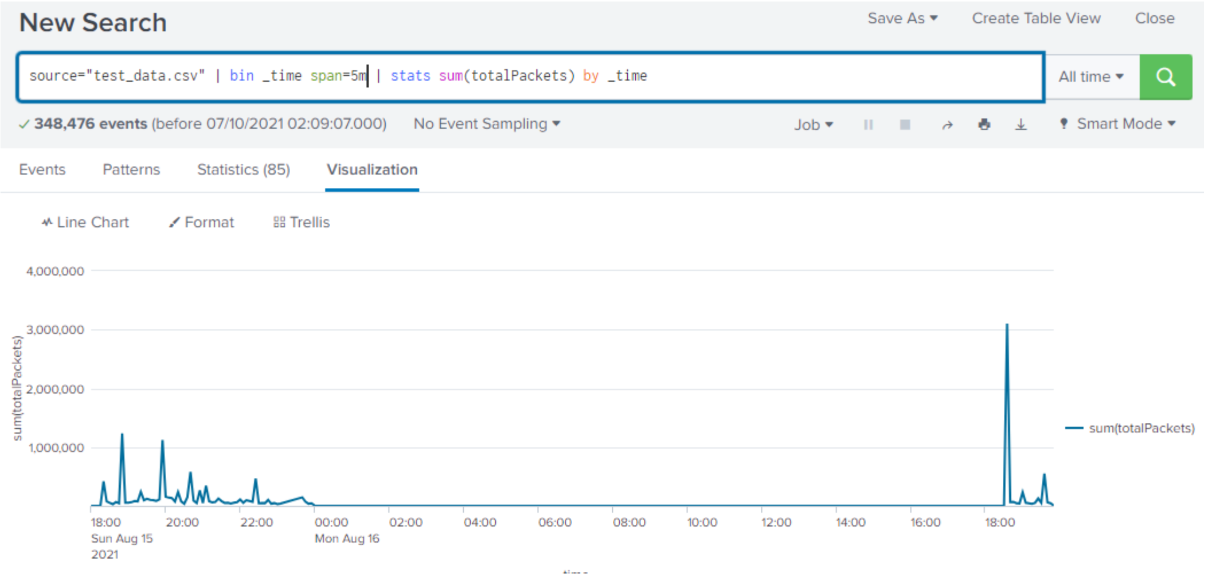
1. **Introduction and data pre-processing**
   We have three dataset CSV files: test_data.csv, training_data.csv, and validation_data_with_label.csv. Firstly, we needed to ingest the files for analysis by uploading the CSV files using Splunk functionality. Once the CSV files were loaded into Splunk, we could see that the dataset didn't contain headers for each column. Thus, we used the Splunk's extract new fields functionality to give names for the 15 fields to construct queries easily.

2. **Overview of the dataset and feature selection using Splunk**
   In order to select and generate features from the dataset, we need to understand the dataset well. We performed an exploratory analysis over the test datasets to get best results.
   The dataset is composed of three parts: training data, test data, and validation data. The training data contains 1,045,455 events, test data contains 348,477 events, and validation data contains 348,485 events with 15 fields. Duration, the number of total packets, the number of bytes transferred in both directions, and the number of bytes transferred from the source to the destination are numerical features, while others are categorical features. As steam ID is just a unique identifier of each record and timestamp is start-time of each transaction happened, they wouldn't help anomaly detection, so we will ignore these field for this analysis.

	duration	**totalPackets**	**bytesBothDir**	**bytesSrcToDst**
Average	515.21	38.41	24487.52	9327.71
Standard Deviation	1047.03	3794.58	2738257.87	899998.68
Min	0.00	0	59	0
Max	3,649.84	1,484,195	1,061,549,557	263,799,746
Missing Value	0	0	0	0

Figure 1: Avg, stdev, min, max of numerical fields

The figure 1 shows the average, standard deviation, minimum and maximum of each numerical field based on Splunk queries (see Appendix A). Standard deviation indicates the amount of data variation. These four numerical fields have large standard deviation because they have extreme values. This shows that there might have indications that there are anomalies in these fields. So, we decided to use these four numerical fields for the anomaly detection.



```
source="test_data.csv" | bin _time span=5m | stats sum(totalPackets) by _time
```

✓ 348,476 events (before 07/10/2021 02:09:07.000)

Figure 2: line chart of sum of totalPackets per 5mins

Also, as can be seen from the figure 2, we can see that the number of total packets increased significantly at 18:35 16/Aug. The reason that from around 00:00 to around 18:07 16/Aug the line chart is flat is because there is no timestamp in between in the test dataset.

	protocol	**srcIP**	**srcPort**	**direction**	**dstIP**	**dstPort**	**state**	**srcToS**	**dstToS**
Top	udp	150.35.87.210	771	<->	150.35.87.232	13363	CON	0.0	0.0
Top's Freq	277,486	10,581	3,263	261,013	241,249	37,807	260,812	347,769	291,219
Top's Freq (%)	79.63	3.04	0.94	74.90	69.23	10.85	74.92	99.81	100.00 (99.99…)
Num of unique values	8	207,030	64,271	6	969	7,370	233	4	3
Missing Value	0	0	0	0	0	0	378	39	57,246

Figure 3: Top, Top's frequency, number of unique items for categorical fields

The figure 3 shows the most frequent value, its frequency, and the number of unique values in the categorical field (see Appendix B). We also calculated the top's frequency in percentage by the frequency divided by the number of total events in dataset (348,477 events). We noticed that there are missing values in state, srcToS, and dstToS fields by using Splunk query (Appendix C).

From the figure 3, we can see that protocol, state, srcToS, dstTos fields have unbalanced distribution, which is good for anomaly detection. In the protocol field, there are 8 protocols: udp, icmp, tcp, rtcp, rtp, arp, esp, and udt. Most of the protocols used are udp, icmp, and tcp. The state field has 233 unique values and many of them are CON. The srcToS field and dstToS are very unbalanced and they have only few unique values. So, we decided to use protocol, state, srcToS and dstToS for the anomaly detection.

The direction field has unbalanced distribution, but it has no meaning without source or destination address. Also, the srcIP and srcPort fields are trivial and many unique values exist.

As a port number is always associated with an IP address and direction is used for the flow of source and destination, I decided to combine some fields as the followings.

	srcIP+srcPort	**dstIP+dstPort**	**srcIP+srcPort+direction+dstIP+dstPort**
Top	150.35.87.196 + 8	150.35.87.232 + 13363	150.35.87.196+ 1025+ <->+ 150.35.83.12+ 53+ 15
Top's Freq	480	37807	15
Top's Freq (%)	0.14	10.85	0.00 (0.004304)
Num of unique values	327,337	9,933	342,808

Figure 4: Top, Top's frequency, number of unique items for combination fields

The figure 4 shows the most frequent value, its frequency, and the number of unique values for combined fields based on the Splunk queries (see Appendix D). As can be seen from the figure 3 and 4, the frequency of the most appeared value in dstPort and dstIP+dstPort are the same. This means that most frequent destination port in the test dataset uses only one destination IP address. The combination of srcIP, srcPort, direction, dstIP and dstPort became too trivial.

Also, as can be seen from Appendix D, there are three most frequent values that appears same times (480) and the frequency of those values are much higher than the others. This means that it might have indication of anomaly and would help anomaly detection. In the dstIP+dstPort field, the most frequent value occupies around 10% of test datasets, which might help anomaly detection. So, we decided to use srcIP+srcPort and dstIP+dstPort.

Finally, we have the following fields for the anomaly detection.

Field	**Data Type**
duration	Numeric
totalPackets	Numeric
bytesBothDir	Numeric
bytesSrcToDst	Numeric
protocol	Categorical
state	Categorical
srcToS	Categorical
dstToS	Categorical
srcIP+srcPort	Categorical
dstIP+dstPort	Categorical

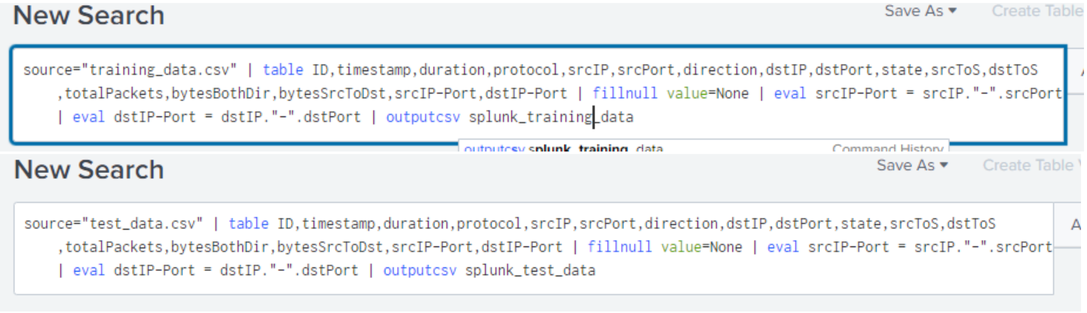
In the first project, basic features were already there without extracting features by ourselves and try to find patterns of attacks from the dataset manually with the guidance. In this second project, we tried to find features that are useful for machine learning for automated detection of attacks.

