



CIS3200 Term Project Tutorial



Authors: Alan Liu

Instructor: [Jongwook Woo](#)

Date: 05/19/2019

Lab Tutorial

Alan Liu (aliu38@calstatela.edu)

05/19/2019

Predicting BMI from Food Availability

Objectives

This tutorial will teach you how to create the best-performing BMI prediction model of the ones tested in the full project. In this lab, you will learn how to:

- Get data manually from USDA ERS website
- Load data into Microsoft Azure ML Studio
- Transform data
- Train model
- Load data into Elasticsearch
- Visualization

Platform Specifications

- Microsoft Azure Machine Learning Studio (Free tier)
 - Maximum Nodes per Experiment: 100
 - Maximum Experiment Duration: 1 hour
 - Node Execution Performance: Single Node
 - Maximum Storage Space: 10GB
- Elasticsearch Service (Deployment consisting of co-located master/data/ingest node, kibana, ml)
 - Instance/Configuration: 'gcp.data.highio.1'
 - CPU Cores: 12 cores
 - Memory: 78GB (1GB provisioned)
 - Storage: 3000GB (30GB provisioned)
 - Instance/Configuration: 'gcp.kibana.1'
 - CPU Cores: 22 cores
 - Memory: 140GB (1GB provisioned)
 - Storage: N/A
 - Instance/Configuration: 'gcp.ml.1'
 - CPU Cores: 64 cores
 - Memory: 270GB (1GB provisioned)
 - Storage: N/A






Step 1: Get data manually from USDA ERS website

The United States Department of Agriculture's Economic Research Service conducted a Food Acquisition and Purchase Survey across the nation and the participant's responses are available for download at the website:

<https://www.ers.usda.gov/data-products/foodaps-national-household-food-acquisition-and-purchase-survey/>

You can optionally also download the Documentation at the same link, which contains the codebook files explaining the columns and their meanings in each dataset.

1. Visit website and download csv dataset from link.

Data Set	Last Updated
Public-Use Data Files and Codebooks	
SAS data files 	2/21/2017
Stata data files 	2/21/2017
CSV data files 	2/21/2017
Documentation 	4/26/2017
Imputed Quantities for FAH and FAFH 	1/31/2018

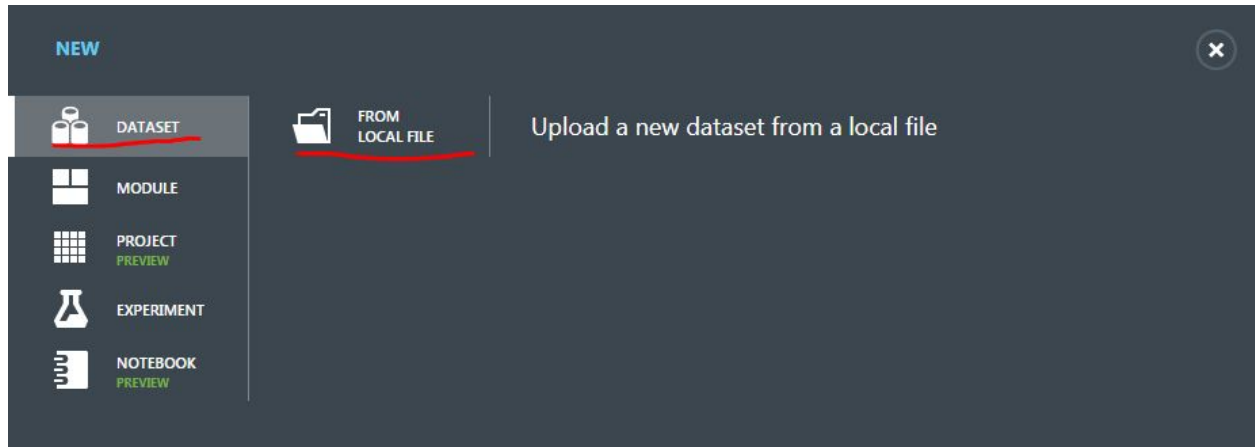
2. Unzip the files into a folder.

