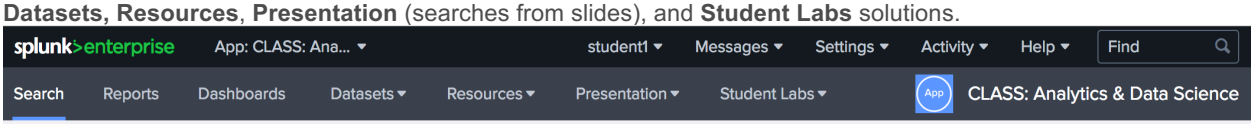
Splunk for Analytics and Data Science Lab Exercises

NOTE: If you finish a lab exercise early, experiment with searches, dashboards, or alerts you would like to consider building when you return to work. We recommend you document them.

This class environment includes an app called **CLASS: Analytics & Data Science,** which includes dropdowns for **Datasets, Resources**, **Presentation** (searches from slides), and **Student Labs** solutions.



Datasets for this Course		
Airline Tweets	Emotion Tweets	Online Retail
Auto MPG	Firewall Traffic	Phishing
BCG Customer Churn	Host Response Time	Port Scans
Call Center	Housing	Power Plant
Census	Iris	Server Power
CPU Utilization	Liver Patients	Spam Texts
Debt & Default	Malware Domains	
Diabetes	Occupancy Sensors	

Resources for this course:

- ML Toolkit Quck Ref
- Algorithm Docs
- Ladder of Powers
- Numeric Outliers
- Generic Text Analytics

Lab Exercise 1 – Review Available Data

Description

Log in to Splunk, customize your Splunk settings, and review the available data.

Steps

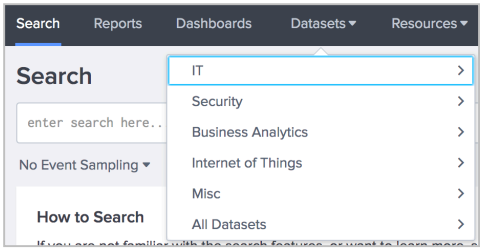
Task 1: Log into Splunk on the classroom server.

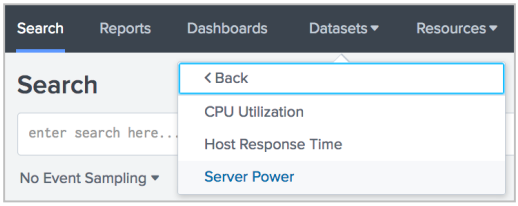
1. Direct your web browser to the class lab system, for example:{Server-name}.splunk.com
2. Log in with the credentials assigned by your instructor.
3. If a tour appears, you can view it or skip it.
4. In the panel on the left, or from the App dropdown, click **CLASS: Analytics & Data Science**.

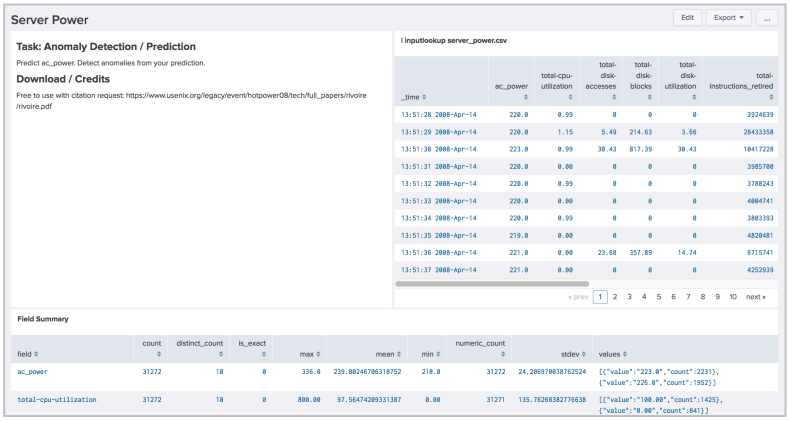
Task 2: Review available data.

5. Click **Data Summary**.
6. Click the **Sourcetypes** tab to examine the sourcetypes available in this course.
7. Click the **Sources** tab and the **Hosts** tabs to see the related Sources and Hosts.
8. In addition to the data in the main index, you'll be working with a variety of data sources that can be previewed from the **Datasets** dropdown in the navigation bar. In each preview, you can see some of the data, a description of the attributes, and a field summary. Take a look at each one.

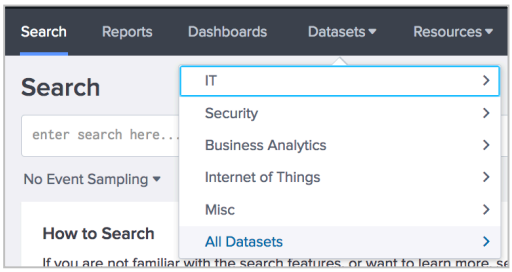
Results Example:







9. To view an alphabetical list of all the datasets in one view, click the **All Datasets** from the dropdown.



Task 3: Change your account settings to reflect your name and local time zone.

10. Click **your login name** on the Splunk bar to edit your account settings.
11. Click **Account Settings**.
12. In the **Full name** field, type your name.
13. From the **Time zone** menu, select your local time zone.
14. Set your **Default app** to **ads (Analytics & Data Science).**
15. Click **Save** and return to the **CLASS: Analytics & Data Science** app.

Lab Exercise 2 – Exploratory Data Analysis

Description

Begin the process of analyzing your data by exploring the `cluster, associate, fieldsummary`, `bin` and `makecontinuous` commands.

Steps

Task 1: Compare using the Patterns tab and the `cluster` command to create groups of events.

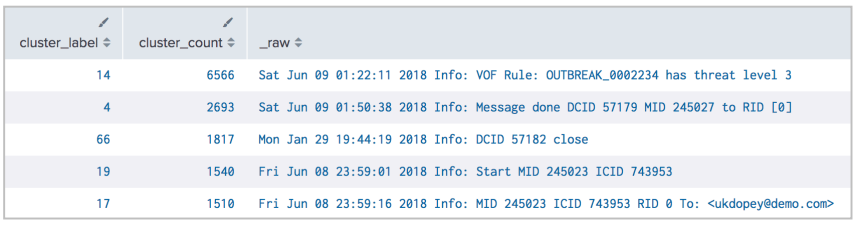
1. Search the `cisco_esa` sourcetype over **All Time**.
2. Switch to the **Patterns** tab to examine the clusters that Splunk recognized.
3. Click the largest cluster to view the text / keywords that Splunk clustered on.
4. Move the slider to the right and then the left to examine how the results change.

Results Example:

5. Switch back to the **Events** tab.
6. Add the `cluster` command to your search with a threshold of `0.5` and `showcount=t`
7. Use the `table` command to include the `cluster_label`, `cluster_count`, and `_raw` fields.
8. Sort descending by `cluster_count`.
9. Save your results as a report and name it **L2S1**

```
sourcetype=cisco_esa
| cluster t=0.5 showcount=t labelfield=cluster_label
| table cluster_label cluster_count _raw
| sort - cluster_count
```

Results example:



cluster_label	cluster_count	_raw
14	6566	Sat Jun 09 01:22:11 2018 Info: VOF Rule: OUTBREAK_0002234 has threat level 3
4	2693	Sat Jun 09 01:50:38 2018 Info: Message done DCID 57179 MID 245027 to RID [0]
66	1817	Mon Jan 29 19:44:19 2018 Info: DCID 57182 close
19	1540	Fri Jun 08 23:59:01 2018 Info: Start MID 245023 ICID 743953
17	1510	Fri Jun 08 23:59:16 2018 Info: MID 245023 ICID 743953 RID 0 To: <ukdopey@demo.com>

10. Which style of interface and results do you prefer? The more automated and color-coded Patterns tab with percentages and keywords specified, or using SPL to specify cluster parameters and view results in the Statistics tab?

Task 2: Explore clustering settings

11. Try modifying the t value to a low value such as 0.1, then a high value such as 0.9. (This is like moving the slider in the Patterns tab).
12. Add `labelonly=t` to the `cluster` command.
13. You can also see keywords from the cluster command. Add the `findkeywords` command to the search by replacing the `table` command with `findkeywords labelfield=cluster_label`
14. Examine the `keywords` that were included and excluded, as well as the values of the included and excluded `keywords` fields, `confidence` field, `percentInputGroup`, and `percentMatched` fields.
15. Save your results as a report and name it **L2S2**

```
sourcetype=cisco_esa
| cluster t=0.5 labelonly=t
| findkeywords labelfield=cluster_label
```

Results example:

confidence	eventTypeable	excludeKeywords	groupID	includeKeywords	numInInputGroup	numMatched	percentInInputGroup	percentMatched	sampleEvent	search	timeendpos	timestartpos	totalEvents	totalInputGroups
1	1		14	threat	6566	6566	0.18381344307270234	0.18381344307270234	Sat Jun 09 01:22:11 2018 Info: VOF Rule: OUTBREAK_0002234 has threat level 3	search sourcetype=cisco_esa threat	24	4	35721	115
1	1		5	to DCID	2676	2676	0.07491391618375745	0.07491391618375745	Sat Jun 09 01:50:38 2018 Info: Message done DCID 57179 MID 245027 to RID [0]	search sourcetype=cisco_esa to DCID	24	4	35721	115
1	1		21	finished	1496	1505	0.04188012653621119	0.042132079169116 2	Fri Jun 08 23:48:11 2018 Info: Message finished MID 245022 done	search sourcetype=cisco_esa finished	24	4	35721	115
1	1		19	Start	1434	1442	0.040144452842865 54	0.040368410738781106	Fri Jun 08 23:59:01 2018 Info: Start MID 245023 ICID 743953	search sourcetype=cisco_esa Start	24	4	35721	115

Task 3: Use the `associate` command to find relationships among fields' values in your data.

Load the phishing dataset to find associations between various fields and the `Result` field over **All time**. Please review the phishing dataset information page for context.

16. Load the phishing dataset with `inputlookup phishing.csv`
17. Create an association table.
18. Filter to see associations that have the `Target_Key` value of `Result`
19. Determine which fields have the strongest association with the value of `Result`
    - What does a `Conditional_Entropy` of `0` mean?
    - Which fields' values seem most related? (These could be used to build effective models.)
20. Save your results as a report and name it **L2S3**

```
| inputlookup phishing.csv
| associate
| search Target_Key=Result
```

Results example:

Reference_Key	Reference_Value	Target_Key	Support	Unconditional_Entropy	Conditional_Entropy	Entropy_Improvement	Top_Conditional_Value	Description
Prefix_Suffix	1	Result	13.25%	0.991	0.000	0.990624	1 (55.69% -> 100.00%)	When 'Prefix_Suffix' has the value '1', the entropy of 'Result' decreases from 0.991 to 0.000.
SSLfinal_State	0	Result	10.56%	0.991	0.130	0.860595	-1 (44.31% -> 98.20%)	When 'SSLfinal_State' has the value '0', the entropy of 'Result' decreases from 0.991 to 0.130.
URL_of_Anchor	-1	Result	29.69%	0.991	0.087	0.903474	-1 (44.31% -> 98.90%)	When 'URL_of_Anchor' has the value '-1', the entropy of 'Result' decreases from 0.991 to 0.087.
URL_of_Anchor	1	Result	22.04%	0.991	0.334	0.656953	1 (55.69% -> 93.84%)	When 'URL_of_Anchor' has the value '1', the entropy of 'Result' decreases from 0.991 to 0.334.

Task 4: Use `fieldsummary` to see a basic field summary.

21. Examine the `retail` index over all time.
22. Set `maxvals` to 2.
23. Do you notice anything unexpected?

```
index=retail
| fieldsummary maxvals=2
```

Results example:

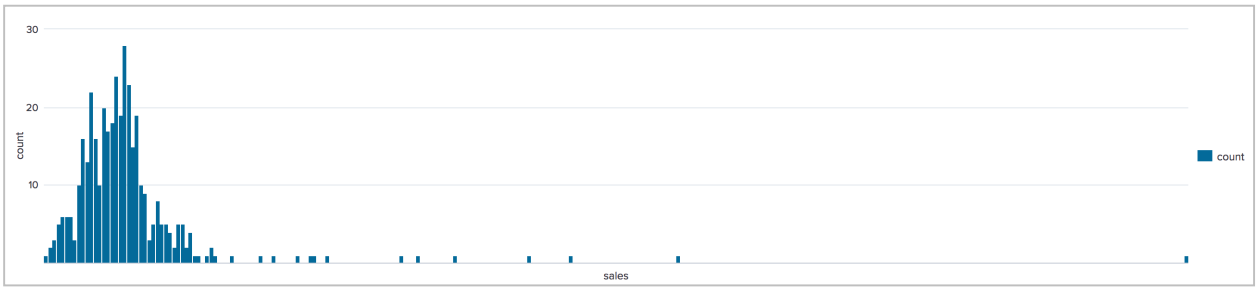
field	count	distinct_count	is_exact	max	mean	min	numeric_count	stdev	values
Country	541909	10	0				0		[{"value":"United Kingdom","count":495478}, {"value":"Germany","count":9495}]
CustomerID	406829	10	0	18287	15287.690570239585	12346	406829	1713.6003033216	[{"value":"17841","count":7983}, {"value":"14911","count":5903}]
Description	540455	10	0	20713	20713	20713	1	0	[{"value":"PAPER CHAIN KIT 50'S CHRISTMAS","count":1210}, {"value":"RABBIT NIGHT LIGHT","count":1051}]
InvoiceNo	541909	10	0	581587	559965.752026781	536365	532618	13428.417280796157	[{"value":"573585","count":1114}, {"value":"581219","count":749}]

Task 5: Use `bin` and `makecontinuous` to make a histogram of vendor sales at Buttercup Games.