Attack	**Patterns**
SPAM	src_ip + dst_ip + dst_port (OR src_ip + dst_port)
Port scan	src_ip
HTTP	dst_ip

Figure 5: patterns of attack in the first assignment

3. Feature generation using Splunk

We used Splunk query and native feature to fill missing values and generate the CSV file that contains new features (srcIP+srcPort and dstIP+srcPort).



```
source="training_data.csv" | table ID,timestamp,duration,protocol,srcIP,srcPort,direction,dstIP,dstPort,state,srcToS,dstToS
    ,totalPackets,bytesBothDir,bytesSrcToDst,srcIP-Port,dstIP-Port | fillnull value=None | eval srcIP-Port = srcIP."-".srcPort
    | eval dstIP-Port = dstIP."-".dstPort | outputcsv splunk_training_data
```

```
source="test_data.csv" | table ID,timestamp,duration,protocol,srcIP,srcPort,direction,dstIP,dstPort,state,srcToS,dstToS
    ,totalPackets,bytesBothDir,bytesSrcToDst,srcIP-Port,dstIP-Port | fillnull value=None | eval srcIP-Port = srcIP."-".srcPort
    | eval dstIP-Port = dstIP."-".dstPort | outputcsv splunk_test_data
```

Figure 6: SPL query to fill missing values and create new features

The figure 6 shows the query to fill missing values and create new features and export the CSV file.

4. Two alternate methods of feature selection and generation using Python

We decided to use two types of unsupervised filter methods for feature selection: univariate filter method and multivariate filter method. Univariate filter methods are to rank individual features and select top N features, while multivariate filter methods are to remove the features that have mutual relationship with other features [1].

For univariate filter methods, we employed Variance Threshold, which remove all low-variance features. For multivariate filter methods, we used pandas corr() methods to determine the correlation between columns and remove highly correlated features to other features.

We will not normalise data as it might get rid of extreme values that could be anomalies.

We also used label encoding library for categorical value as one hot encoding gave memory error.

5. Two anomaly detection technique to build six models

We employed two different unsupervised anomaly detection techniques: isolation forest and local outlier factor. As we prepared three different feature sets from Splunk and two different feature selection techniques described earlier, we created six models in total to predict labels.

6. Isolation Forest experimental design, scoring and results

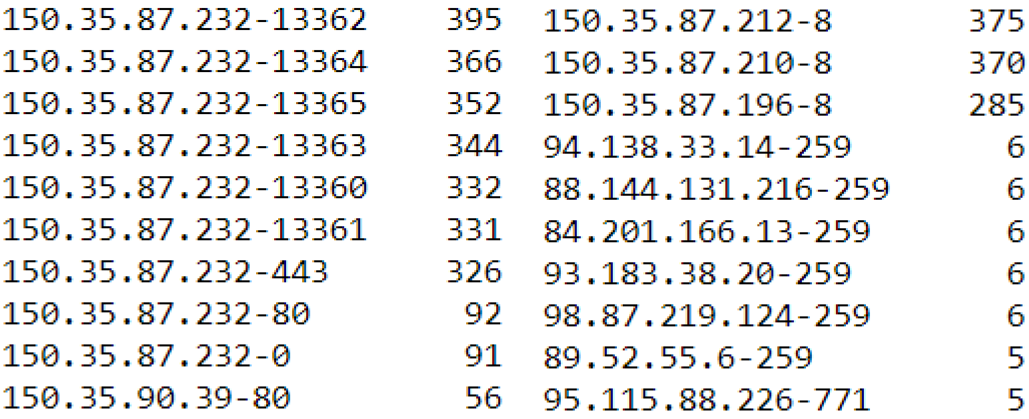
$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Figure 7: isolation forest score

Isolation forest isolates records by selecting a random split value between max and min values of a randomly selected feature [2].
We used isolation forest library from sklearn, which predicts labels either 1 or -1; 1 for normal and -1 for anomaly. The decision of predicted label is based on the depth of each tree created and the score calculated by the algorithm; if s is smaller than 0.5, it is likely to be normal, otherwise could be anomaly (Figure 7).
Isolation forest has some hyper parameter to tune the model. We decided to tune 'contamination' parameter, which is the proportion of outliers in dataset.

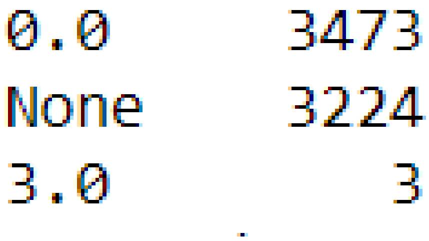
Contamination/feature set	Splunk selected feature set	Variation threshold selected feature set	Correlation based selected feature set
0.01	2909	7761	6700

Figure 8: number of anomalies with 0.1 contamination



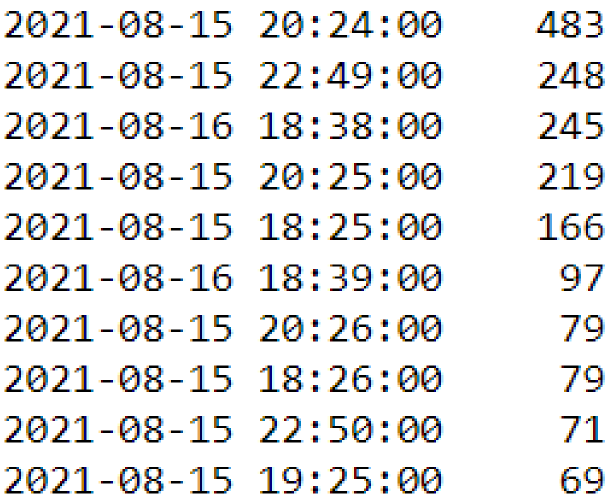
```
150.35.87.232-13362     395  150.35.87.212-8        375
150.35.87.232-13364     366  150.35.87.210-8        370
150.35.87.232-13365     352  150.35.87.196-8        285
150.35.87.232-13363     344  94.138.33.14-259         6
150.35.87.232-13360     332  88.144.131.216-259       6
150.35.87.232-13361     331  84.201.166.13-259        6
150.35.87.232-443       326  93.183.38.20-259         6
150.35.87.232-80         92  98.87.219.124-259        6
150.35.87.232-0          91  89.52.55.6-259           5
150.35.90.39-80          56  95.115.88.226-771        5
```

Figure 9: distribution of srcIP and dstIP of predicted anomalies of isolation forest with 0.01 contamination



```
0.0       3473
None      3224
3.0          3
```

Figure 10: distribution of attacked services of predicted anomalies of isolation forest with 0.01 contamination



```
2021-08-15 20:24:00    483
2021-08-15 22:49:00    248
2021-08-16 18:38:00    245
2021-08-15 20:25:00    219
2021-08-15 18:25:00    166
2021-08-16 18:39:00     97
2021-08-15 20:26:00     79
2021-08-15 18:26:00     79
2021-08-15 22:50:00     71
2021-08-15 19:25:00     69
```

Figure 11: number of detected attacks per 1 min binned timestamp

We used Grid Search to get optimised contamination parameter with the validation dataset and got 0.01 is the optimised parameter.
The figure 8 shows the number of anomalies on each feature set with contamination 0.01.

7. Local Outlier Factor experimental design, scoring and results

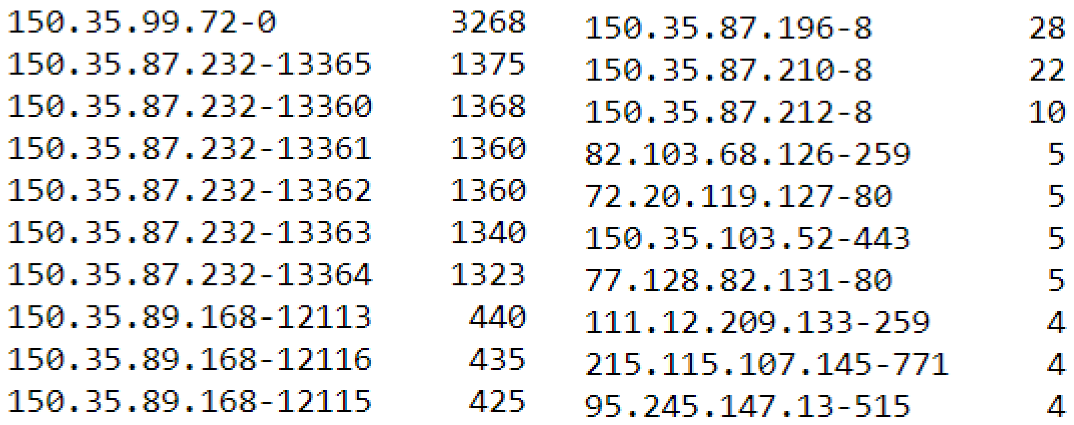
$$LOF_k(p) = \frac{1}{k}\sum\nolimits_{o \in N_{(p,k)}} \frac{lrd_k(o)}{lrd_k(p)}$$

Figure 12: Local Outlier Factor Score

Local Outlier Factor measures the local deviation of density of the dataset with respect to its neighbour and identify the samples that have lower density than their neighbours [3].
We used local outlier factor library from sklean, which predicts labels either 1 or -1; 1 for normal and -1 for anomaly. The decision of predicted label is based on the degree of abnormality of the record and the score calculated by the algorithm; comparing the score if the score is abnormal, it could be anomaly (Figure 12).
Local outlier factor has some hyper parameter to turn the model. We decided to tune ‘n_neightbors’ parameter, which is number of neighbours to measure the distance from the point to n th neighbour.