Step 2: Load data into Microsoft Azure ML Studio

We need to upload the data into our Azure ML as a dataset before we can use them as modules.

1. Login to your Azure ML Studio account using your school email: <https://studio.azureml.net>

2. On the bottom left corner, click **New** and click on **Dataset** on the left sidebar that pops up, and upload your csv files by clicking **From Local File**:



3. Navigate to folder where you extracted the dataset, and select **faps_access_puf.csv** and enter the name for the data set "**faps_access_puf**" and click the checkmark.

Upload a new dataset

SELECT THE DATA TO UPLOAD:

faps_access_puf.csv

☐ This is the new version of an existing dataset

ENTER A NAME FOR THE NEW DATASET:

faps_access_puf

SELECT A TYPE FOR THE NEW DATASET:

Generic CSV File with a header (.csv) ▼

PROVIDE AN OPTIONAL DESCRIPTION:



4. Repeat above step 3 for the csv files:

faps_fafhevent_puf.csv

faps_fafhitem_puf.csv

faps_individual_puf.csv

Step 3: Transforming the data

Before using the data for training the model, we have to clean the data and make sure we only use the columns we need. We also have to make sure the metadata for the columns are correct.

1. **Make a new experiment** by clicking on the **New** button and **Blank Experiment**. Name it something you will remember.
2. From your experiment items sidebar, drag and drop the **faps_access_puf** into the experiment.
3. Drag and drop a **Select Columns in Dataset** module into the project.
4. Connect the output port from the **faps_access_puf** module to the input port of the **Select Columns in Dataset** module.
5. Click on the **Select Columns in Dataset** module and in the properties panel, click **Launch Column Selector**.
6. Select the columns **By Name** from the available columns:
hhnum, snap1, snap2, snap3, snap4, snap5, snap6, snap8,
ss1, ss2, ss3, ss4, ss5, ss6, ss7, ss8,
sm1, sm2, sm3, sm4, sm5, sm6, sm7, sm8,
co1, co2, co3, co4, co5, co6, co7, co8,
cs1, cs2, cs3, cs4, cs5, cs6, cs7, cs8,
mlg1, mlg2, mlg3, mlg4, mlg5, mlg6, mlg7, mlg8,
dist_co, dist_cs, dist_mlg,
ff1, ff2, ff3, ff4, ff5, ff6, ff7, ff8,
nonff1, nonff2, nonff3, nonff4, nonff5, nonff6, nonff7, nonff8,
nearff_dist, nearnonff_dist

7. Drag and drop your **faps_individual_puf** dataset into the project, and another **Select Columns in Dataset** module, and connect them together like in step 4.