24. Examine `vendor_sales` sourcetype for the last month.
25. Sum the `price` by `Vendor` and rename the value `sales`
26. Group the sums into ranges with the `bin` command.
27. Count by `sales`
28. Add `makecontinuous` to make the sales field continuous.
    - What do you see from the distribution? (Do you see any outliers?)
29. Save your results as a report and name it **L2S4**

```
sourcetype=vendor_sales earliest=-1mon
| stats sum(price) as sales by Vendor
| bin sales span=100
| stats count by sales
| makecontinuous sales
```

Results example:



OPTIONAL TASK

Task 6: Use `bin` and `makecontinuous` to make a histogram of electrical output.

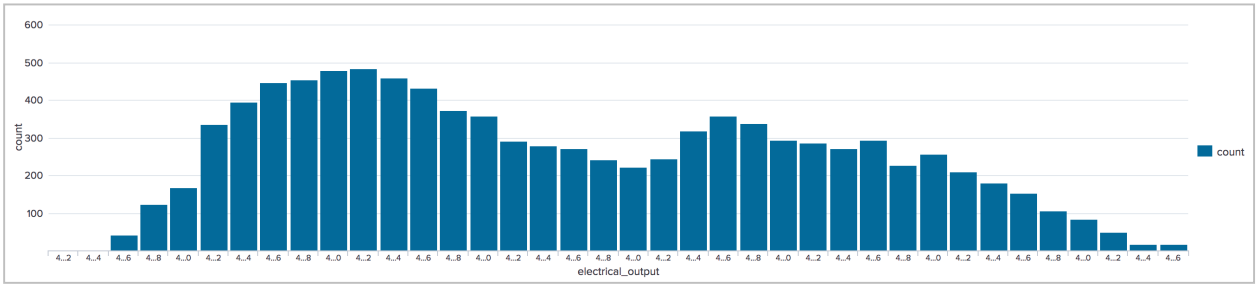
30. Examine the powerplant dataset.
31. Bin `electrical_output` with a `span` of 2.
32. Count by `electrical_output` and make it continuous.

    What do we see from the distribution?

    (bimodal “camel" / not normally distributed)
33. Save your results as a report and name it **L2S5**

```
| inputlookup powerplant.csv
| bin electrical_output span=2
| stats count by electrical_output
| makecontinuous electrical_output
```

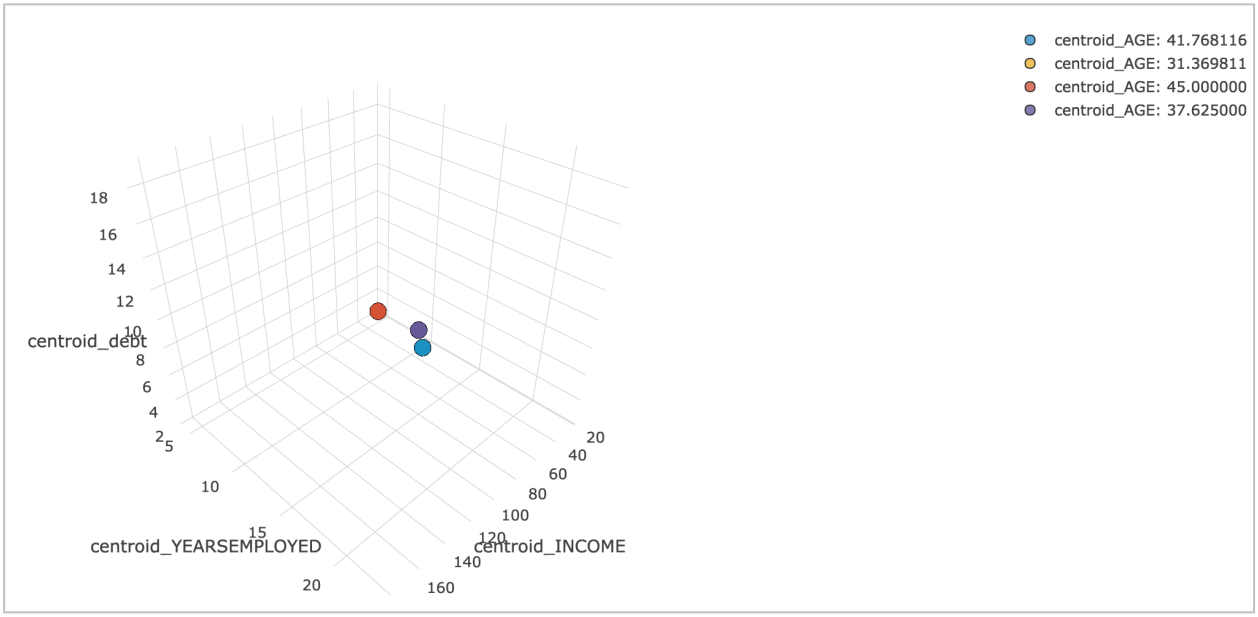
Results example:



Task 7: Use `kmeans` to cluster debtors by age, years employed, income, and debt.

34. Examine the **Debt & Default** dataset.
35. Create a new field called `debt` that is the sum of `CARDDEBT` and `OTHERDEBT`
36. Use `kmeans` to create 4 clusters.
37. Cluster by `INCOME`, `debt`, `AGE`, and `YEARSEMPLOYED`
38. Make a table the new fields created by `kmeans` that start with `centroid`.
39. Visualize as a **3D Scatterplot**.
40. Save your results as a report and name it **L2S6**

```
| inputlookup default.csv
| eval debt=(CARDDEBT + OTHERDEBT)
| kmeans k=4 INCOME debt AGE YEARSEMPLOYED
| table centroid*
```



splunk>

Lab Exercise 3 – Machine Learning Workflow

Description

Practice loading and splitting data (the first two steps of the Machine Learning Workflow). Fit, apply, and save a model. Explore the Machine Learning Toolkit showcases.

(1) Load data –from a search, a lookup, a loadjob command, etc.
(2) Split the data into training and testing sets randomly.

Remember, it is necessary to evaluate the model using data not used during its training phase (test data).

Steps

Task 1: Split a dataset into a training set and a test set. Confirm that the data was split.

1. Load the `auto-mpg.csv` dataset.
2. Use the `sample` command with the `partitions` option to split your data into multiple sets.
3. Make sure you also set a seed number for repeatability.
4. Search a subset of the data using the `partition_number` field.
5. Confirm that the data was split by searching the other subset.

```
| inputlookup auto-mpg.csv
| sample partitions=2 seed=123456
| search partition_number=0
```

Results Example

accel	cyl	displ	hp	mpg	name	origin	partition_number	weight	yr
11	8	318	150	18	plymouth satellite	1	0	3436	70
12	8	304	150	16	amc rebel sst	1	0	3433	70
10.5	8	302	140	17	ford torino	1	0	3449	70
9	8	454	220	14	chevrolet impala	1	0	4354	70

```
| inputlookup auto-mpg.csv
| sample partitions=2 seed=123456
| search partition_number=1
```

Results Example

accel	cyl	displ	hp	mpg	name	origin	partition_number	weight	yr
12	8	307	130	18	chevrolet chevelle malibu	1	1	3504	70
11.5	8	350	165	15	buick skylark 320	1	1	3693	70
10	8	429	198	15	ford galaxie 500	1	1	4341	70
10	8	455	225	14	buick estate wagon (sw)	1	1	3086	70

6. OPTIONAL: In a single search, modify one partition and keep the other pure, fit a linear regression from weight and evaluate the performance with the `regressionstatistics` macro.
7. If you completed the optional question, **save** your search as a report called **L3S1**

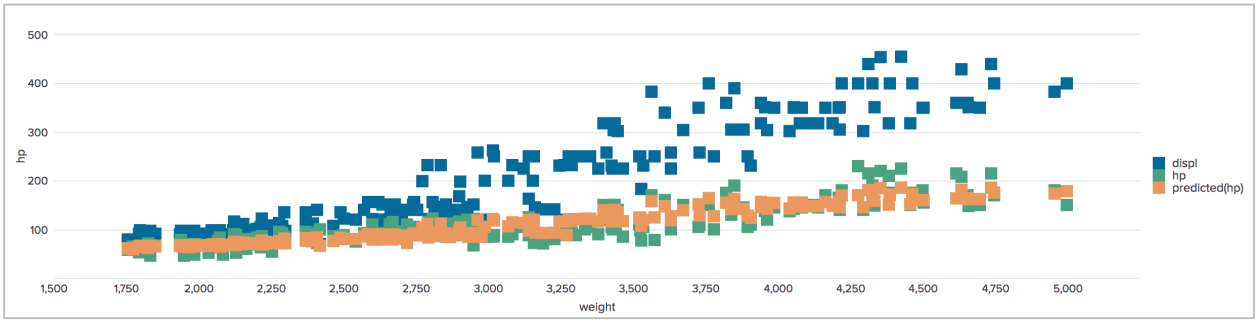
```
| inputlookup auto-mpg.csv
| sample partitions=2 seed=123456
| appendpipe [ | where partition_number=0 | fit LinearRegression mpg from weight into mod2_task1 ]
| where partition_number=1
| apply mod2_task1 as prediction
| `regressionstatistics(mpg, prediction)`
```

Task 2: Fit a linear regression to the automotive dataset.

8. Open the `auto-mpg.csv` dataset.
9. Split the data into 2 partitions and set a seed.
10. Search the first partition.
11. Use `fields` to isolate `hp, weight` and `displ`
12. Fit a linear regression to predict `hp` from `weight` and `displ`
13. Save the model as **LinRegAuto**
14. Use `untable` on `weight, series, and hp`
15. Use `fields` to show those three fields.
16. Visualize as **Scatter**
17. Save your results as a report and name it **L3S2**

```
| inputlookup auto-mpg.csv
| sample partitions=2 seed=123456
| search partition_number=0
| fields hp weight displ
| fit LinearRegression hp from weight displ into LinRegAuto
| untable weight series hp
| fields series weight hp
```

Results Example



18. Does it look like using displacement to help predict horsepower isn't really improving the predictions? What happens if you remove the `displ` field from your search?

    It is underfit

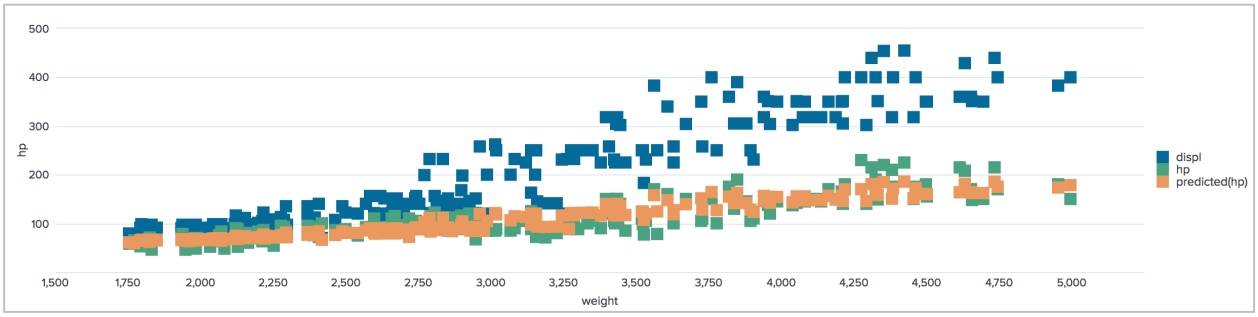
19. Return the `displ` field to your search text and execute the search again so that your model is updated.

Task 3: Apply the linear regression model you made.

20. Open the `auto-mpg.csv` dataset.
21. Split the data into 2 partitions and set a seed.
22. Search the 2nd partition.
23. Use `fields` to isolate `hp`, `weight` and `displ`
24. Apply **LinRegAuto**
25. Use `untable` on `weight`, `series` and `hp`
26. Use `fields` to show those three fields.
27. Visualize as **Scatter**

```
| inputlookup auto-mpg.csv
| sample partitions=2 seed=123456
| search partition_number=1
| fields hp weight displ
| apply LinRegAuto
| untable weight series hp
| fields series weight hp
```

Results Example



Task 4: Save the applied linear regression as a report.

28. You may want to compare how your model performed on this test data versus future test data.
29. Save the applied model visualized as Scatter Chart as a report and name it **L3S3**

Task 5: Explore the Machine Learning Toolkit Showcases.

30. Choose a **Showcase** to examine from the Machine Learning Toolkit.
31. Can you find where the Showcase splits the data for training and testing?
32. Scroll down and examine the visualizations.
33. Adjust some of the settings, click **Fit Model** and notice the change in results.
34. To the right of the Fit Model button, click **Open in Search** to examine how the search was constructed.
35. Copy the | `inputlookup` segment of the search.
36. Click **Experiments > Create New Experiment**, fill out the form and click **Create**
37. Paste your search segment in the search bar, fill in the fields in the Algorithm section, click **Fit Model**
38. Add another field for predicting and click **Fit Model**
39. Change the Algorithm can click **Fit Model**
40. Click the **Experiment History** tab to view all versions of the experiment. You can click **>** for details. Prediction Results show are for the most recent draft of the Experiment.
41. The word Draft highlighted in yellow at the top of your screen reminds you your Experiment hasn't been saved. When you're done exploring, click **Save**
42. Write down any ideas you had for projects you'd like to explore back at your workplace.

OPTIONAL TASK

Task 6: If you have time, experiment with data munging commands (`fillnull`, `rex`, `autoregress`, `eval`, `where`, and `search field=*` to filter out missing values).

HINTS:

```
eval{foo}=1 | fillnull
search field=*
rex mode=sed "s/\W//g"
```

43. Compare results before and after munging your data.
44. Document any searches you may want to use back at work.

Lab Exercise 4 –Algorithms, Preprocessing and Feature Extraction

Description

Practice preprocessing data, using algorithms to build models, and testing models you've built.

Task 1: Select a field to use for building a model to predict home values.