N_neightbors/feature set	Splunk selected feature set	Variation threshold selected feature set	Correlation based selected feature set
10	33884	20244	21213

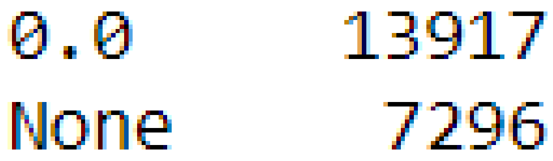
Figure 13: number of anomalies on each feature set with 10 n-neightbor



```
150.35.99.72-0           3268
150.35.87.232-13365      1375
150.35.87.232-13360      1368
150.35.87.232-13361      1360
150.35.87.232-13362      1360
150.35.87.232-13363      1340
150.35.87.232-13364      1323
150.35.89.168-12113       440
150.35.89.168-12116       435
150.35.89.168-12115       425
```

```
150.35.87.196-8          28
150.35.87.210-8          22
150.35.87.212-8          10
82.103.68.126-259         5
72.20.119.127-80          5
150.35.103.52-443         5
77.128.82.131-80          5
111.12.209.133-259        4
215.115.107.145-771       4
95.245.147.13-515         4
```

Figure 14: distribution of srcIP and dstIP of predicted anomalies of LOF with 10 n_neightbours



```
0.0       13917
None       7296
```

Figure 15: distribution of attacked services of predicted anomalies of LOF with 10 n_neightbours

We calculated the f1 score of prediction on validation data and picked best n_neightbor parameter, which was 10.
The figure 13 shows the number of anomalies on each feature set with 10 n-neightbor.

8. Attack scenario using timestamp and conclusion

We believe that isolation forest with correlation-based feature selection method is the best model bacause LOF with correlation-based feature selection detects much more anomalies and far less srcIP+Port, dstIP+Port, and destination type of service that appear on top 10 in test dataset than isolation forest. We considered that there are 112 victims and 3904 attackers with 7254 records. The first attack detected was at 2021-08-15 18:24:13.357670 and the last attack was 2021-08-16 19:49:55.940349. There were largest number of attacks detected at around 2021-08-15 20:24:00. Many of identified attacks used ICMP and TCP protocols and attacked services are mostly 0.0. My guess of the attack is ICMP based DDoS attack.
We finally created the final CSV file as the proof of our work, and it contains the stream IDs that are identified as attacks.

Task 2

1. **Introduction and supervised learning model**

We have the provided three datasets: training dataset, testing dataset and validation dataset. They all come with ture labels.

We decided to use Logistic Regression for supervised learning model and for generating adversarial samples. Logistic regression uses logistic function to model probability of event, in this case anomaly or not.

I picked 6 features that will not affect the botnet functionality: ‘duration’,'totalPackets','bytesBothDir','bytesSrcToDst','srcPort','state'.

We used label encoder to categorical values. We also decided to use KBest feature selection method with f_classif, which calculate ANOVA F-value, to select top 3 features, which are 'duration', 'srcPort', 'state'.

The prediction scores on the test dataset are below.

Accuracy score	0.8435241674511298
F1 score (average = ‘macro’)	0.608170271206472
Precision score (average = ‘macro’)	0.7256017044745846
Recall score (average = ‘macro’)	0.5883061831804419

Figure 16: scores of predictions on the test dataset

2. **Adversarial sample generation**

In order to create adversarial sample, we chose one IP address (150.35.87.196) that was detected as anomaly and appeared the most in the predicted result.

Label	**Count**
0	13711
1	4250

Figure 17: Count predicted labels of 150.35.87.196

Label	**Count**
1	17961

Figure 18: Count true labels of 150.35.87.196

$$x' \leftarrow x + \epsilon \cdot sign\left(\frac{\partial loss_{true}}{\partial x}\right)$$

Figure 19: Fast Gradient Sign Method formula

We used Fast Gradient Descent Method (FGDM) to create adversarial samples that perturb the inputs to be miss-classified by logistic regression. The formula above shows the way to perturb the inputs: X’ is adversarial sample, the epsilon is very small number, and the rest is the gradient of cost function with respect to X.

3. Attack logistic regression

Using the fast gradient sign method formula (the figure 19), we have created new input for logistic regression. We set epsilon value to 0.5 to see the input changed.

Label	Count
0	4233
1	13728

Figure 19: Count predicted labels of 150.35.87.196, with adversarial samples

As can be seen from the figure 17 and 19, the logistic regression model misclassified the adversarial samples that was generated by fast gradient sign method.

4. How to change the raw traffic of the chosen IP address

So far, our logistic regression was trained with the three features: 'duration', 'srcPort', 'state'. By manipulating those features by FGDM, some of anomalies were identified as normal.

To make the modified record consistent pm the raw traffic records, we should just include very small portion/subset, such as 10 out of 1000. Also, we should make it consistent with data types as well: srcPort is integer, duration is float, state is string.

Reference:

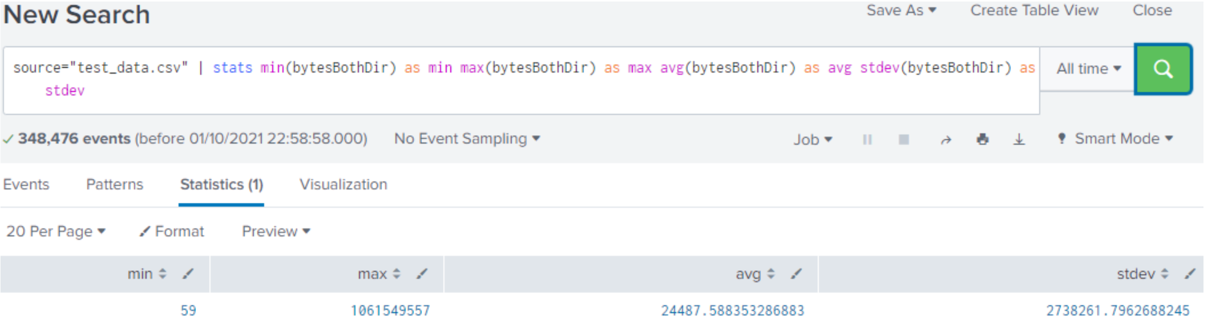
[1] https://stackabuse.com/applying-filter-methods-in-python-for-feature-selection/

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html#sklearn.neighbors.LocalOutlierFactor

Appendix:

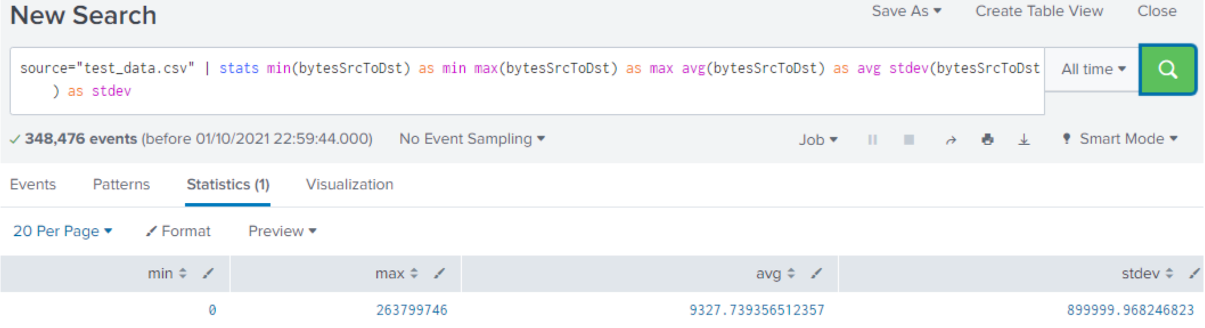
Appendix A: min, max, average, standard deviation for each numeric field



```
source="test_data.csv" | stats min(bytesBothDir) as min max(bytesBothDir) as max avg(bytesBothDir) as avg stdev(bytesBothDir) as
    stdev
```

✓ 348,476 events (before 01/10/2021 22:58:58.000)