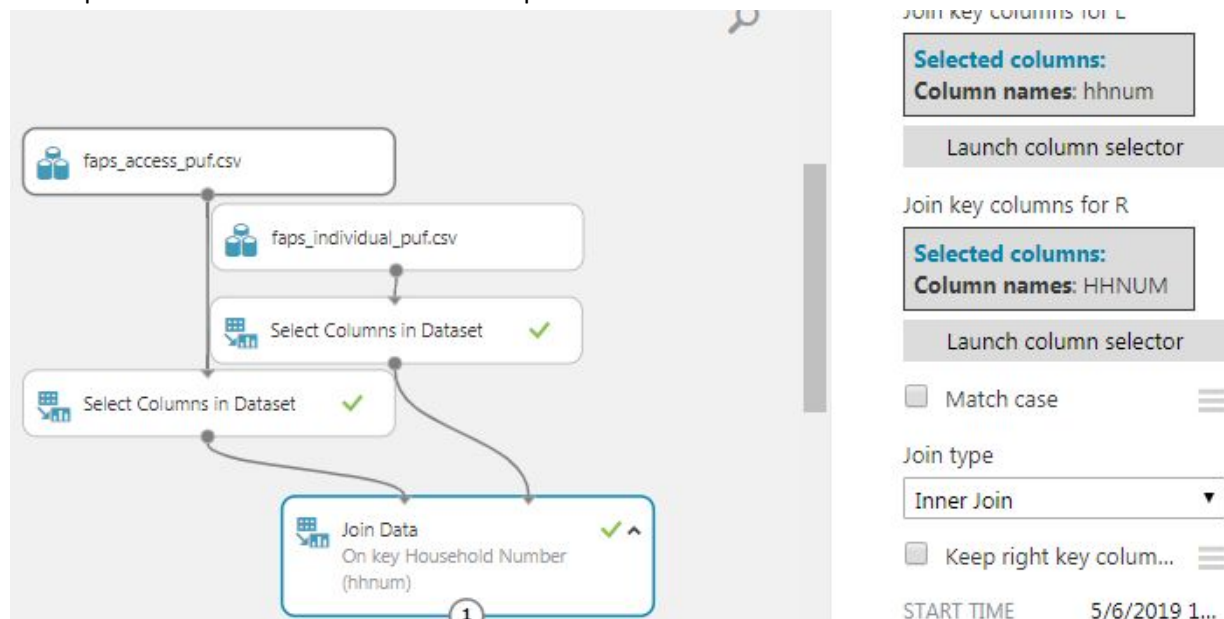
8. Click on the new **Select Columns in Dataset** module and **Launch Column Selector**.

9. Select the columns **By Name** from the available columns:
HHNUM, PNUM, NDINNERSOUT, HEALTHSTATUS, BMICAT, BMI

10. Add a **Join Data** module to the experiment, and connect the output ports of both **Select Columns in Dataset** modules to the input ports of **Join Data**. Make sure the one for **faps_access_puf** is on the **left input** and the one for **faps_individual_puf** is on the **right input**.

11. In the **Join Data** property panel, select **hhnum** for the **Join key columns for L**, and select **HHNUM** for the **Join key columns for R**. Make sure **Match case** and **Keep right key column** is **unchecked**. Set the **Join type** to **Inner Join**.

The experiment should look like this at this point:



12. Add the module **Execute Python Script** to the experiment. Connect the **Join Data** output to this module's input and use this code from Github:

<https://github.com/shozonu/usda-foodaps-bmi-model/blob/master/py/prep1.py>

```
import pandas as pd

def azureml_main(dataframe1 = None, dataframe2 = None):
    # For each row
    for index, row in dataframe1.iterrows():
        hnum = dataframe1['hhnum'].iat[index]
        pnum = dataframe1['PNUM'].iat[index]

        # Append create unique pid by appending pnum to end of hhnum
        pid = int("{}{}".format(hnum, pnum), 10)
        dataframe1["hhnum"].iat[index] = pid

    # Rename columns
    dataframe1.rename(columns={"hhnum" : "pid"}, inplace = True)
    dataframe1.rename(columns={"NDINNERSOUT" : "ndinnersout"}, inplace = True)
    dataframe1.rename(columns={"HEALTHSTATUS" : "healthstatus"}, inplace = True)
    dataframe1.rename(columns={"BMICAT" : "bmicat"}, inplace = True)
    dataframe1.rename(columns={"BMI" : "bmi"}, inplace = True)

    # Drop pnum column
    dataframe1.drop(["PNUM"], axis=1, inplace = True)

    return dataframe1
```

This Python script will create a unique id from the household number and person number field. It will also rename the upper case columns, the new id column, and remove the extra pnum column.

13. Add a **Edit Metadata** module. Select the **pid** column. Leave the **Data type** and **Categorical** dropdown menu as **unchanged**. Change the **fields** dropdown to **Clear feature**.