1. Load the housing dataset.
2. Use the `sample` command with the `partitions` option to split your data into multiple sets.
3. Make sure you also set a seed number for repeatability.
4. Search a subset of the data using the `partition_number` field.
5. Preprocess the data using FieldSelector to choose a field that could best predict `median_house_value`

   ```
   | inputlookup housing.csv
   | sample partitions=3 seed=54
   | search partition_number < 2
   | fit FieldSelector median_house_value from * into selector_model
   | fields fs*
   ```
6. Which field did `FieldSelector` choose?

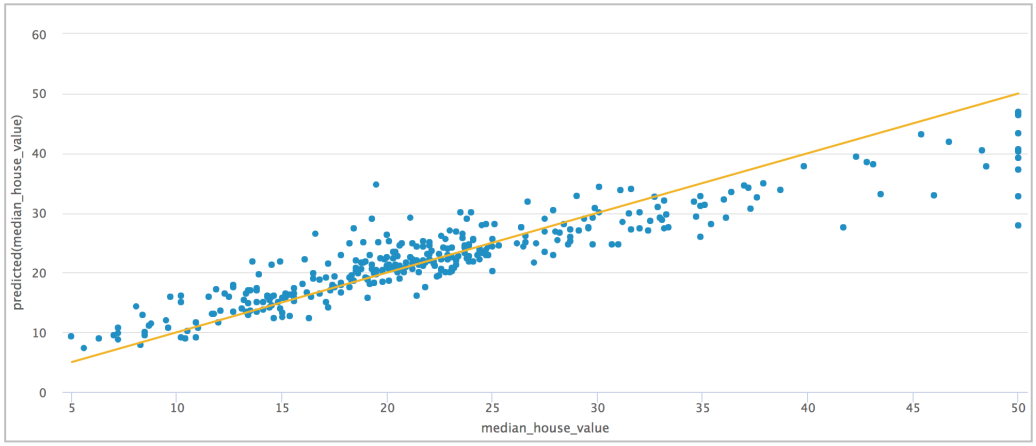
   crime_rate

Task 2: Use the field you determined to fit a model to predict home values.

7. Load the housing dataset.
8. Use the same partition split you did for the previous task.
9. Use the field selected from the previous task to fit a model to predict `median_house_value`
10. Name your model **median_house_value_model**
11. Table the results.
12. Visualize as **Scatter Line Chart**
13. Change the algorithm you chose until you are satisfied with the results. (The model will update).
14. Save your results as a report and name it **L4S1**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number < 2
| fit RandomForestRegressor median_house_value from crime_rate into median_house_value_model
| table median_house_value predicted(median_house_value)
```

Results Example



How might you transform or clean the data to make a more generalizable model?

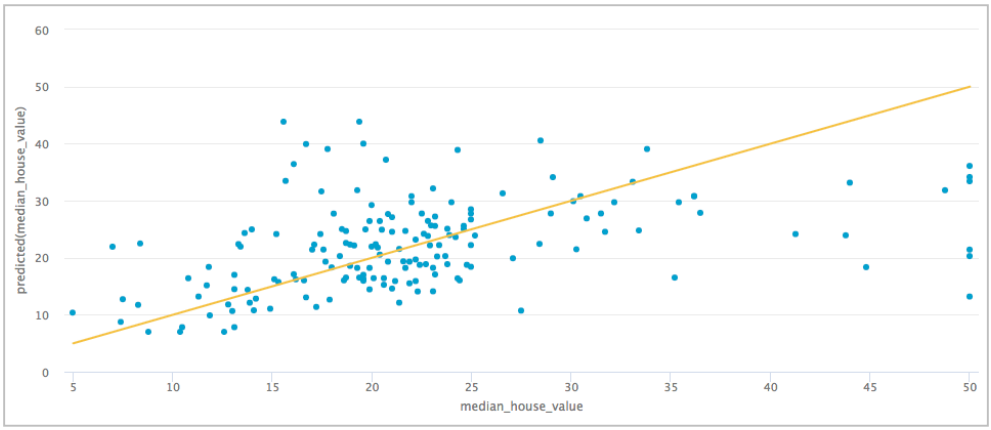
Remove outliers, etc.

Task 3: Test your new model to predict home values.

15. Use the same partitioning and seed you did for the previous task.
16. This time search the test partition.
17. Apply your model.
18. Table the results.
19. Visualize as **Scatter Line Chart**
20. Save your results as a report and name it **L4S2**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number =2
| apply median_house_value_model
| table median_house_value predicted(median_house_value)
```

Results Example



Task 4: Use a macro to assess your model's performance.

21. Use the same partitioning, seed, and partition number you did for the previous task.
22. Use the `regressionstatistics` macro to assess your model's performance.
23. Save your results as a report and name it **L4S3**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number =2
| apply median_house_value_model as prediction
| `regressionstatistics(median_house_value,prediction)`
```

Results Example

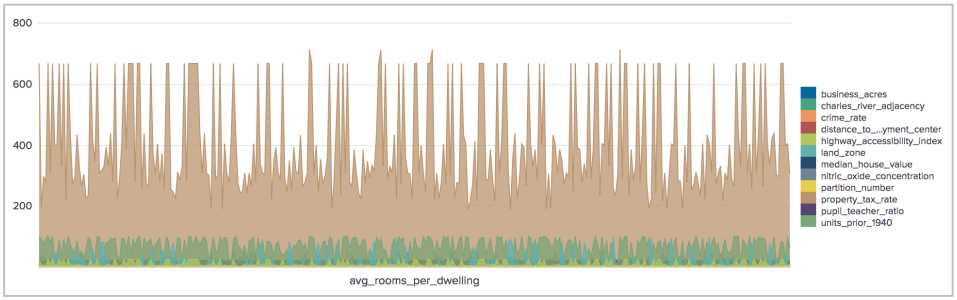
rSquared	RMSE
-0.0480	9.25

Task 5: Scale the housing data using StandardScaler

24. Load and partition the housing data again.
25. Table the fields.
26. Visualize as **Area Chart**
27. Save your results as a report.

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number < 2
```

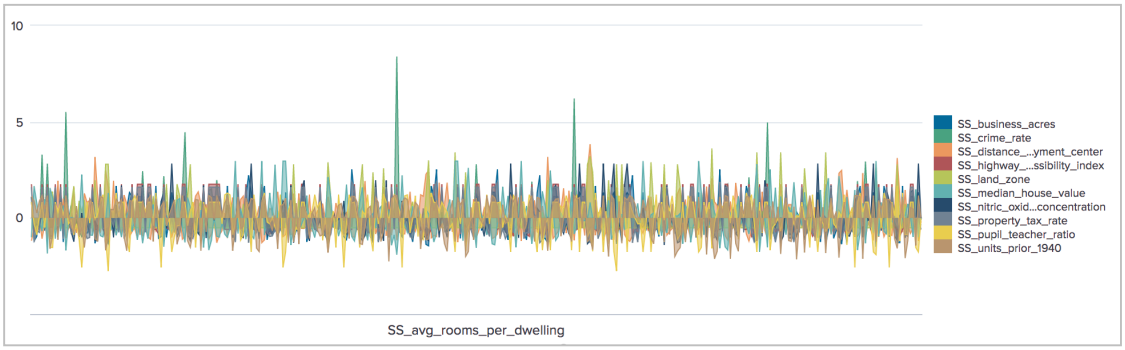
Results Example



28. This time use StandarScaler to scale and normalize these fields: `"avg_rooms_per_dwelling", "business_acres", "crime_rate", "distance_to_employment_center", "highway_accessibility_index", "land_zone", "median_house_value", "nitric_oxide_concentration", "property_tax_rate", "pupil_teacher_ratio", "units_prior_1940"`
29. Name your model **scaler_model**
30. Create a table of the scaled fields.
31. Visualize as **Area Chart**, save as report **L4S5** and compare to the unscaled chart.

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number < 2
| fit StandardScaler "avg_rooms_per_dwelling", "business_acres", "crime_rate", "distance_to_employment_center", "highway_accessibility_index", "land_zone", "median_house_value", "nitric_oxide_concentration", "property_tax_rate", "pupil_teacher_ratio", "units_prior_1940" into scaler_model
| table SS_*
```

Results Example

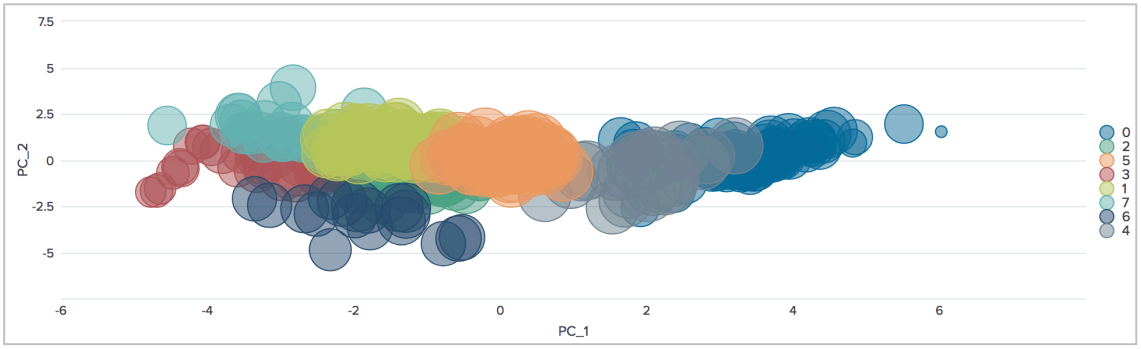


Task 6: Reduce the dimensionality of the data using PCA

32. Load and partition the housing data again.
33. Apply your **scaler_model**
34. Reduce the dimensionality to 3 using PCA
35. Name the model **PCA_reduction**
36. Fit a KMeans model to cluster your data based on the three new principal component fields (which begin with PC_)
37. Table the `cluster` field and the 3 new fields
38. Visualize as **Bubble Chart**
39. Save your results as a report and name it **L4S6**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number < 2
| apply scaler_model
| fit PCA "SS_avg_rooms_per_dwelling", "SS_business_acres", "SS_crime_rate",
"SS_distance_to_employment_center", "SS_highway_accessibility_index", "SS_land_zone",
"SS_median_house_value", "SS_nitric_oxide_concentration", "SS_property_tax_rate",
"SS_pupil_teacher_ratio", "SS_units_prior_1940" k=3 into PCA_reduction
| fit KMeans PC_*
| table cluster PC_*
```

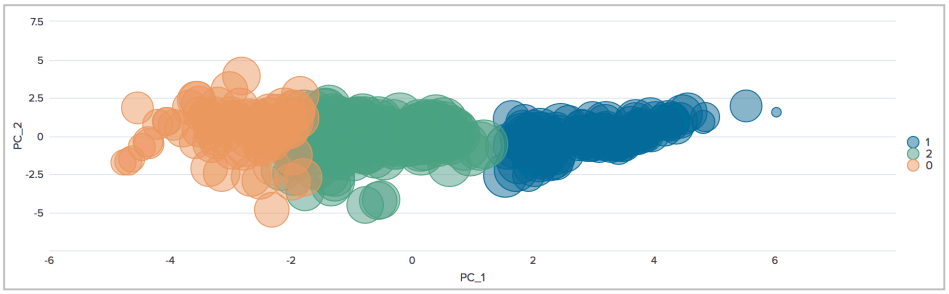
Results Example



40. Try setting a value of 3 groups for the k in KMeans and compare the results

```
| fit KMeans k=3 PC_*
```

Results Example

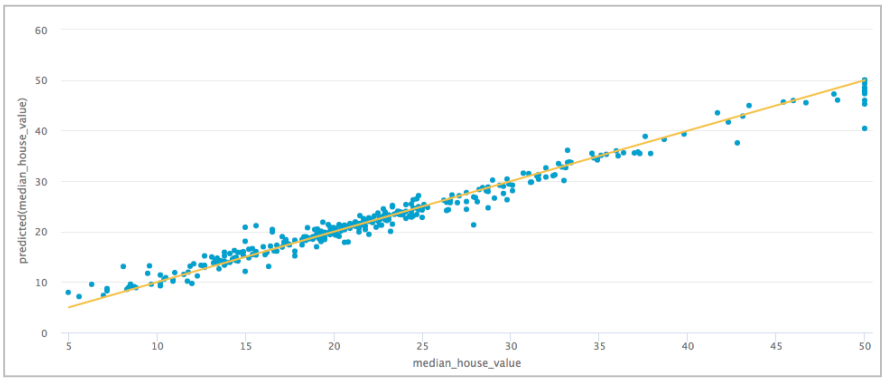


Task 7: Predict the mean house value from the principal components instead of the single field you used before and compare the results.