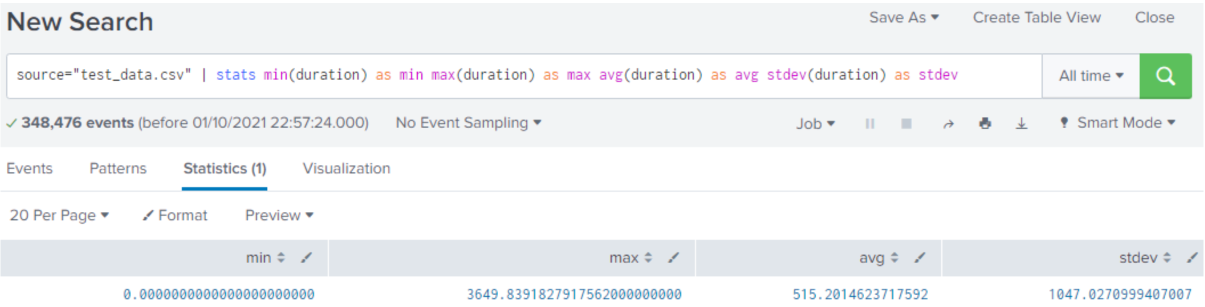
min	max	avg	stdev
59	1061549557	24487.588353286883	2738261.7962688245



```
source="test_data.csv" | stats min(bytesSrcToDst) as min max(bytesSrcToDst) as max avg(bytesSrcToDst) as avg stdev(bytesSrcToDst
    ) as stdev
```

✓ 348,476 events (before 01/10/2021 22:59:44.000)

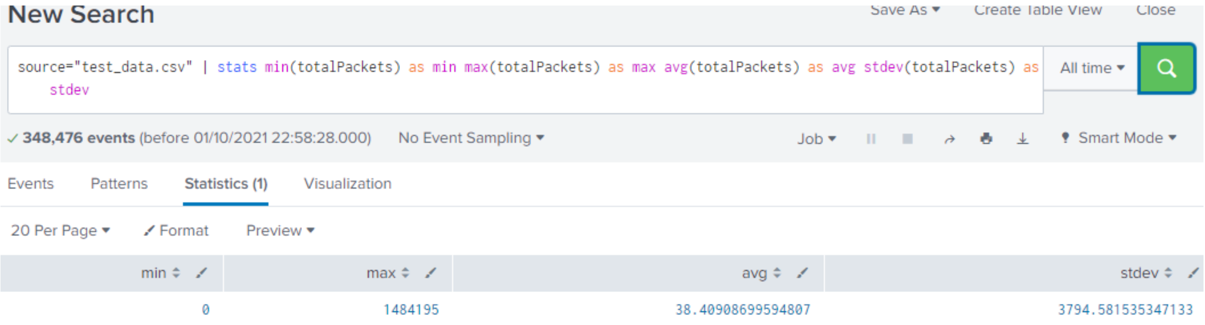
min	max	avg	stdev
0	263799746	9327.739356512357	899999.968246823



```
source="test_data.csv" | stats min(duration) as min max(duration) as max avg(duration) as avg stdev(duration) as stdev
```

✓ 348,476 events (before 01/10/2021 22:57:24.000)

min	max	avg	stdev
0.00000000000000000000	3649.83918279175620000000	515.2014623717592	1047.0270999407007

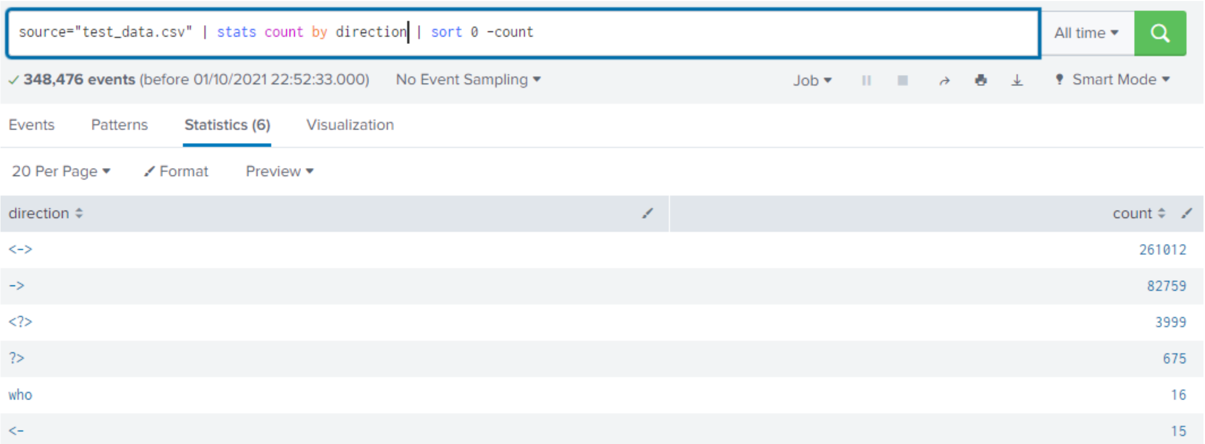


```
source="test_data.csv" | stats min(totalPackets) as min max(totalPackets) as max avg(totalPackets) as avg stdev(totalPackets) as
    stdev
```

✓ 348,476 events (before 01/10/2021 22:58:28.000)

min	max	avg	stdev
0	1484195	38.40908699594807	3794.581535347133

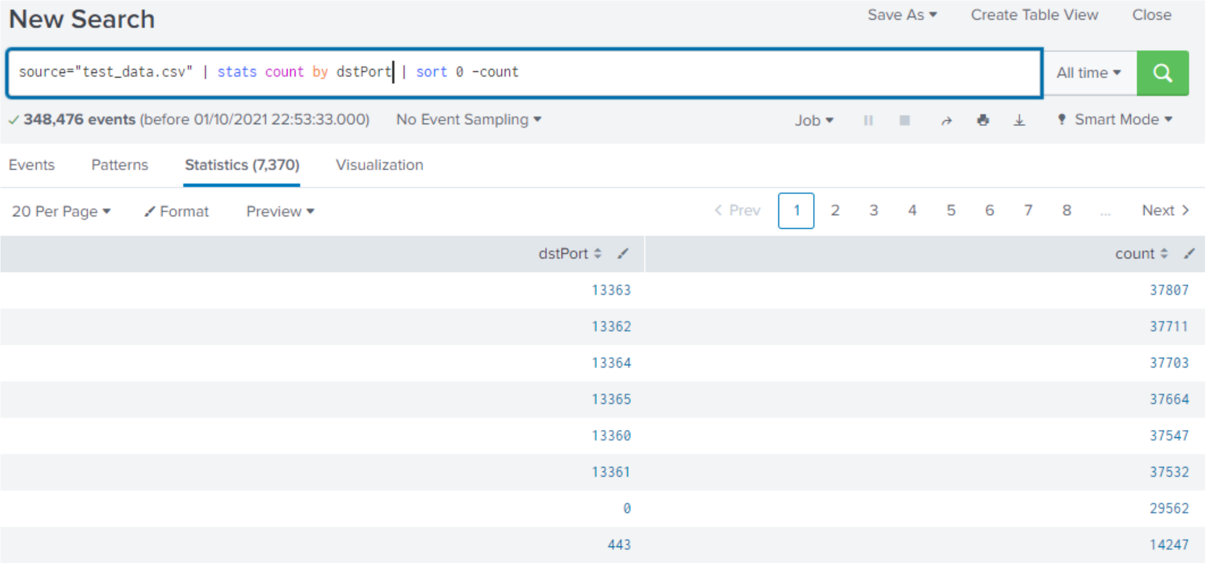
Appendix B: top, top frequency, number of unique items for each field



```
source="test_data.csv" | stats count by direction | sort 0 -count
```