14. Add a **Apply SQL Transformation** module. Connect the previous output to this input. We need to remove rows that contain the error flags specified in the official data codebook, using the SQL script from Github: <https://github.com/shozonu/usda-foodaps-bmi-model/blob/master/sql/tran1.sql>

```
-- Filter out rows with these columns
-- that contain error flags.
SELECT * FROM t1
WHERE bmi > 0
AND bmi != 'V'
AND bmicat != 'V'
AND bmicat != 'E'
AND healthstatus != 'D'
AND healthstatus != 'R'
AND ndinnersout != 'D'
AND ndinnersout != 'R';
```

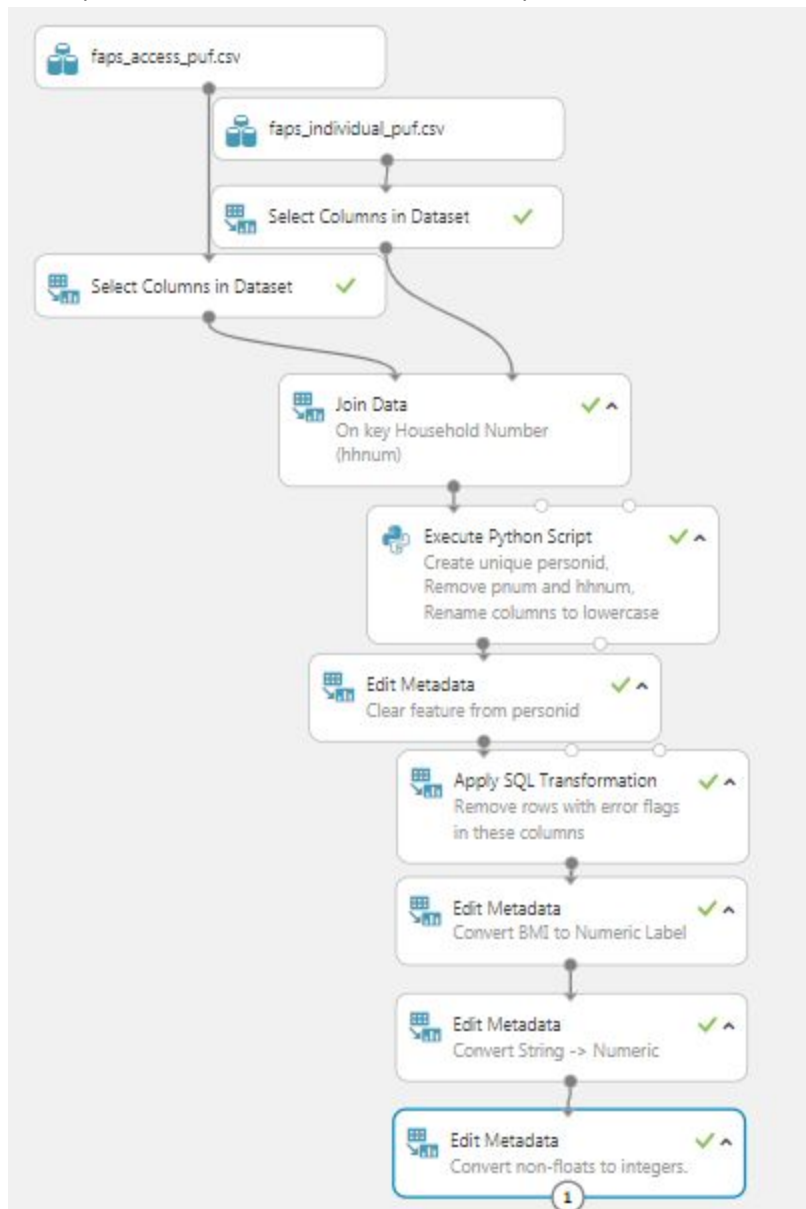
15. Connect the previous output to the input of a new **Edit Metadata** module. Select the **bmi** column, and **uncheck allow duplicates**. Make the data type **Floating point**. Set it to **Make non-categorical**. Set the fields to **Label**.