41. Begin with the previous search.
42. Replace the step for fitting PCA with a step to apply the PCA model (with no value set for k).
43. Fit a model to predict median_house_value using the same algorithm you used before.
44. Save as **PCA_regression**
45. Table median_house_value and predicted(median_house_value)
46. View as **Scatter Line Chart**
47. Save your results as a report and name it **L4S7**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number < 2
| apply scaler_model
| apply PCA_reduction
| fit RandomForestRegressor median_house_value from PC_* into PCA_regression
| table median_house_value predicted(median_house_value)
```

Results Example

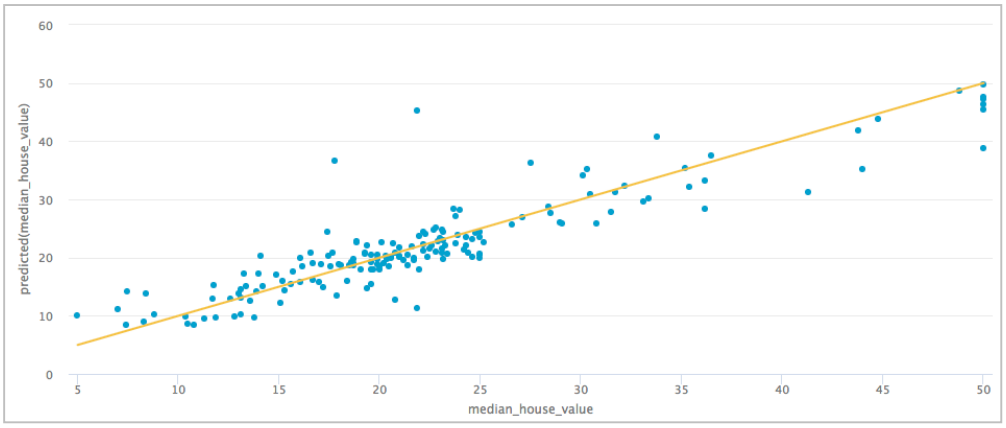


Task 8: Test the new model.

48. Use the previous search.
49. This time, search the test partition of data.
50. Delete the step for fitting the predictive model with a step to apply it.
51. Save your results as a report and name it **L4S8**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number = 2
| apply scaler_model
| apply PCA_reduction
| apply PCA_regression
| table median_house_value predicted(median_house_value)
```

Results Example



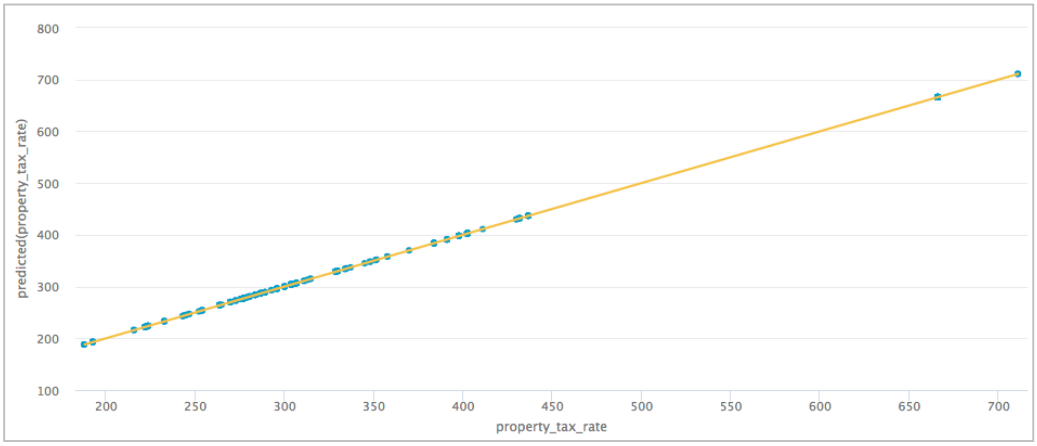
OPTIONAL TASK

Task 9: Predict the property tax rate from the same principal components.

52. Use the training search you used for fitting the previous model.
53. For fitting, replace the regression algorithm with a classification algorithm.
54. Name the model **PCA_classification**
55. View as **Scatter Line Chart**
56. Change classification algorithms and repeat the search until you are satisfied with the results.
57. Save your results as a report and name it **L4S9**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number < 2
| apply scaler_model
| apply PCA_reduction
| fit DecisionTreeClassifier property_tax_rate from PC_* into PCA_classification
| table property_tax_rate, predicted*
```

Results Example

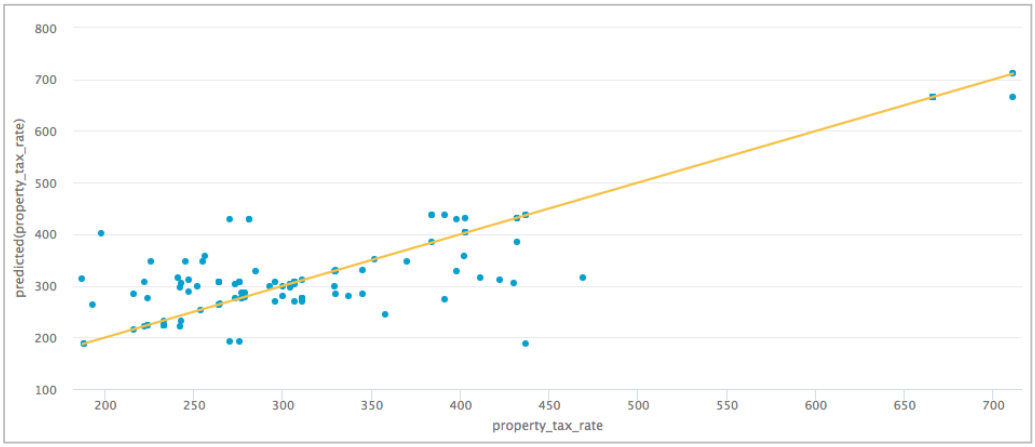


OPTIONAL TASK
Task 10: Test your new model.

58. Use the previous search.
59. Change the partition to search the test data.
60. Replace the fit step with applying the model.
61. View as **Scatter Line Chart**
62. Save your results as a report and name it **L4S10**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number = 2
| apply scaler_model
| apply PCA_reduction
| apply PCA_classification
| table property_tax_rate, predicted*
```

Results Example



OPTIONAL TASK

Task 11: Use a macro to assess your model's performance.

63. Use the same partitioning, seed, and partition number you did for the previous task.
64. Use the classificationreport macro to assess your model's performance.
65. Save your results as a report and name it **L4S11**

```
| inputlookup housing.csv
| sample partitions=3 seed=54
| search partition_number =2
| apply scaler_model
| apply PCA_reduction
| apply PCA_classification as prediction
| `classificationreport(property_tax_rate, prediction)`
```

Results Example

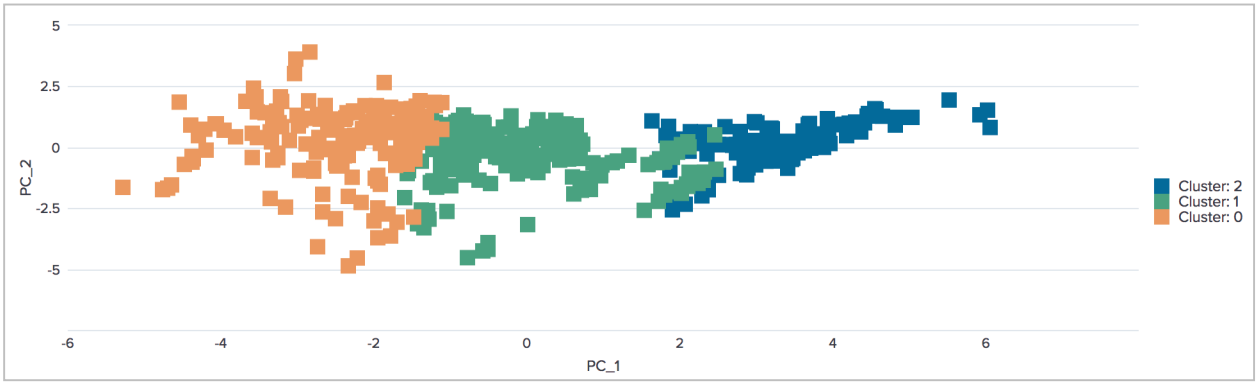
class	accuracy	precision	recall	f1	count
188	0.75	1	0.75	0.86	4
193	1	0.3333333333333333	1	0.5000000000000000	1
198	0	0	0	0	1
222	0.3333333333333333	0.5	0.3333333333333333	0.4	3

OPTIONAL TASK
Task 12: Cluster the data by types of neighborhoods

66. Load the housing data.
67. Apply your **scaler_model**
68. Apply your **PCA_reduction**
69. Fit a KMeans model to build 3 clusters based on the PC_* fields.
70. Name the model **PCA_cluster**
71. Use eval to create a cluster field that begins with “Cluster: “ followed by the cluster number.
72. Table the clusters and the PC_* fields.
73. View as **Scatter Line Chart** with the line set to **off**.
74. Save your results as a report and name it **L4S12**

```
| inputlookup housing.csv
| apply scaler_model
| apply PCA_reduction
| fit KMeans k=3 "PC_1" "PC_2" "PC_3" into PCA_cluster
| eval cluster= "Cluster: " + cluster
| table cluster, "PC_1", "PC_2", "PC_3"
```

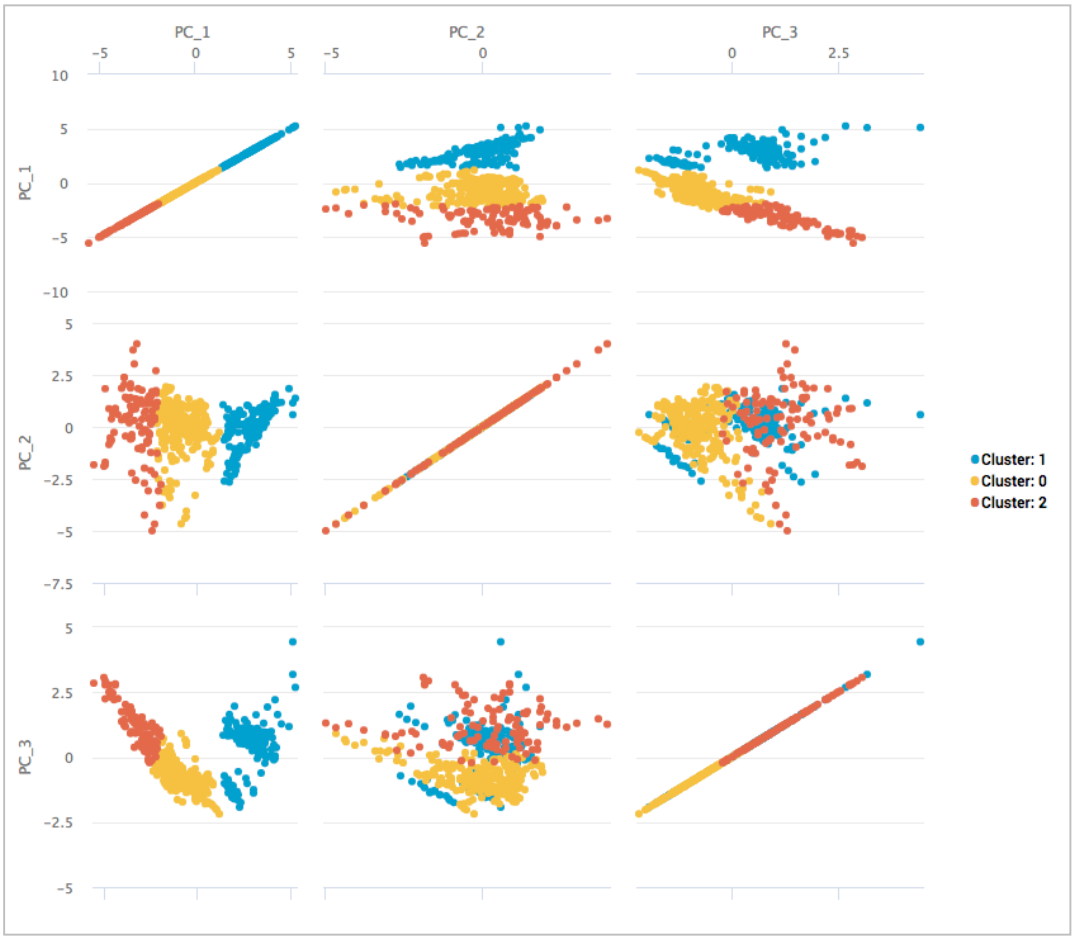
Results Example



OPTIONAL TASK
Task 13: Assess the principal components of the model.

75. Keep the previous search and change the visualization type to **Scatterplot Matrix**
76. Save your results as a report and name it **L4S13**
77. Which principal component seems to have contributed most accurately to the clustering task?
    PC_1
78. Which cluster seems to have been the most challenging to identify?
    Cluster 1
79. What might you do next?
    Examine the characteristics of each cluster
    Consider how meaningful the grouping strategy is to your purpose.
    Improve performance by reducing the number of principal components.

Results Example



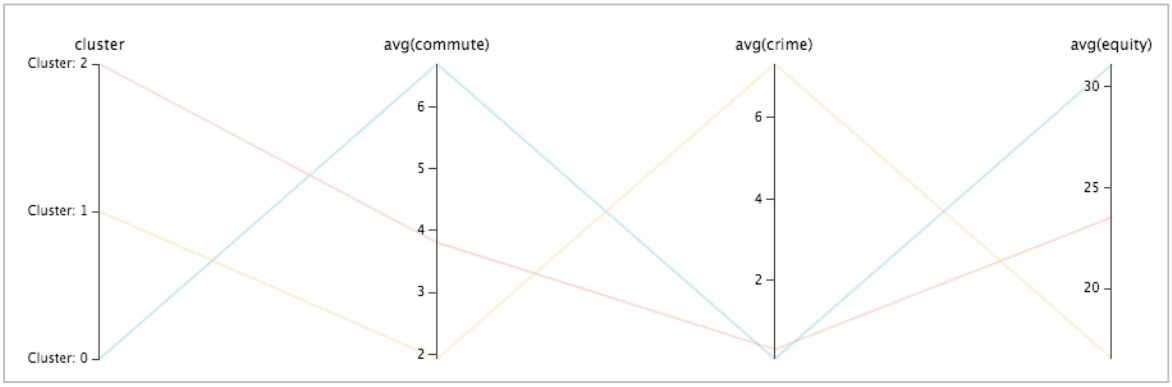
OPTIONAL TASK

Task 14: Examine the characteristics of the clusters.