✓ **348,476 events** (before 01/10/2021 22:52:33.000) No Event Sampling

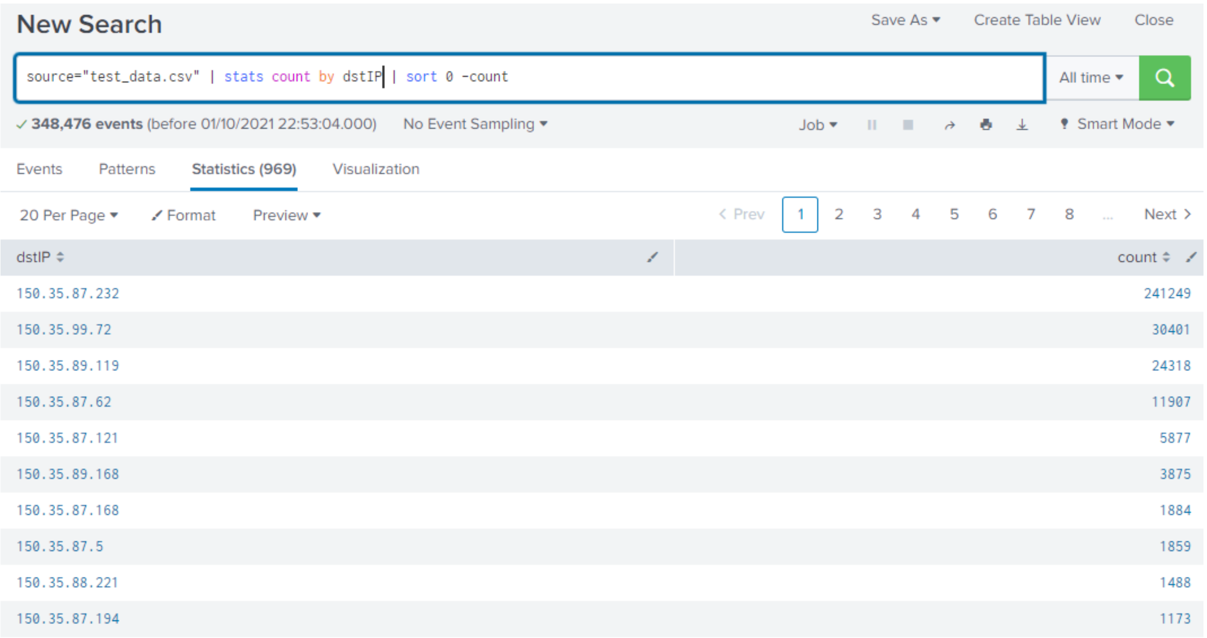
direction	count
<->	261012
->	82759
<?>	3999
?>	675
who	16
<-	15



```
source="test_data.csv" | stats count by dstPort | sort 0 -count
```

✓ **348,476 events** (before 01/10/2021 22:53:33.000) No Event Sampling

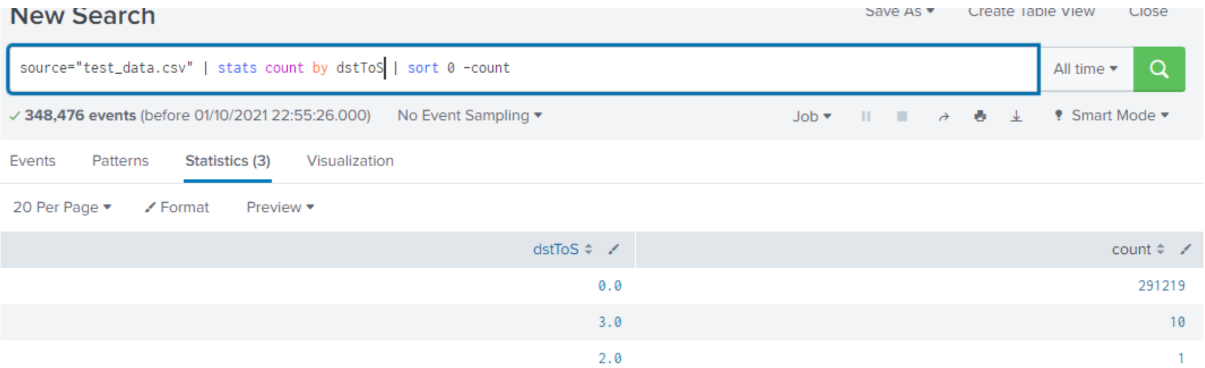
dstPort	count
13363	37807
13362	37711
13364	37703
13365	37664
13360	37547
13361	37532
0	29562
443	14247



```
source="test_data.csv" | stats count by dstIP | sort 0 -count
```

✓ **348,476 events** (before 01/10/2021 22:53:04.000) No Event Sampling

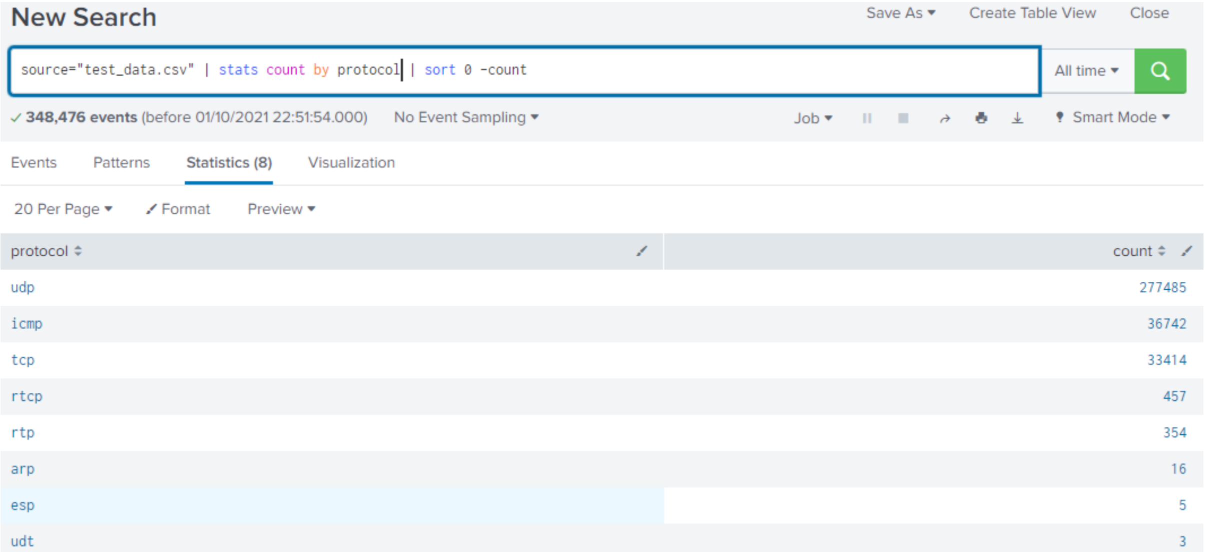
dstIP	count
150.35.87.232	241249
150.35.99.72	30401
150.35.89.119	24318
150.35.87.62	11907
150.35.87.121	5877
150.35.89.168	3875
150.35.87.168	1884
150.35.87.5	1859
150.35.88.221	1488
150.35.87.194	1173



```
source="test_data.csv" | stats count by dstToS | sort 0 -count
```

✓ 348,476 events (before 01/10/2021 22:55:26.000)

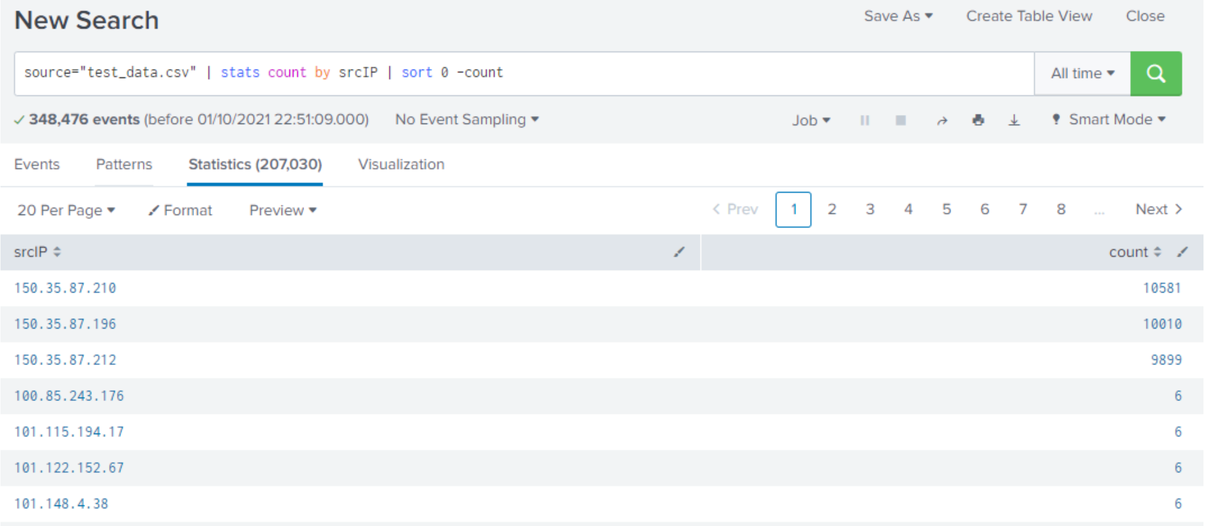
dstToS	count
0.0	291219
3.0	10
2.0	1



```
source="test_data.csv" | stats count by protocol | sort 0 -count
```

✓ 348,476 events (before 01/10/2021 22:51:54.000)