16. Connect the previous output to the input of a new **Edit Metadata** module. Select the columns: **bmicat, ndinnersout, healthstatus**. **Uncheck allow duplicates**. Make the data type **Integer**. Set it to **Make categorical**. Set the fields to **Features**.

17. Connect the previous output to the input of a new **Edit Metadata** module. Select the columns: **snap1, snap2, snap3, snap4, snap5, snap6, snap8, ss1, ss2, ss3, ss4, ss5, ss6, ss7, ss8, sm1, sm2, sm3, sm4, sm5, sm6, sm7, sm8, co1, co2, co3, co4, co5, co6, co7, co8, cs1, cs2, cs3, cs4, cs5, cs6, cs7, cs8, mlg1, mlg2, mlg3, mlg4, mlg5, mlg6, mlg7, mlg8, ff1, ff2, ff3, ff4, ff5, ff6, ff7, ff8, nonff1, nonff2, nonff3, nonff4, nonff5, nonff6, nonff7, nonff8**

Uncheck allow duplicates. Make the data type **Integer**. Set it to **Make non-categorical**. Set the fields to **Unchanged**.

The experiment should look like this at this point:



Step 4: Train model

Now that we're done cleaning the data, we can move onto the setting up the training portion for predicting the BMI.

1. First we want to remove the outliers so that we can improve overall model performance. Add a Clip Values module to the experiment.
2. In the properties panel, set the **Set of thresholds** to **ClipPeaksAndSubpeaks**. Set **Threshold** to **Percentile**. Set the upper percentile to **90** and the lower percentile to **10**. Set **substitute value for peak** to **Missing**. Set **substitute value for subpeak** to **Missing**. Launch the column selector and select **bmi**. Check the **Overwrite flag** box and **uncheck** the **Add Indicator column** box.

It should look like this:

Properties Project

▲ Clip Values

Set of thresholds

ClipPeaksAndSubpeaks ▼

Threshold

Percentile ▼

Percentile number of u... ≡

90

Percentile number of lo... ≡

10

Substitute value for pe... ≡

Missing ▼

Substitute value for su... ≡

Missing ▼

List of columns

Selected columns:
Column names: bmi

Launch column selector

☒ Overwrite flag ≡

☐ Add indicator colu... ≡

3. Add a **Clean Missing Data** module to the experiment and connect the last output to this input. Make sure that **All columns** are selected. Leave the minimum and maximum value ratio at 0 and 1. Make sure cleaning mode is set to **Remove entire row**.

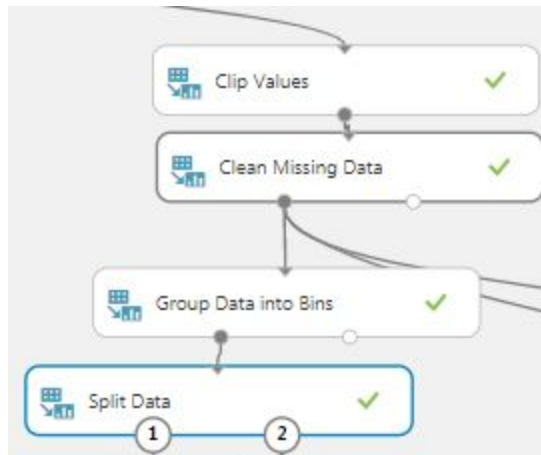
4. Connect the last module to a new **Group Data into Bins** module. In the properties panel, set the **Binning mode** to **Quantiles**. Set **Number of bins** to **10**. Set **Quantile normalization** to **percent**. Select the following columns:

snap1, snap2, snap3, snap4, snap5, snap6, snap8,
ss1, ss2, ss3, ss4, ss5, ss6, ss7, ss8,
sm1, sm2, sm3, sm4, sm5, sm6, sm7, sm8,
co1, co2, co3, co4, co5, co6, co7, co8,
cs1, cs2, cs3, cs4, cs5, cs6, cs7, cs8,
mlg1, mlg2, mlg3, mlg4, mlg5, mlg6, mlg7, mlg8,
ff1, ff2, ff3, ff4, ff5, ff6, ff7, ff8,
nonff1, nonff2, nonff3, nonff4, nonff5, nonff6, nonff7, nonff8

Set the **Output mode** to **Inplace**, and make sure the **Tag columns as categorical** is unchecked.