80. Use the previous search but delete the last step of tabling the clusters by principal component.
81. Use the **Statistics** tab to examine the original data and make a list of fields important to you.
82. Use eval to shorten the names of those important fields.
83. Chart the important fields by cluster.
84. Visualize as **Parallel Coordinates**
85. Save your results as a report and name it **L4S14**

```
| inputlookup housing.csv
| apply scaler_model
| apply PCA_reduction
| apply PCA_cluster
| eval cluster= "Cluster: " + cluster
| eval commute=distance_to_employment_center, crime=crime_rate, equity=median_house_value
| chart avg(commute), avg(crime), avg(equity) by cluster
```

Results Example



Task 15: Try similar tasks using the Machine Learning Toolkit Experiments

Explore the **Predict Numeric Series** and **Classification** showcases and how to complete a typical split-fit-apply procedure:

86. In the **CLASS: Analytics & Data Science** app, look through the data sets and choose one that requires prediction / classification.
87. **Predict Numeric Fields > Predict Server Power Consumption**
88. Wait until the model is fit. (the **Fit Model** button goes from green to gray: "Fitting Model" and back to green.) Notice that the default uses all the fields to predict power consumption.
89. Scroll down and examine the visualizations.
90. Scroll back up, and to the right of the Fit button, click **Open in Search**
91. Notice that the search text is similar to the searches you've been doing. Copy the search text
92. Close the **Show SPL** window, scroll up, and click **Experiments**
93. Click **Create New Experiment**
94. Name your experiment **L4S15 Predict Server Power Consumption from All Fields** and click **Create**
95. In the box under Enter a search, paste the search you copied earlier
96. Read through the search text and set the Algorithm section to match
    - Check the estimate the intercept field
    - In the Field to predict dropdown, select **ac_power**
    - In the Fields to use for predicting dropdown, click **Select All Fields**
97. Adjust some settings and fields and save them under new experiments for comparison
98. Try the same process for other classification or predict numeric fields tasks. Take into consideration:
    - Is the target field to predict numeric or categorical?
    - Which algorithm(s) can be used?

If the target field is numeric, you can use DecisionTreeRegressor, KernelRidge, LinearRegression, RandomForestRegressor, Lasso, ElasticNet, Ridge, or SGDRegressor.

If the target field is categorical, you can use DecisionTreeClassifier, LogisticRegression, RandomForestClassifier, SVM, GaussianNB, SGDClassifier.

If the target field is categorical as well as binary, you could also use BernoulliNB.

Remember that the train-test split is unnecessary for unlabeled (clustering) tasks.

https://www.splunk.com/pdfs/solution-guides/machine-learning-quick-ref-guide.pdf

Lab Exercise 5 – Market Segmentation & Transactional Analysis

Description

Use clustering to find market segments and groups of similar customers. Review the retail index from the Search tab. Find similar groupings of customers using k-means clustering and visualize them in a scatterplot.

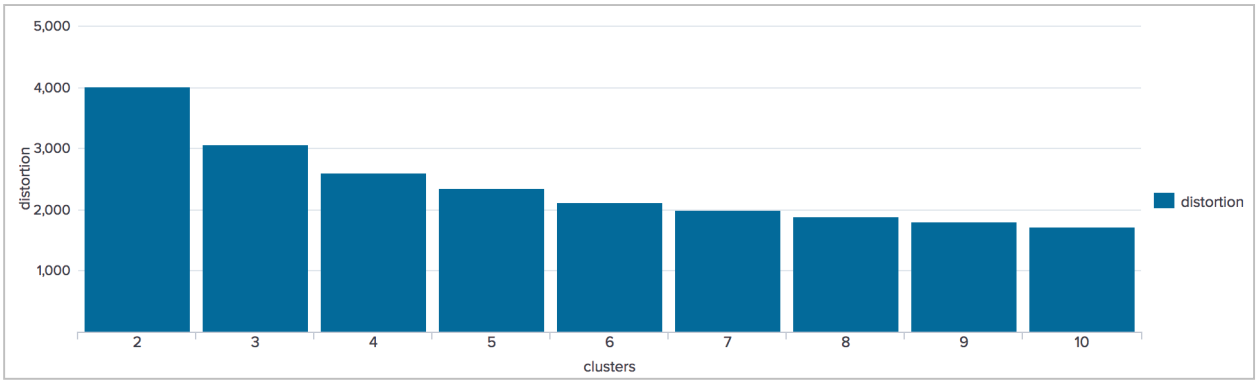
Steps

Task 1: Extract relevant numeric features from the retail index, normalize the data & determine a value for k

1. Create a search of the retail index over **All time** and limit it to customers in the United Kingdom.
2. Use `eval` to create a monetary field by multiplying `Quantity` and `UnitPrice`
3. Use `eval` to create a `recency` field by subtracting `_time` from now.
4. Use `stats` to compute minimum recency as `c_recency`, the distinct count of invoices as `c_frequency` and the `sum` of monetary values as `c_monetary` by `CustomerID`
5. Use `eventstats` to calculate the 99th percentile of each feature. Then, use `where` to filter out values that are above that threshold.
6. Use `StandardScaler` to normalize the data. (You may want to use event sampling for speed.)
7. Use `kmeans` (the command) with a `range` of k (e.g. 2-10)
8. Use `table` to show `clusters` and `distortion` as a **Column Chart**
9. Save your results as a report and name it **L5S1**

```
index=retail Country="United Kingdom" Quantity>0 UnitPrice>0 CustomerID=*
| eval monetary = Quantity * UnitPrice
| eval recency = now() - _time
| stats min(recency) as c_recency, sum(monetary) as c_monetary, dc(InvoiceNo) as c_frequency by CustomerID
| eventstats p99(c_frequency) as p99_f, p99(c_monetary) as p99_m, p99(c_recency) as p99_r
| where c_recency < p99_r AND c_frequency < p99_f AND c_monetary < p99_m
| fit StandardScaler c_*
| kmeans k=2-10 SS_c_frequency SS_c_monetary SS_c_recency
| table clusters distortion
```

Results example:



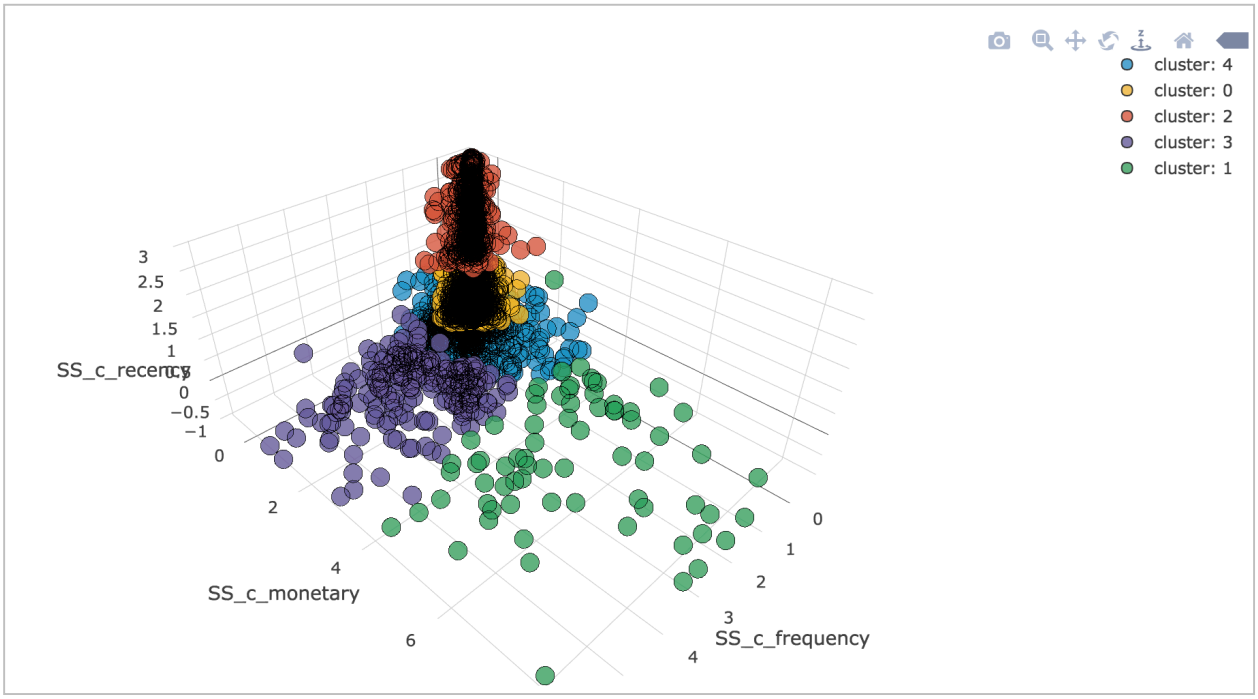
Task 2: Fit a `KMeans` model.

10. Fit a `KMeans` model on the z-scores from the previous search before using the `fit` command (you'll also need `eventstats` and `where`). **Save** the model.
    - Remember that `KMeans` is case sensitive when used with `fit`
11. Also save your results as a report and name it **L5S2**

```
index=retail Country="United Kingdom" Quantity>0 UnitPrice>0 CustomerID=*
| eval monetary = Quantity * UnitPrice
| eval recency = now() - _time
| stats min(recency) as c_recency count as c_frequency, sum(monetary) as c_monetary by
CustomerID
| eventstats p99(c_frequency) as p99_f, p99(c_monetary) as p99_m, p99(c_recency) as p99_r
| where c_recency < p99_r AND c_frequency < p99_f AND c_monetary < p99_m
| fit StandardScaler c_*
| fit KMeans k=5 SS* into clusterer
| table cluster SS_*
```

12. View the results as a 3D Scatter Chart.

Results Example



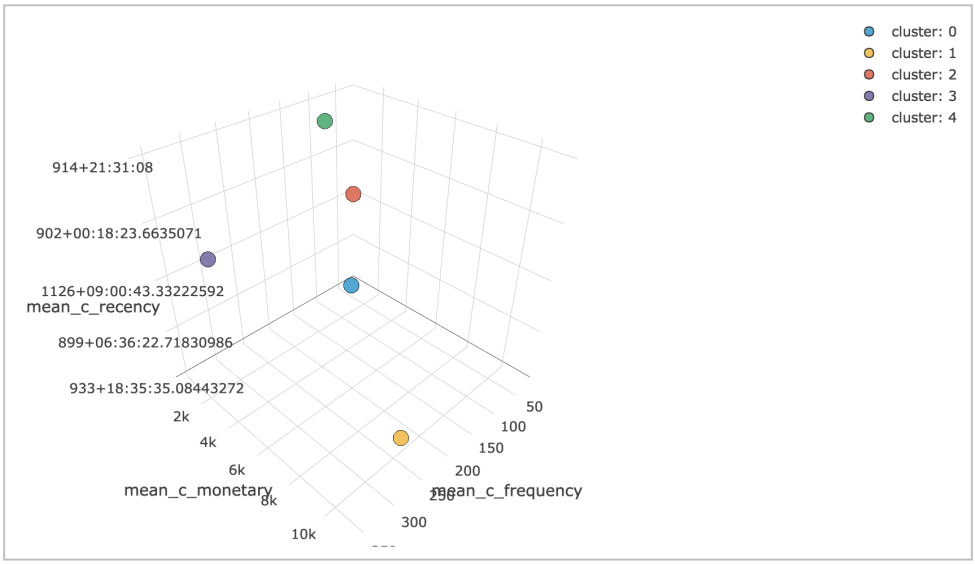
Task 3: Examine the average values of frequency, monetary, and recency fields within each cluster.

13. To the previous search, add a `stats` command for `mean` as `mean_* by cluster`
14. Use `eval` and `to string` to convert `mean_c_recency` to a readable format.
15. View in the **Statistics** tab and (optionally) in **3D Scatterplot**
16. Save your results as a report and name it **L5S3**

```
index=retail Country="United Kingdom" Quantity>0 UnitPrice>0 CustomerID=*
| eval monetary = Quantity * UnitPrice
| eval recency = now() - _time
| stats min(recency) as c_recency count as c_frequency, sum(monetary) as c_monetary by
CustomerID
| eventstats p99(c_frequency) as p99_f, p99(c_monetary) as p99_m, p99(c_recency) as p99_r
| where c_recency < p99_r AND c_frequency < p99_f AND c_monetary < p99_m
| fit StandardScaler c_*
| apply clusterer
| table cluster c_*
| stats mean(*) as mean_* by cluster
| eval mean_c_recency = tostring(mean_c_recency, "duration")
| table cluster mean_c_*
```

Results Example:

cluster	mean_c_frequency	mean_c_monetary	mean_c_recency
0	37.40052770448549	634.4819118733509	933+18:35:35.08443272
1	282.5211267605634	11150.518169014083	899+06:36:22.71830986
2	26.104097452934663	418.2920166112958	1126+09:00:43.33222592
3	356.4454976303318	3919.5716587677734	902+00:18:23.6635071
4	137.26502732240436	2398.1514617486337	914+21:31:08



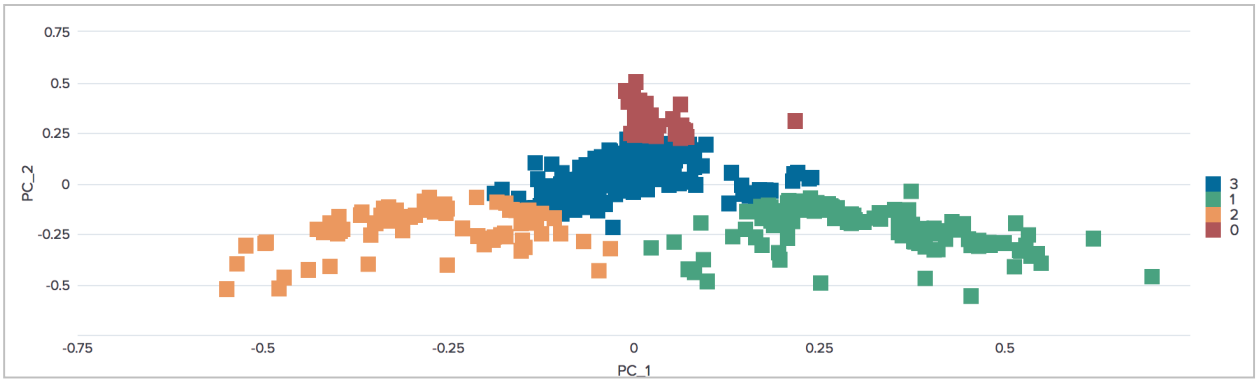
17. How would you describe a typical member of one of the clusters?