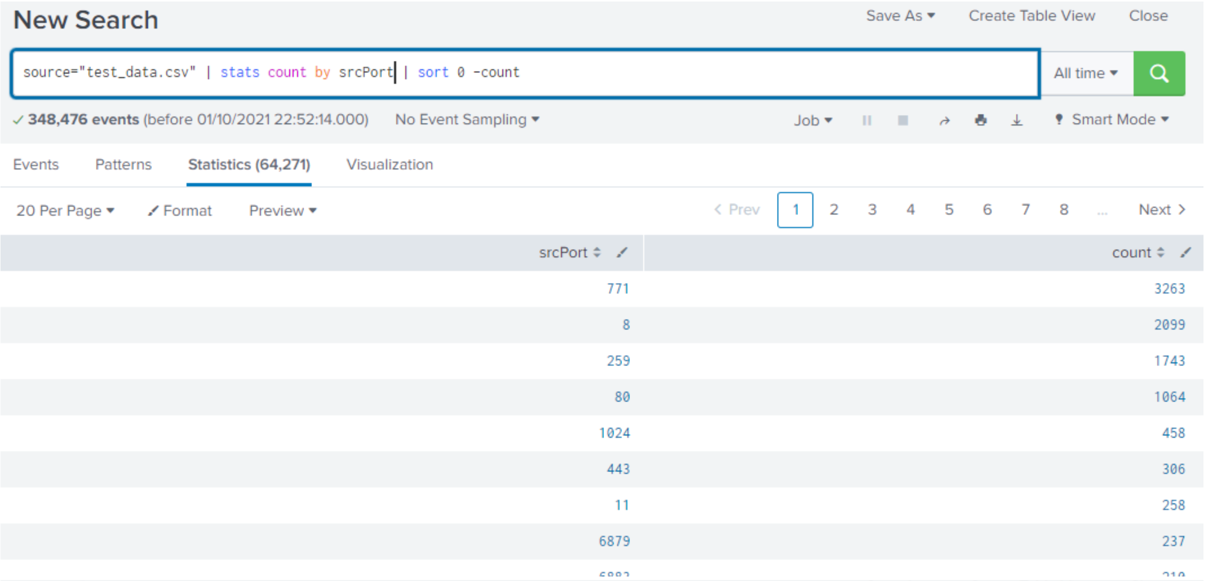
protocol	count
udp	277485
icmp	36742
tcp	33414
rtcp	457
rtp	354
arp	16
esp	5
udt	3



```
source="test_data.csv" | stats count by srcIP | sort 0 -count
```

✓ 348,476 events (before 01/10/2021 22:51:09.000)

srcIP	count
150.35.87.210	10581
150.35.87.196	10010
150.35.87.212	9899
100.85.243.176	6
101.115.194.17	6
101.122.152.67	6
101.148.4.38	6

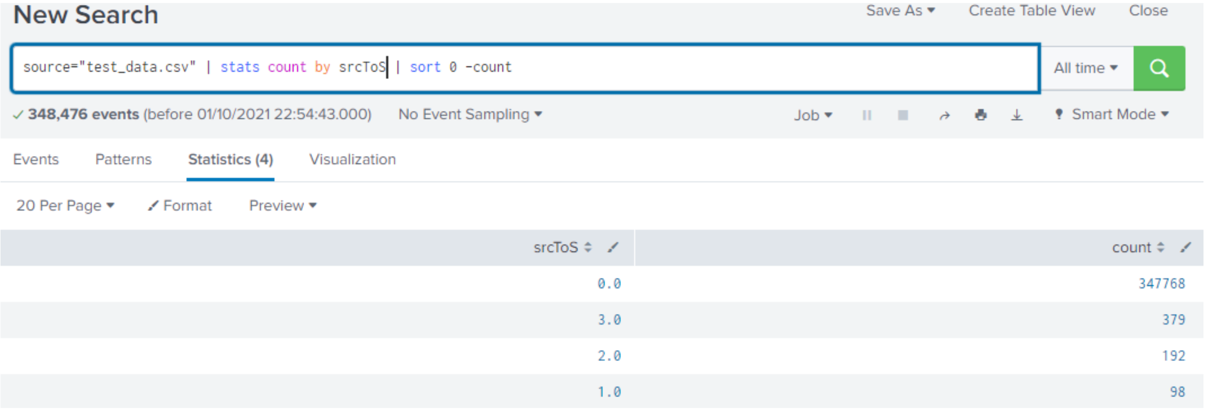


```
source="test_data.csv" | stats count by srcPort | sort 0 -count
```

✓ 348,476 events (before 01/10/2021 22:52:14.000)

Statistics (64,271)

srcPort	count
771	3263
8	2099
259	1743
80	1064
1024	458
443	306
11	258
6879	237

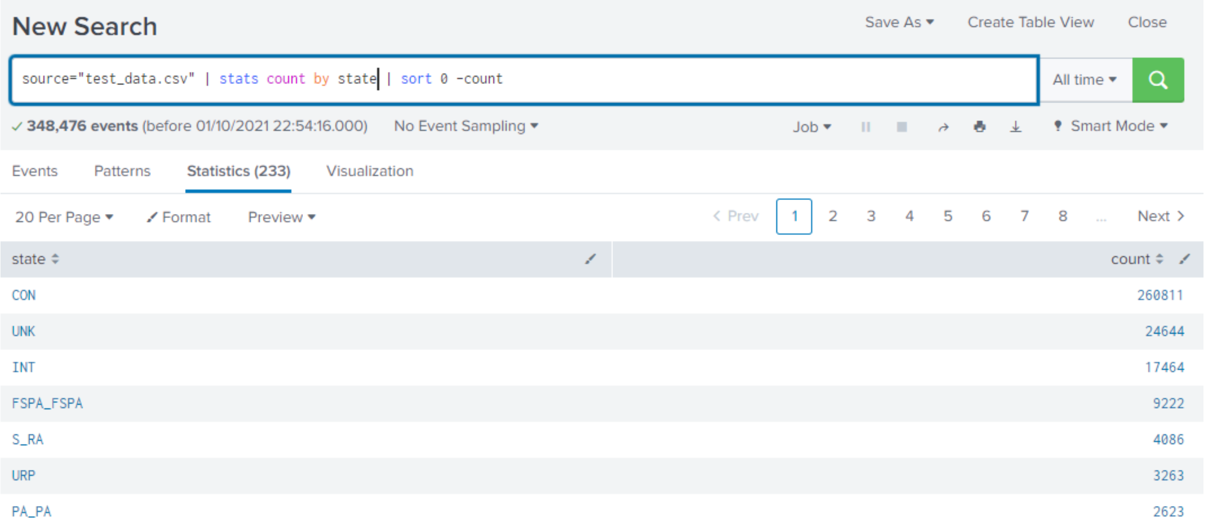


```
source="test_data.csv" | stats count by srcToS | sort 0 -count
```

✓ 348,476 events (before 01/10/2021 22:54:43.000)

Statistics (4)

srcToS	count
0.0	347768
3.0	379
2.0	192
1.0	98



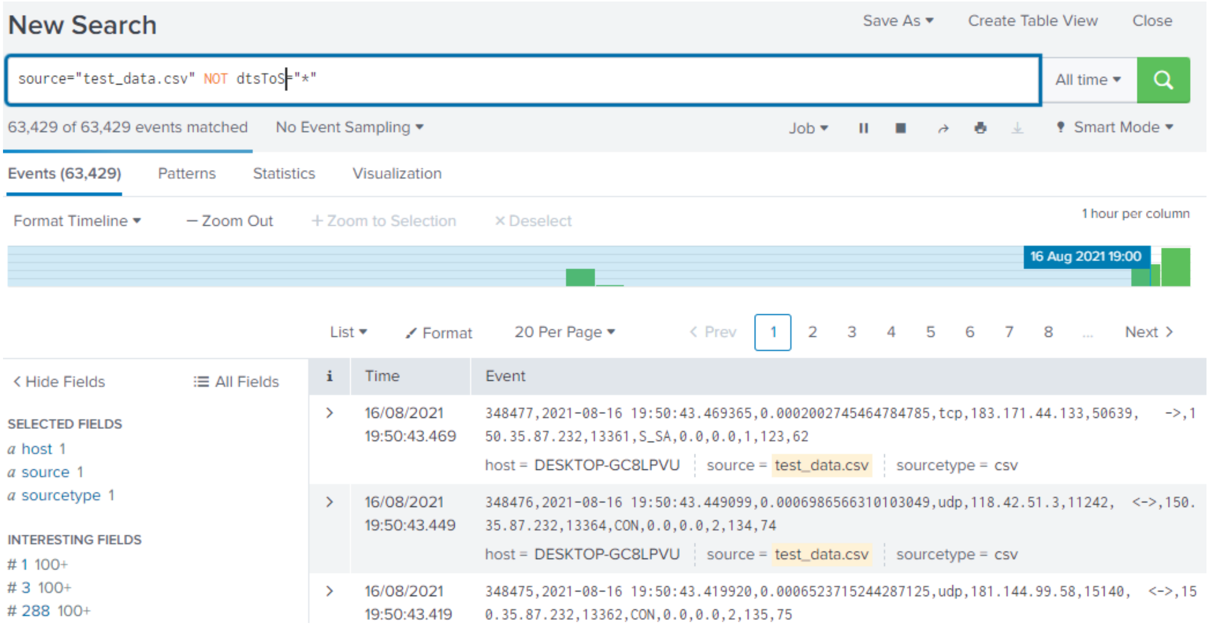
```
source="test_data.csv" | stats count by state | sort 0 -count
```