Task 4: Cluster the U.K. customers in the retail index based on the descriptions of the products they purchased.

18. Use 1:100 event sampling.
19. Use `stats` to count by `CustomerID` and `Description`
20. Use `fit` and `TFIDF` on `Description`
21. Reduce the dimensionality to two with `PCA`.
22. Cluster with `KMeans k=4` on the principal components.
23. Use `table` to show `clusters` and `PC*`
24. View as **Scatter**.
25. Save your results as a report and name it **L5S4**

```
index=retail Country="United Kingdom" Quantity>0 UnitPrice>0 CustomerID=*
| stats count by CustomerID Description
| fit TFIDF Description
| fit PCA k=2 Description_*
| fit KMeans k=4 PC_*
| table cluster PC*
```

Results example:

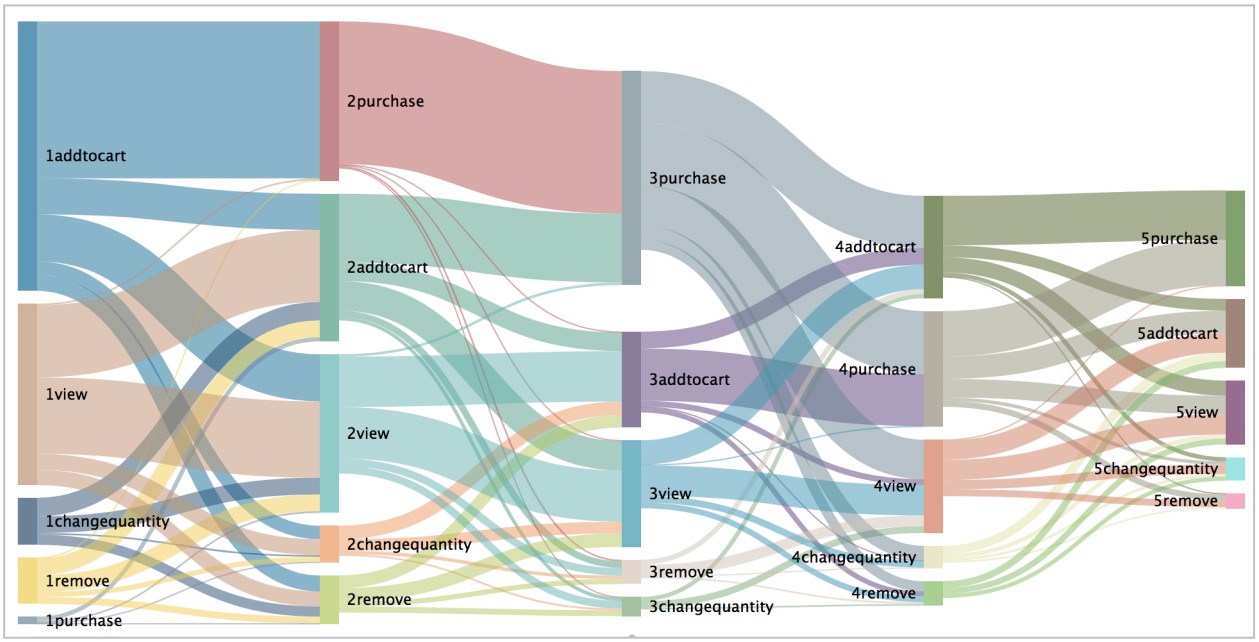


Task 5: Build meta transactions by `JSESSIONID` using `reverse`, `streamstats`, and `autoregress`, to examine web purchase action stages.

26. Search `sourcetype=access_combined`
27. Use `reverse` to put the events in order of oldest first.
28. Use `streamstats` to count stages by `JSESSIONID`
29. Use `eval` to concatenate actions to stages.
30. Use `sort` to sort by `JSESSIONID`
31. Use `autoregress` to create the previous_action of each action as a new field.
32. Use `where` to start with stage 1 and end at stage 6.
33. Use `stats` to count by previous_action and action.
34. Visualize as **Sankey Diagram**
35. Save your results as a report and name it **L5S5**

```
sourcetype=access_combined action=* earliest=-1d
| reverse
| streamstats count as stage by JSESSIONID
| eval action = stage.action
| sort JSESSIONID
| autoregress action as previous_action
| where stage > 1 AND stage < 6
| stats count by previous_action action
```

Results Example



splunk>

Lab Exercise 6 – Anomaly Detection

Description

Use the `cluster` command to find and analyze potential anomalies with multiple data sources.

Task 1: Use the cluster command.

1. Examine clusters over the last day and compare them to clusters over the last hour to determine if there are clusters that only exist in the last hour.
2. Search `sourcetype=cisco*` over the last day.

   NOTE: Use `earliest=-1d` in your search string.
3. Use the `cluster` command with a threshold of 0.8; set `labelonly` too `true` and `showcount` to `true`.
4. Use the `stats` command to list the values and the minimum time for each cluster label (by `cluster_label, cluster_count`).
   - List values of `_raw` as `raw`
   - Get the minimum `_time` as `time`
5. Use the `eval` command to create a field named `one_hour_ago`
6. `hour` should equal the current time minus 60*60.
7. Use the `where` command to find clusters where the value of the `time` field is greater than the value of the `one_hour_ago` field.
8. In case there are lots of clusters, to see the most populous clusters first, sort the results in descending order by the size of the cluster.
9. To make the results easier to read, use the `fields` command to return the `cluster_label, cluster_count`, and `raw` fields.
10. Save your results as a report and name it **L6S1**

```
index=main sourcetype=cisco* earliest=-1d
| cluster field=punct t=0.8 labelonly=t showcount=t match=ngramset
| stats values(_raw) as raw min(_time) as time by cluster_label, cluster_count
| eval one_hour_ago = now() - (60 * 60)
| where time > one_hour_ago
| sort -cluster_count
| fields cluster_count, cluster_label, raw
```

Results example:

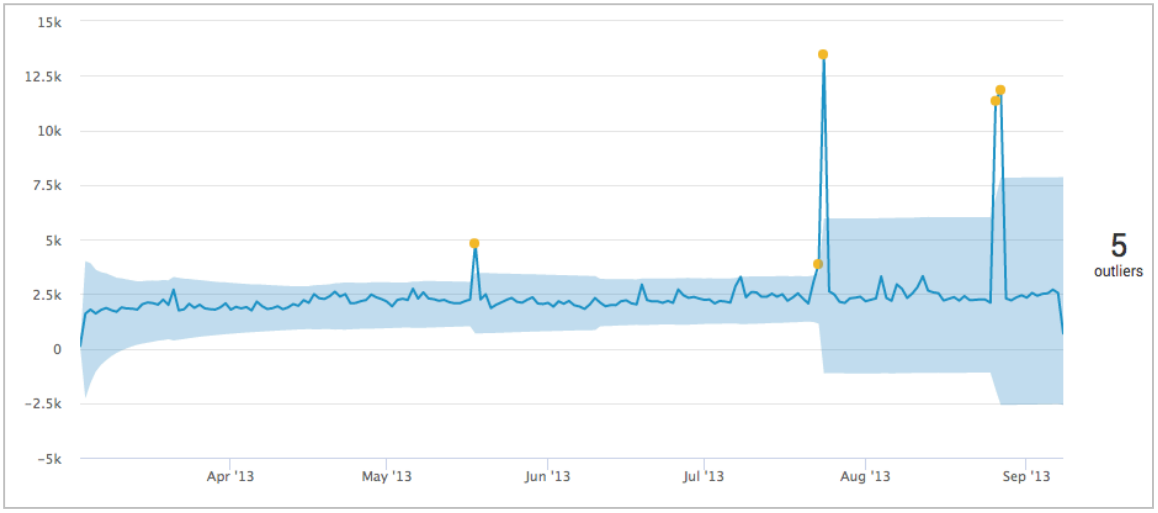
cluster_count	cluster_label	raw
1	8	Tue Sep 25 21:11:30 2018 Info: MID 245549 Subject '[users@httpd] Re: make mod_cache not cache cookies but cache contents from\r\n application side'

Task 2: Find anomalies in the port scans dataset.

11. Load the `portscans` data with `inputlookup`
12. Use `bin` to get a `span` of one day.
13. Use stats to count by _time
14. Use `streamstats` with a window of 100 to track the count averages and standard deviations.
15. Create upper and lower bounds for standard deviation using `eval` with a multiplier of your choice.
16. Visualize as an **Outlier** chart.
17. Save your results as a report and name it **L6S2**

```
| inputlookup portscans.csv
| bin _time span=1d
| stats count by _time
| streamstats window=100 avg(count) as avg, stdev(count) as stdev
| eval multiplier = 3
| eval lower_bound = avg - (stdev * multiplier)
| eval upper_bound = avg + (stdev * multiplier)
| eval outlier = if(count < lower_bound OR count > upper_bound, 1, 0)
| table _time count lower_bound upper_bound outlier
```

Results Example

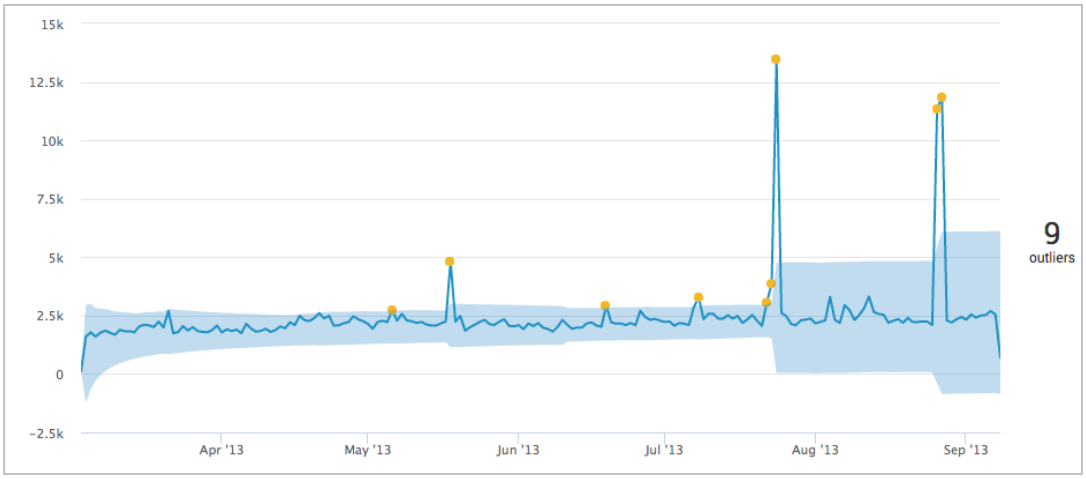


Task 3: Modify anomaly thresholds.

18. Adjust the thresholds and observe the results.
19. Save your results as a report and name it **L6S3**

```
| inputlookup portscans.csv
| bin _time span=1d
| stats count by _time
| streamstats window=100 avg(count) as avg, stdev(count) as stdev
| eval multiplier = 2
| eval lower_bound = avg - (stdev * multiplier)
| eval upper_bound = avg + (stdev * multiplier)
| eval outlier = if(count < lower_bound OR count > upper_bound, 1, 0)
| table _time count lower_bound upper_bound outlier
```

Results Example

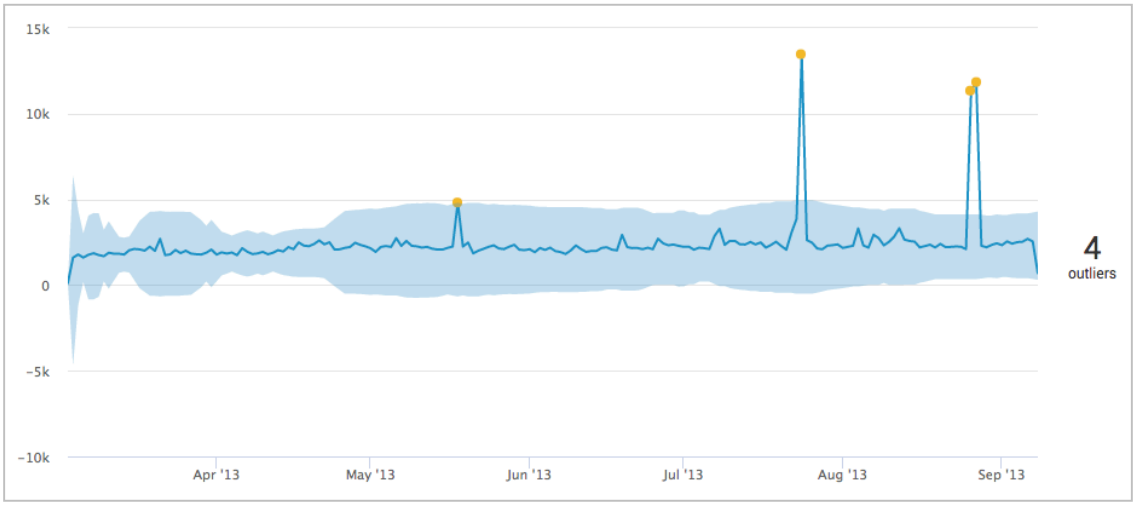


Task 4: Examine the differences between standard deviation and median absolute deviation results.

20. Compare the standard deviation results to median absolute deviation using `streamstats`
21. Save your results as a report and name it **L6S4**

```
| inputlookup portscans.csv
| bin _time span=1d
| stats count by _time
| streamstats window=100 median(count) as medianCount
| eval absDev=(abs('count'-medianCount))
| streamstats window=100 median(absDev) as medianAbsDev
| eval lowerBound=(medianCount-medianAbsDev*15),
upperBound=(medianCount+medianAbsDev*15)
| eval outlier=if(count < lowerBound OR count > upperBound, 1, 0)
| table _time count lowerBound upperBound outlier
```

Results Example

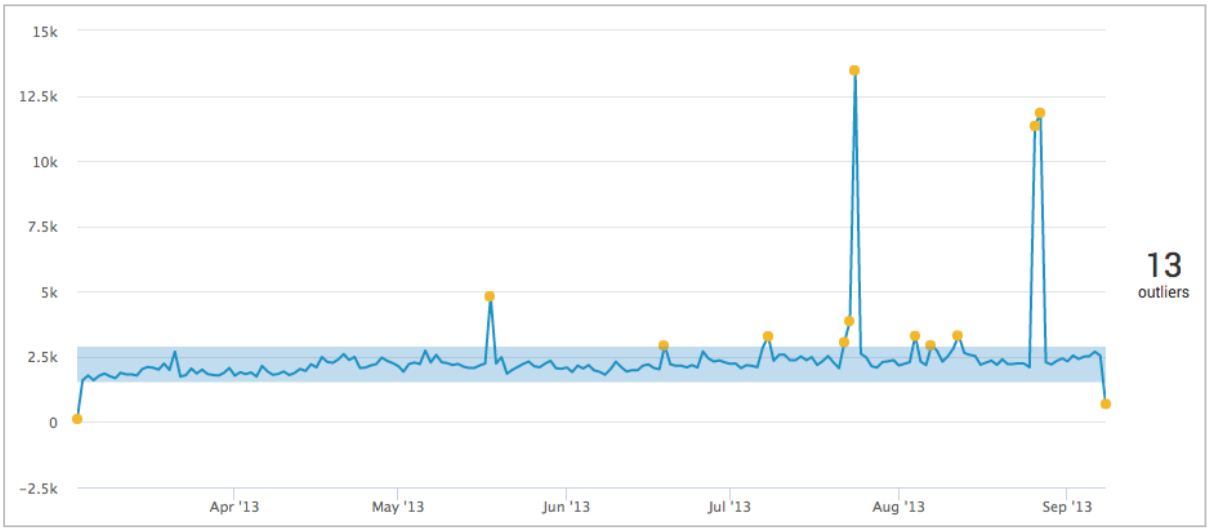


Task 5: Examine the differences between standard deviation and IQR results.

22. Compare the standard deviation results to IQR using `timechart` and `eventstats`
23. Save your results as a report and name it **L6S5**

```
| inputlookup portscans.csv
| timechart span=1d count
| eventstats median(count) as median, p25(count) as p25, p75(count) as p75
| eval IQR = p75 - p25
| eval multiplier = 2
| eval lower_bound = median - (IQR * multiplier)
| eval upper_bound = median + (IQR * multiplier)
| eval outlier = if(count < lower_bound OR count > upper_bound, 1, 0)
| table _time count lower_bound upper_bound outlier
```

Results Example



OPTIONAL TASK

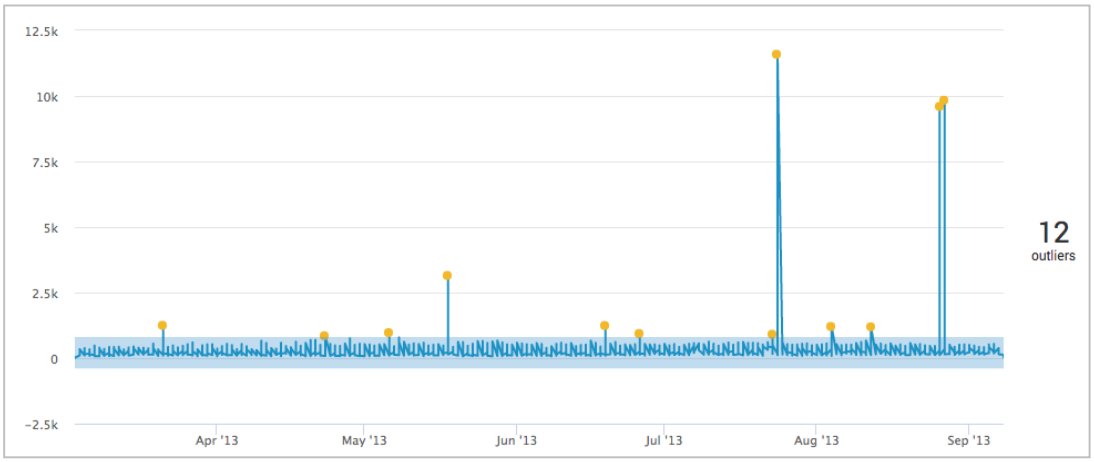
Task 6: Examine anomalies per host using the bin command and IQR.

24. Change the search to calculate anomalies per host (rather than using only _time) and use `bin`
25. Save your results as a report and name it **L6S6**

Sample Search:

```
| inputlookup portscans.csv
| bin _time span=1d
| stats count by _time, host
| eventstats median(count) as median, p25(count) as p25, p75(count) as p75
| eval IQR = p75 - p25
| eval multiplier = 2
| eval lowerBound = median - (IQR * multiplier)
| eval upperBound = median + (IQR * multiplier)
| eval outlier = if(count < lowerBound OR count > upperBound, 1, 0)
| table _time count lowerBound upperBound outlier
```

Results Example

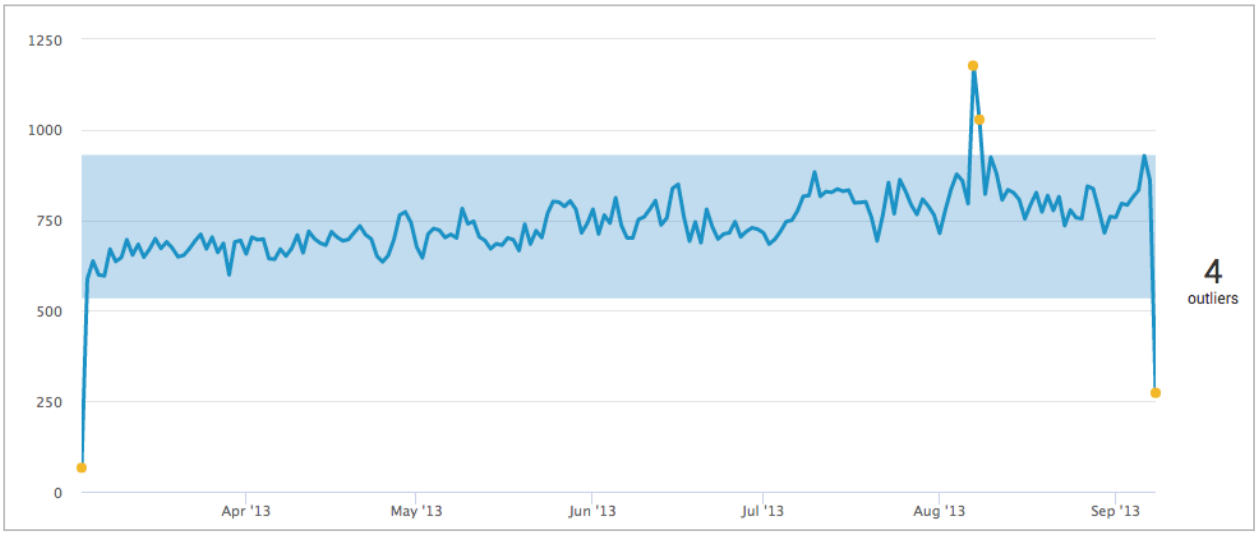


OPTIONAL TASK
Task 7: Examine a distinct count of sources.

26. Try changing `count` to a distinct count of `src` and removing by host
27. Save your results as a report and name it **L6S7**

```
| inputlookup portscans.csv
| timechart span=1d dc(src) as distinctCount
| eventstats median(distinctCount) as median, p25(distinctCount) as p25, p75(distinctCount) as p75
| eval IQR = p75 - p25
| eval multiplier = 2
| eval lower_bound = median - (IQR * multiplier)
| eval upper_bound = median + (IQR * multiplier)
| eval outlier = if(distinctCount < lower_bound OR distinctCount > upper_bound, 1, 0)
| table _time distinctCount lower_bound upper_bound outlier
```

Results Example



OTIONAL TASK

Task 8: Examine a novel search.

28. Examine the following outliers search of call center data, where `count` is the metric of interest. (Please run the search in the **MLTK** app, open in Search, and view its **Statistics** tab.)

```
| inputlookup call-center.csv
| eval _time=strptime(_time, "%Y-%m-%dT%H:%M:%S")
| bin _time span=15m
| eval HourOfDay=strftime(_time, "%H")
| eval BucketMinuteOfHour=strftime(_time, "%M")
| eval DayOfWeek=strftime(_time, "%A")
| stats avg(count) as avg stdev(count) as stdev by
HourOfDay,BucketMinuteOfHour,DayOfWeek,source
| eval lowerBound=(avg-stdev*exact(2)), upperBound=(avg+stdev*exact(2))
| fields lowerBound,upperBound,HourOfDay,BucketMinuteOfHour,DayOfWeek,source
| outputlookup state.csv
```

Results Example

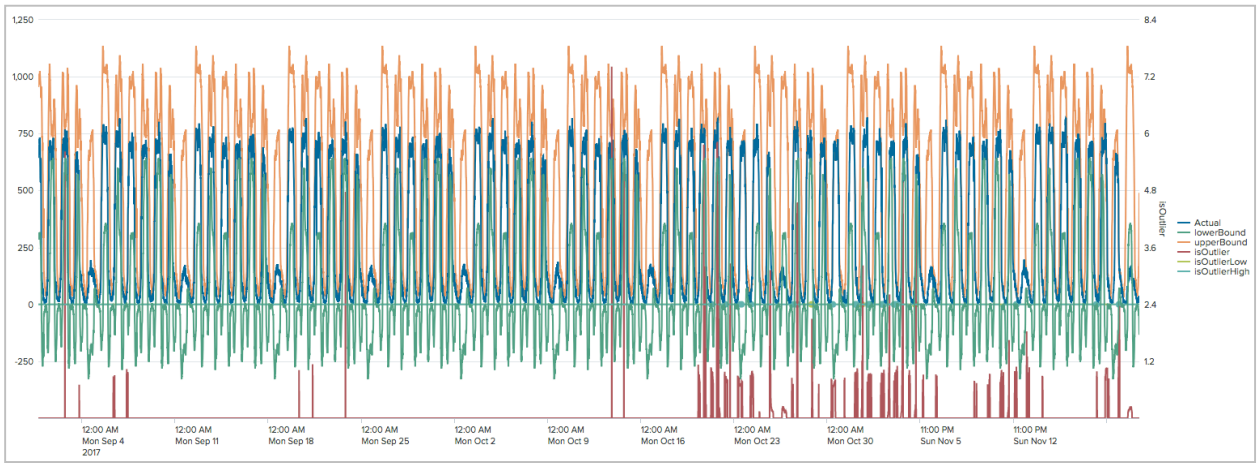
lowerBound	upperBound	HourOfDay	BucketMinuteOfHour	DayOfWeek	source
1	1	00	00	Friday	si_active_agents
312.1933842321313	970.6399491012019	00	00	Friday	si_call_volume
425.525312212663	1892.308021120671	00	00	Friday	si_groups_mapping
17.241499387023070	84.758500612976930	00	00	Friday	si_kpi_elements_cti_asgi
220	220	00	00	Friday	si_kpi_elements_sgi
1	1	00	00	Monday	si_active_agents

29. Save the search as **L6S8**
30. What might the purpose of the outputlookup step be?
    To persist the data for future use
31. What might the use of the by clause be?
    To split events by time spans, entity types, etc. to make the behavior record more useful

32. Examine and execute the following search. Visualize as **Line** with **overlays** of `IsOutlier`, `IsOutlierHigh` and `IsOutlierLow` and with View as Axis s set to **On**. Save it as **L6S9**

```
| inputlookup call-center.csv
| eval _time=strptime(_time, "%Y-%m-%dT%H:%M:%S")
| where source="si_call_volume"
| bin _time span=15m
| eval HourOfDay=strftime(_time, "%H")
| eval BucketMinuteOfHour=strftime(_time, "%M")
| eval DayOfWeek=strftime(_time, "%A")
| stats max(count) as Actual by
HourOfDay,BucketMinuteOfHour,DayOfWeek,source,_time
| lookup state.csv HourOfDay as HourOfDay BucketMinuteOfHour as
BucketMinuteOfHour DayOfWeek as DayOfWeek source as source OUTPUT upperBound
lowerBound
| search Actual="*"
| eval isOutlierLow=if(Actual < lowerBound , abs(Actual-
lowerBound)/lowerBound, 0)
| eval isOutlierHigh=if(Actual > upperBound, abs(Actual-
upperBound)/upperBound, 0)
| eval isOutlier=if(Actual < lowerBound OR Actual > upperBound,
abs(Actual)/abs(upperBound-lowerBound), 0)
| fields _time, "Actual", lowerBound, upperBound,
isOutlier,isOutlierLow,isOutlierHigh
```

Results Example



33. What might the purpose of `| search Actual="*"` be?

    To limit the results to only events with count values

34. Why might there be three evals statements determining outliers?

    To document outliers in general, plus specify those that are above the high threshold, and those that are below the low threshold. (A concerningly low call volume probably requires a different response than a high call volume)

35. Examine this last search snippet and save it as L6S10

```
| makeresults count=2
| streamstats count as count
| eval time=case(count=2,relative_time(now(),"+2d"),count=1,now())
| makecontinuous time span=15m
| eval _time=time
| eval HourOfDay=strftime(_time, "%H")
| eval BucketMinuteOfHour=strftime(_time, "%M")
| eval DayOfWeek=strftime(_time, "%A") …
```

Results Example

BucketMinuteOfHour	DayOfWeek	HourOfDay	_time	count	time
45	Friday	17	2018-07-20 17:45:00		1532108700
59	Friday	17	2018-07-20 17:59:41	1	1532109581
15	Friday	18	2018-07-20 18:15:00		1532110500
30	Friday	18	2018-07-20 18:30:00		1532111400

36. How could `makeresults` be useful?

To provide high and low outlier boundaries that extend into the future, against which you can compare new events.

Lab Exercise 7 – Estimation and Prediction

Description

Fit a linear regression to estimate and make predictions.