✓ 348,476 events (before 01/10/2021 22:54:16.000)

Statistics (233)

state	count
CON	260811
UNK	24644
INT	17464
FSPA_FSPA	9222
S_RA	4086
URP	3263
PA_PA	2623

Appendix C: missing values



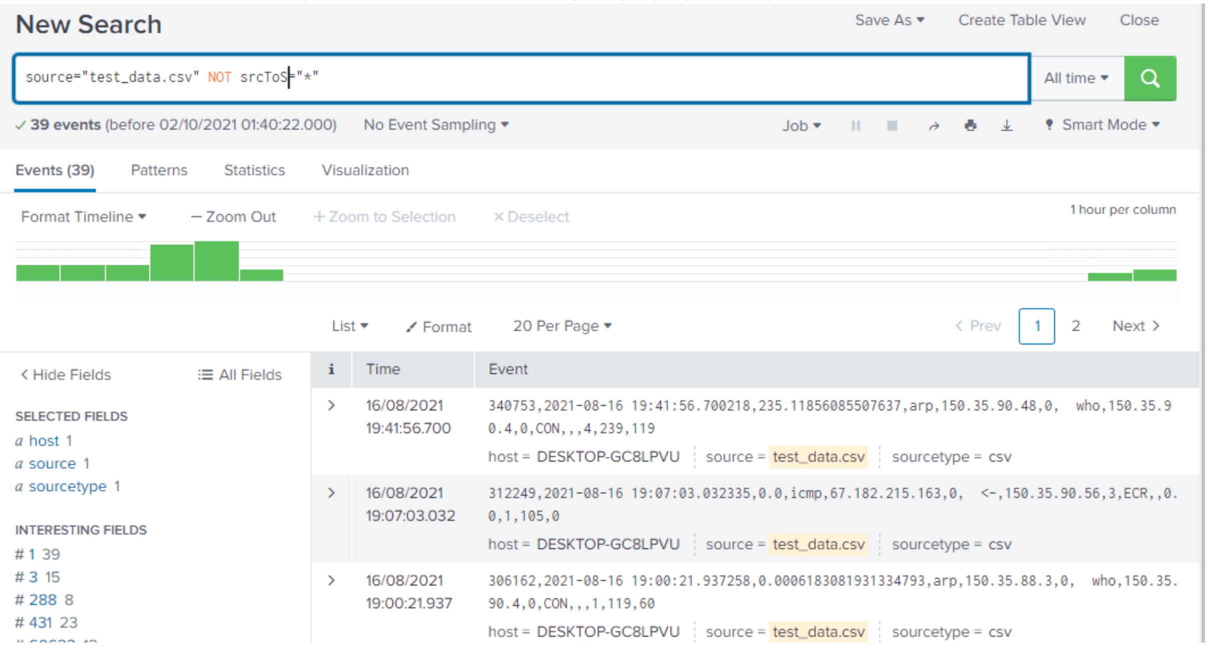
```
source="test_data.csv" NOT dtsToS="*"
```

63,429 of 63,429 events matched No Event Sampling ▾

SELECTED FIELDS
a host 1
a source 1
a sourcetype 1

INTERESTING FIELDS
1 100+
3 100+
288 100+

i	Time	Event
>	16/08/2021 19:50:43.469	348477,2021-08-16 19:50:43.469365,0.0002002745464784785,tcp,183.171.44.133,50639, ->,150.35.87.232,13361,S_SA,0.0,0.0,1,123,62 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv
>	16/08/2021 19:50:43.449	348476,2021-08-16 19:50:43.449099,0.0006986566310103049,udp,118.42.51.3,11242, <->,150.35.87.232,13364,CON,0.0,0.0,2,134,74 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv
>	16/08/2021 19:50:43.419	348475,2021-08-16 19:50:43.419920,0.0006523715244287125,udp,181.144.99.58,15140, <->,150.35.87.232,13362,CON,0.0,0.0,2,135,75



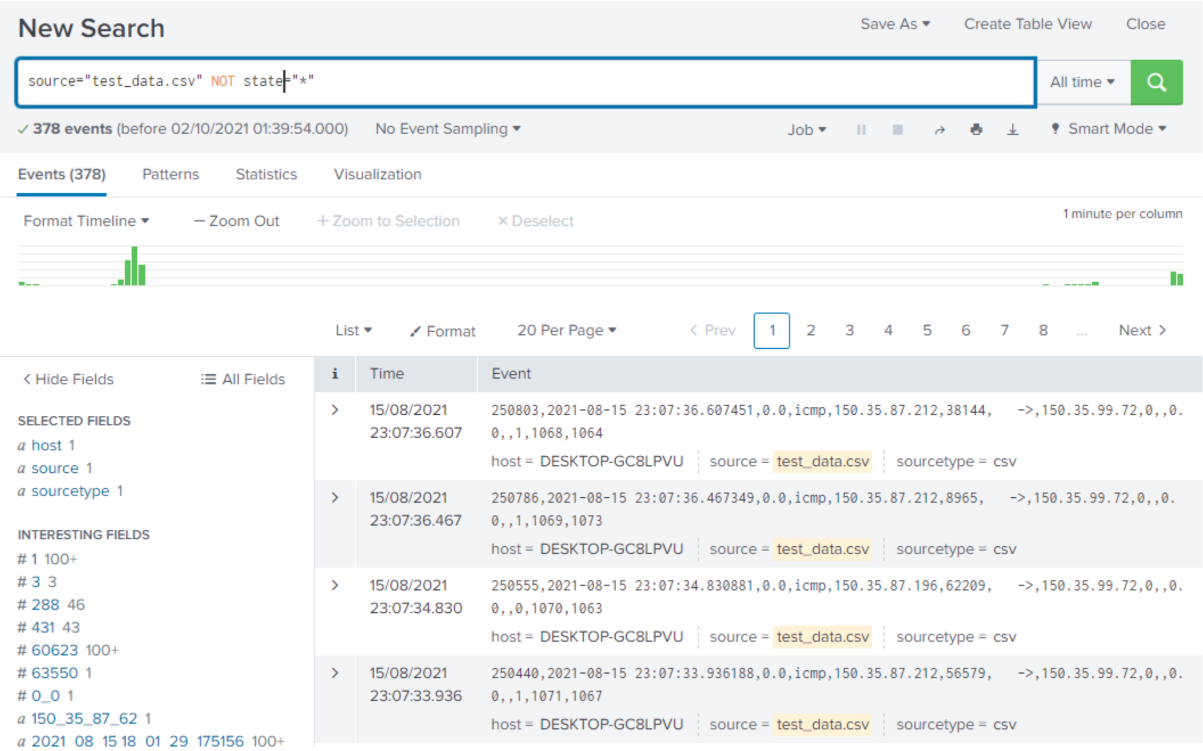
```
source="test_data.csv" NOT srcToS="*"
```

✓ 39 events (before 02/10/2021 01:40:22.000) No Event Sampling ▾

SELECTED FIELDS
a host 1
a source 1
a sourcetype 1

INTERESTING FIELDS
1 39
3 15
288 8
431 23

i	Time	Event
>	16/08/2021 19:41:56.700	340753,2021-08-16 19:41:56.700218,235.11856085507637,arp,150.35.90.48,0, who,150.35.90.4,0,CON,,,4,239,119 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv
>	16/08/2021 19:07:03.032	312249,2021-08-16 19:07:03.032335,0.0,icmp,67.182.215.163,0, <-,150.35.90.56,3,ECR,,0.0,1,105,0 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv
>	16/08/2021 19:00:21.937	306162,2021-08-16 19:00:21.937258,0.0006183081931334793,arp,150.35.88.3,0, who,150.35.90.4,0,CON,,,1,119,60 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv



```
source="test_data.csv" NOT state="*"
```

✓ 378 events (before 02/10/2021 01:39:54.000)