Steps

Task 1: Fit a linear regression to predict horsepower from weight.

1. Load the automobile dataset (`auto-mpg.csv`).
2. `fit` a linear regression predicting `hp` based on a field of your choice.
   Remember, it must be numeric.
3. **Save** it into a new model named **hp**
4. Save your results as a report and name it **L7S1**

```
| inputlookup auto-mpg.csv
| sample partitions=2 seed=123456
| search partition_number=0
| fit LinearRegression hp from weight into hp
| table hp predicted(hp)
```

Results Example

hp	predicted(hp)
150	122.412934266
150	122.2894528
140	122.948020619
220	160.198262872
215	158.469522347

Task 2: Add more features to your linear regression models and reflect on what you have learned.

5. Each time you add new features, adjust your seed such that you have a different test portion and avoid overfitting.
6. How much do they affect your R^2 value?
7. How can the commands you learned earlier help make more informative decisions about which features to use, before you actually fit the model and test its goodness of fit?
8. Save your results as a report and name it **L7S2**
9. Here's an example to get started:

```
| inputlookup auto-mpg.csv
| sample partitions=2 seed=123456
| search partition_number=0
| fit LinearRegression hp from displ cyl into hp
| `regressionstatistics(hp,predicted(hp))`
```

Task 3: Model web traffic growth.

10. Examine `access_combined` from six weeks ago until two weeks ago.
11. Make a `timechart` counting the number of events at intervals of one day.
12. `fit` a linear regression model to predict web traffic for the next two weeks and name it **web**
13. Save your results as a report and name it **L7S3**

```
sourcetype=access_combined earliest=-6w@w latest=-2w@w
| timechart count span=1d
| fit LinearRegression count from _time as prediction into web
```

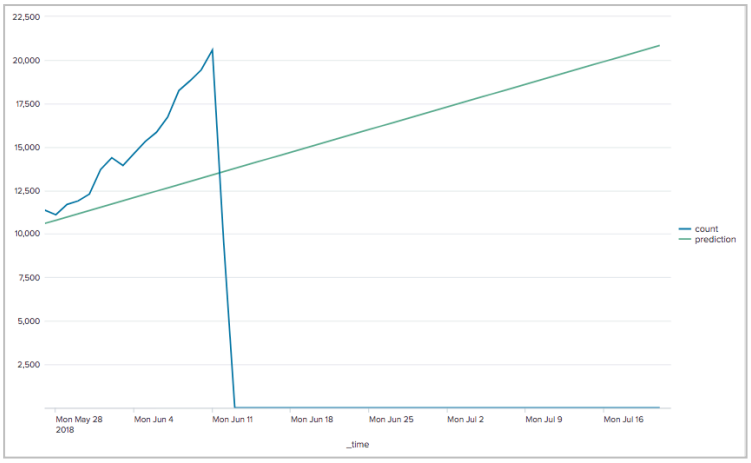
Results Example

What thoughts do you have about assuming linearity of this data?

14. `apply` your linear regression to predict web traffic from two weeks ago to six weeks from now.
15. Save your results as a report and name it **L7S4**

```
sourcetype=access_combined earliest=-2w@w latest=+6w@w
| timechart count span=1d
| apply web
```

Results Example



Task 4: Use eval to create a quadratic component of time, fit two models, and compare.

16. Examine `access_combined` between 6 and 2 weeks ago.
17. Create a `timechart` with a span of 1 day.
18. Use eval to create `t2 as _time * _time`
19. `fit` another linear regression count from `_time t2` into `quad_prediction`
20. Remove the `t2` field.
21. Save your results as a report and name it **L7S5**

```
sourcetype=access_combined earliest=-6w@w latest=-2w@w
| timechart count span=1d
| eval t2 = _time * _time
| fit LinearRegression count from _time t2 into quad_prediction
| fields - t2
```

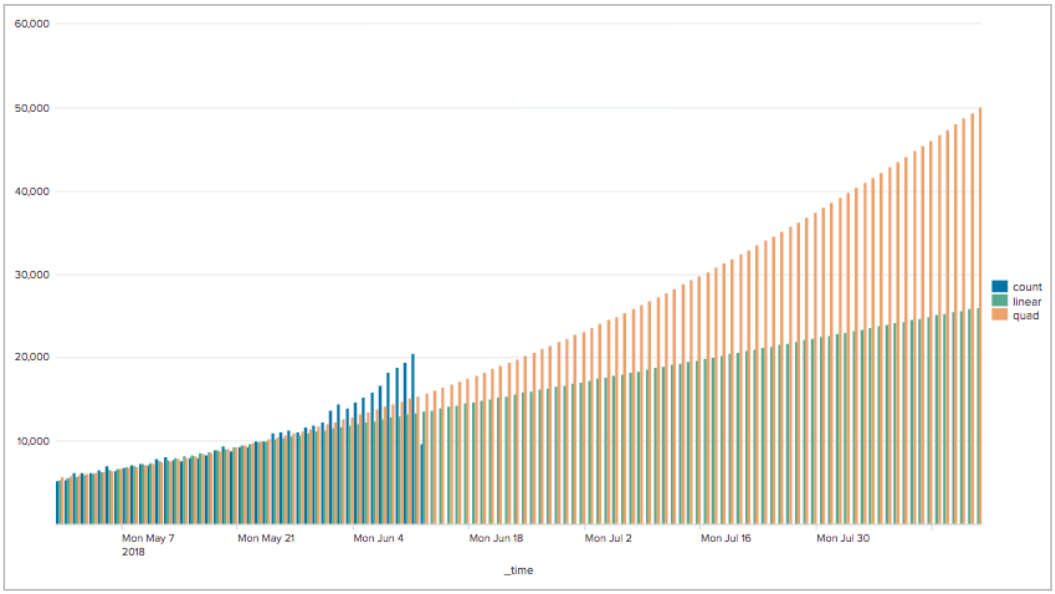
Results Example

Task 5: Compare two models:

22. Examine `access_combined` between 6 weeks ago and 10 weeks from now.
23. Create a `timechart` with a span of 1 day.
24. Use `eval` to create `t2` as `_time * _time`
25. `apply web` as `linear`
26. `apply` quad_prediction as `quad`
27. Remove the `t2` field.
28. Visualize as **Column** chart with `quad` as overlay.
29. Save your results as a report and name it **L7S6**

```
sourcetype=access_combined earliest=-6w@w latest=+10w@w
| timechart count span=1d
| eval t2 = _time * _time
| apply web as linear
| apply quad_prediction as quad
| fields - t2
```

Results Example



Task 6: Hone a prediction

30. **Save** this search (over the **Last 7 days**) as **L7S7**

```
sourcetype=access_combined
| timechart span=1h sum(price) as total_sales
| predict total_sales
```

31. Search across the previous 3 months and experiment with various algorithm options.
32. When you have a combination of options you find interesting, save it as **L7S8** with the new options.

Algorithm option	Algorithm type	Description
LL	Local level	A univariate model with no trends and no seasonality. Requires a minimum of 2 data points. The LL algorithm is the simplest algorithm and computes the levels of the time series. For example, each new state equals the previous state, plus the Gaussian noise.
LLT	Local level trend	A univariate model with trend, but no seasonality. Requires a minimum of 3 data points.
LLP	Seasonal local level	A univariate model with seasonality. The number of data points must be at least twice the number of periods. The LLP algorithm takes into account the cyclical regularity of the data, if it exists. If you know the number of periods, specify the `period` option. If you do not set the `period`, this algorithm tries to calculate it. LLP returns an error message if the data is not periodic.
LLP5	Combines LLT and LLP models for its prediction.	If the time series is periodic, LLP5 computes two predictions, one using LLT and the other using LLP. The algorithm then takes a weighted average of the two values and outputs that as the prediction. The confidence interval is also based on a weighted average of the variances of LLT and LLP.
LLB	Bivariate local level	A bivariate model with no trends and no seasonality. Requires a minimum of 2 data points. LLB uses one set of data to make predictions for another. For example, assume it uses dataset Y to make predictions for dataset X. If `holdback=10`, LLB takes the last 10 data points of Y to make predictions for the last 10 data points of X.
BiLL	Bivariate local level	A bivariate model that predicts both time series simultaneously. The covariance of the two series is taken into account.

33. What conclusions can you draw?

    More data is needed to account for possible seasonality

Lab Exercise 8 – Classification

Description

Classify events and evaluate your classifications.

Steps

Task 1: Create a field-selecting model.

Create a `FieldSelector` model to determine a useful field for classification.

1. Load the phishing data (`phishing.csv`)
2. Fit the `FieldSelector` model as categorical, percentile, param 0.05
3. Predict from all fields.
4. Manually note the field(s) selected.
5. Save your results as a report and name it **L8S1**

```
| inputlookup phishing.csv
| fit FieldSelector
type=categorical
mode=percentile
param=5
Result from *
| table fs_*
```

6. Field(s) selected: _____

   SSLfinal_State

Task 2: Fit a model to classify novel data

Now that you know what might be a good predictor field, fit a logistic regression, SVM, or other classification model to classify future data.

7. Load the phishing data (`phishing.csv`)
8. Split your data into training and testing sets. (Recall Lab Exercise 2.)
9. `fit` the model to the training set from the field selected and name it **SVM_classifier**
10. Save your results as a report and name it **L8S2**

```
| inputlookup phishing.csv
| sample partitions=2 seed=1234
| search partition_number=0
| fit SVM Result from SSLfinal_State into SVM_classifier
```

Results Example

Abnormal_URL	DNSRecord	Domain_registration_length	Favicon	Google_Index	HTTPS_token	Iframe	Links_in_tags	Links_pointing_to_page	Page_Rank	Prefix_Suffix	Redirect	Request_URL	Result	RightClick	SFH	SSLfinal_State
1	-1	-1	1	1	-1	1	-1	1	-1	-1	0	1	-1	1	-1	1
-1	-1	-1	1	1	-1	1	-1	0	-1	-1	0	1	-1	1	-1	-1
1	-1	1	1	1	-1	1	-1	0	-1	-1	0	-1	-1	1	-1	-1
1	-1	-1	1	1	1	1	1	0	-1	-1	0	1	-1	1	-1	1
-1	1	1	1	1	1	1	0	-1	1	-1	0	-1	1	1	-1	1

11. `apply` the model to the test data.
12. Save your results as a report and name it **L8S3**

```
| inputlookup phishing.csv
| sample partitions=2 seed=1234
| search partition_number=1
| apply SVM_classifier as prediction
| `classificationstatistics(prediction, Result)`
```

Results Example

class	accuracy	precision	recall	f1	count
Weighted Average	0.89	0.89	0.89	0.89	5545

13. Try changing to one of the other fields that might be a good predictor.
14. Make sure you change your `seed` to get a different train-test split.
15. Save your results as a report and name it **L8S4**
16. How does it affect the precision and recall of the classification?
    - Precision = true_positives / (true_positives + false_positives)
    - Recall = true_positives / (true_positives + false_negatives)

    Example answer: improves recall at the expense of precision

Task 3: Train a classifier to predict customer churn at Buttercup Games (BCG).

17. Review and load the BCG churn dataset. You will be predicting the inactive field's values.
18. Split the data into training and testing sets (remember to use a seed).
19. Use `FieldSelector` to obtain a subset of features to use.
    NOTE: FieldSelector changes values that are categorical into their respective dummy variables. For example, if sex=male and sex=female are both features, when we fit the model, we only use sex.
20. Train the classifier on the training data and save the model.
21. Apply the model on the testing data and evaluate its performance with the `` `classificationstatistics` `` macro and the `` `confusionmatrix` `` macro.
22. Change the seed value and add or remove a feature from your model, seeing how it affects your model's performance.
23. Save your model, click Continue to edit, and name the report **L8S5**.

```
| inputlookup bcg_churn.csv
| sample partitions=3 seed=42
| where partition_number < 2
| fit FieldSelector mode=k_best param=5 type=categorical inactive from *
| fields fs_*
```

Results Example

fs_dedicated_server=no	fs_dedicated_server=yes	fs_morning_use	fs_social=yes	fs_support_tickets
1.0	0.0	265.1	1.0	1.0
1.0	0.0	161.6	1.0	1.0
1.0	0.0	243.4	0.0	0.0
0.0	1.0	299.4	0.0	2.0

```
| inputlookup bcg_churn.csv
| sample partitions=3 seed=42
| where partition_number < 2
| fit GaussianNB inactive from dedicated_server morning_use social support_tickets into
GaussianNB_classifier
```

```
| inputlookup bcg_churn.csv
| sample partitions=3 seed=42
| where partition_number=2
| apply GaussianNB_classifier as prediction
| `classificationstatistics(inactive, prediction)`
```

Results Example

class	accuracy	precision	recall	f1	count
Weighted Average	0.86	0.85	0.86	0.85	1133

OPTIONAL TASK

Task 4: Train a classifier to predict SMS spam using TFIDF features.

24. Review and load the Spam Text dataset.
25. Split the data into training and testing sets (remember to use a seed).
26. Use `fit` with `TFIDF` to extract features from the text data.
27. Use `fit` with a classifier to train and save a model.
28. Name your model **TFIDF_classifier**
29. Evaluate your model's performance with the `confusionmatrix` macro.
30. Try adding `stop_words=english` to the TFIDF's options and reassess performance.
31. Try modifying the classifier's parameters to improve the model.
32. If you do improve your model, try changing your seed, retraining and testing to see if your model does better again with a "new" set of training data.
33. Save your results as a report, click Continue to edit and name it **L8S6**

```
| inputlookup spam.csv
| sample partitions=4 seed=42
| where partition_number < 2
| fit TFIDF text stop_words=english into vectorizer
| fit RandomForestClassifier result from text_* into TFIDF_classifier


| inputlookup spam.csv
| sample partitions=4 seed=42
| where partition_number >= 2
| apply vectorizer
| apply TFIDF_classifier as prediction
| `confusionmatrix(result, prediction)`
```

Results Example