SELECTED FIELDS

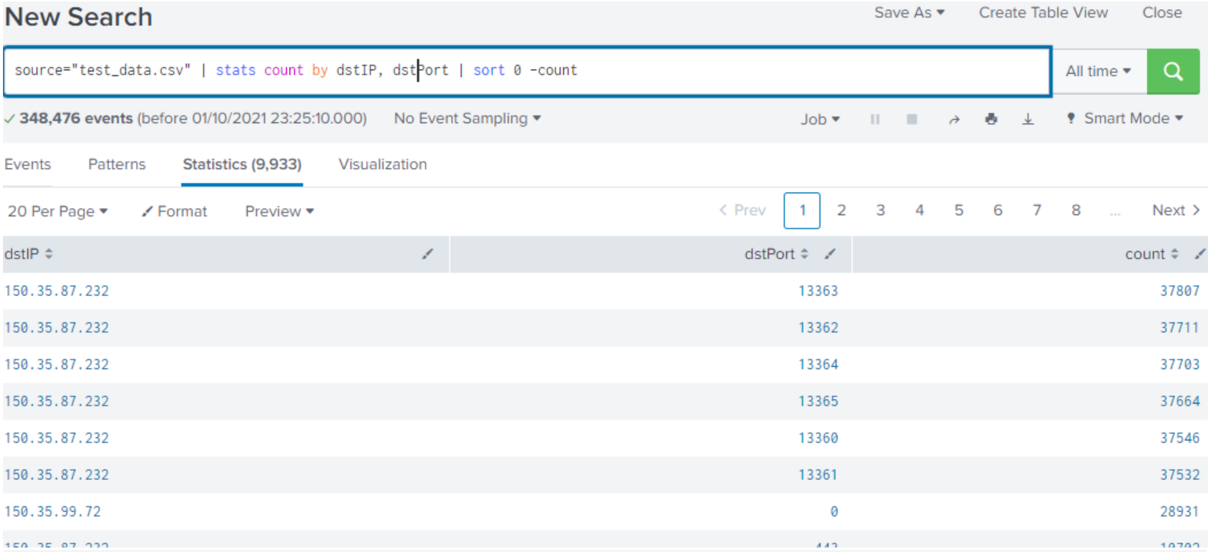
- a host 1
- a source 1
- a sourcetype 1

INTERESTING FIELDS

- # 1 100+
- # 3 3
- # 288 46
- # 431 43
- # 60623 100+
- # 63550 1
- # 0_0 1
- a 150_35_87_62 1
- a 2021 08 15 18 01 29 175156 100+

i	Time	Event
›	15/08/2021 23:07:36.607	250803,2021-08-15 23:07:36.607451,0.0,icmp,150.35.87.212,38144, ->,150.35.99.72,0,,0.0,,1,1068,1064 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv
›	15/08/2021 23:07:36.467	250786,2021-08-15 23:07:36.467349,0.0,icmp,150.35.87.212,8965, ->,150.35.99.72,0,,0.0,,1,1069,1073 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv
›	15/08/2021 23:07:34.830	250555,2021-08-15 23:07:34.830881,0.0,icmp,150.35.87.196,62209, ->,150.35.99.72,0,,0.0,,0,1070,1063 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv
›	15/08/2021 23:07:33.936	250440,2021-08-15 23:07:33.936188,0.0,icmp,150.35.87.212,56579, ->,150.35.99.72,0,,0.0,,1,1071,1067 host = DESKTOP-GC8LPVU source = test_data.csv sourcetype = csv

Appendix D: top, top frequency, number of unique items for combination fields



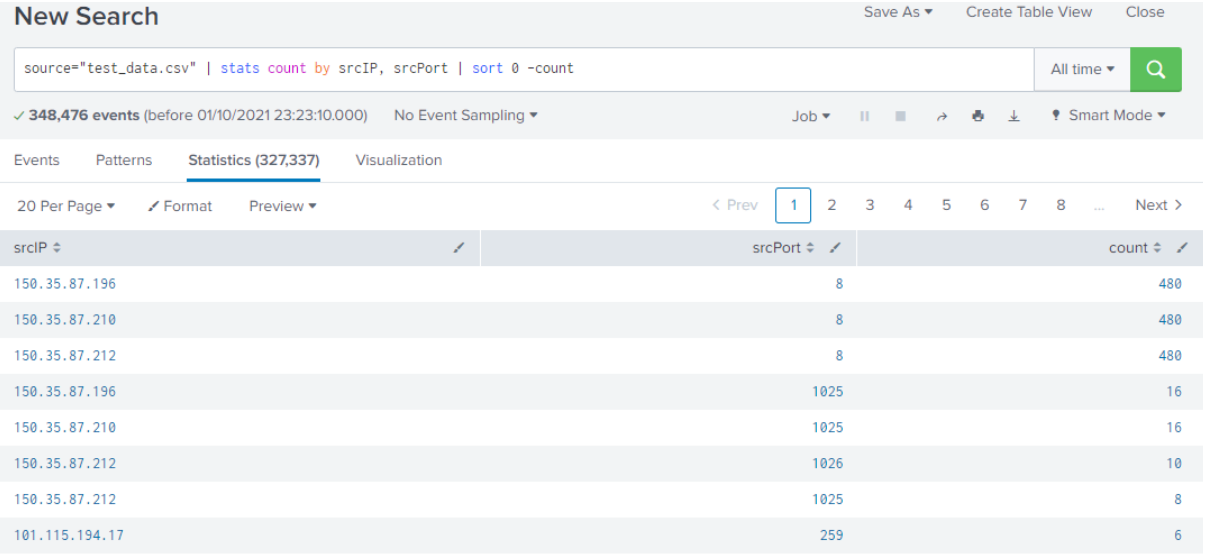
New Search

```
source="test_data.csv" | stats count by dstIP, dstPort | sort 0 -count
```

✓ **348,476 events** (before 01/10/2021 23:25:10.000) No Event Sampling ▾

Statistics (9,933)

dstIP	dstPort	count
150.35.87.232	13363	37807
150.35.87.232	13362	37711
150.35.87.232	13364	37703
150.35.87.232	13365	37664
150.35.87.232	13360	37546
150.35.87.232	13361	37532
150.35.99.72	0	28931
150.35.87.232	443	10702



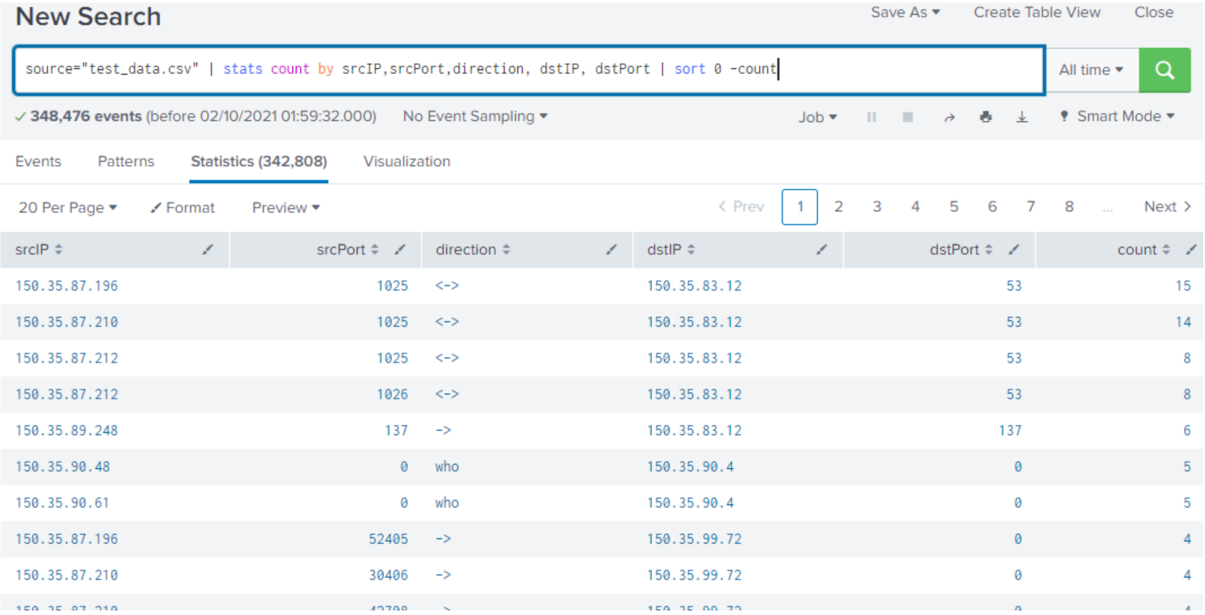
New Search

```
source="test_data.csv" | stats count by srcIP, srcPort | sort 0 -count
```

✓ **348,476 events** (before 01/10/2021 23:23:10.000) No Event Sampling ▾

Statistics (327,337)

srcIP	srcPort	count
150.35.87.196	8	480
150.35.87.210	8	480
150.35.87.212	8	480
150.35.87.196	1025	16
150.35.87.210	1025	16
150.35.87.212	1026	10
150.35.87.212	1025	8
101.115.194.17	259	6



New Search

```
source="test_data.csv" | stats count by srcIP,srcPort,direction, dstIP, dstPort | sort 0 -count
```

✓ **348,476 events** (before 02/10/2021 01:59:32.000) No Event Sampling ▾

Statistics (342,808)

srcIP	srcPort	direction	dstIP	dstPort	count
150.35.87.196	1025	<->	150.35.83.12	53	15
150.35.87.210	1025	<->	150.35.83.12	53	14
150.35.87.212	1025	<->	150.35.83.12	53	8
150.35.87.212	1026	<->	150.35.83.12	53	8
150.35.89.248	137	->	150.35.83.12	137	6
150.35.90.48	0	who	150.35.90.4	0	5
150.35.90.61	0	who	150.35.90.4	0	5
150.35.87.196	52405	->	150.35.99.72	0	4
150.35.87.210	30406	->	150.35.99.72	0	4
150.35.87.210	42708	->	150.35.99.72	0	4