5. Connect the previous output to the input of a new **Split Data** module. Set the **Splitting mode** to **Split Rows**. Set the **Fraction of rows in the first output** to **0.66**. Make sure the **Randomized split** box is **checked**. Set the **Random seed** to **4321**. Set the **Stratified split** to **False**.

Your experiment should look like this at this point:



6. Add a **Bayesian Linear Regression** module.

7. Add a **Tune Model Hyperparameters** module and connect the **Bayesian Linear Regression** to the untrained model input.

8. In the **Tune Model Hyperparameters** properties panel, set it to **Random sweep** with a maximum number of runs of **8**. Set the **Random seed** to **321**. Select the **bmi** column. Set the **Metric for measuring performance for regression** to **Coefficient of determination**.

It should look like this:

▲ Tune Model Hyperparameters

Specify parameter sweeping m...

Random sweep ▼

Maximum number of runs... ≡

8

Random seed ≡

321

Label column

Selected columns:
Column names: bmi

Launch column selector

Metric for measuring perf... ≡

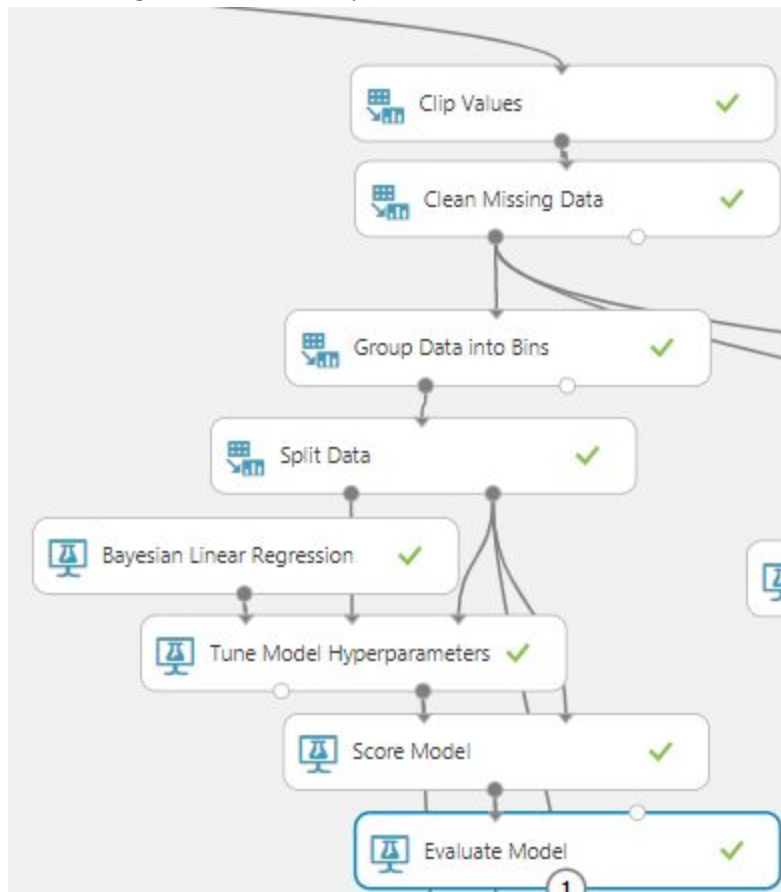
Accuracy ▼

Metric for measuring perf... ≡

Coefficient of determinator ▼

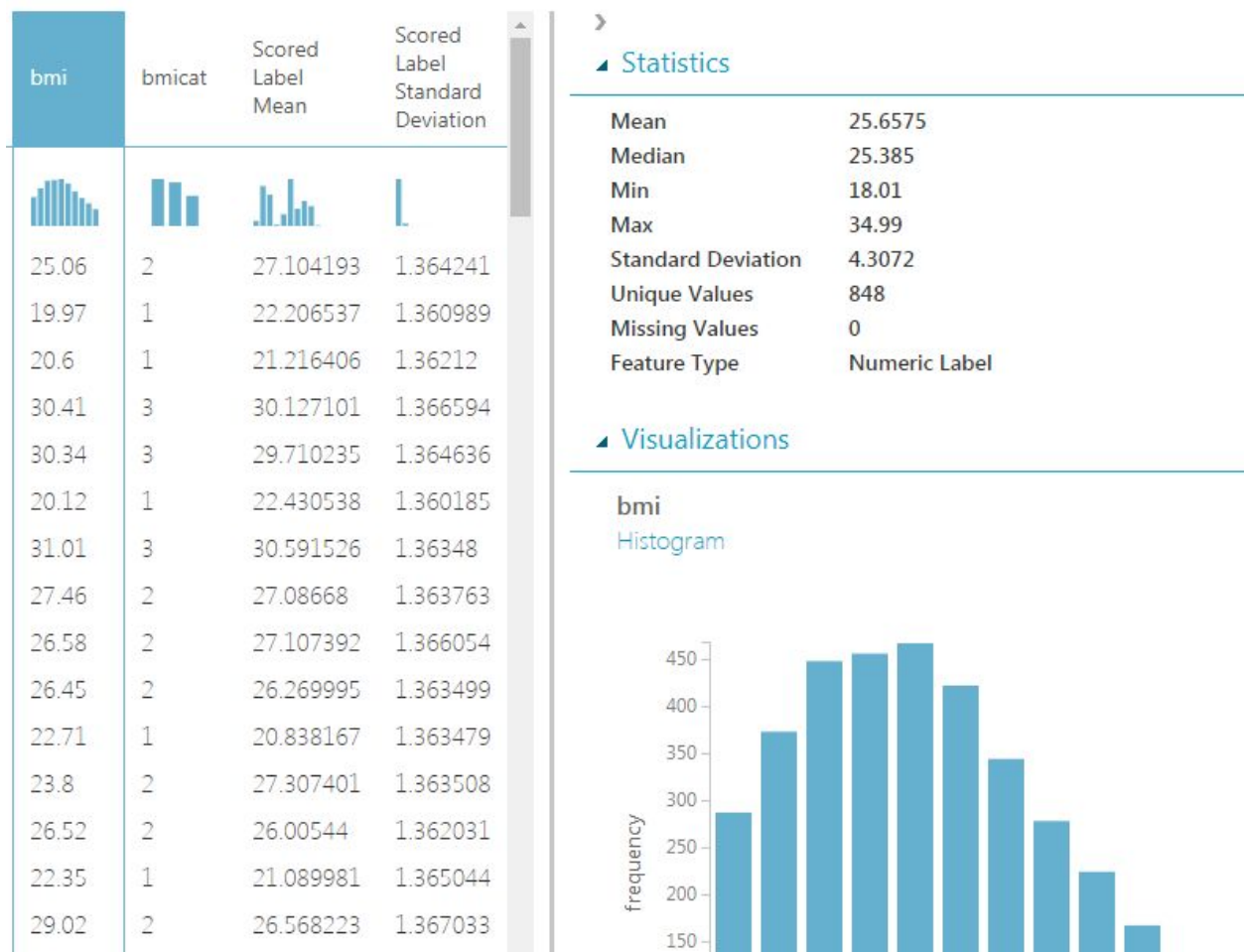
9. Connect the first output of the **Split Data** to the second input of **Tune Model Hyperparameters**.
10. Connect the second output of **Tune Model Hyperparameters** to the first input of a new **Score Model** module.
11. Connect the third output of the **Split Data** module to the third input of **Tune Model Hyperparameters** and the second input of **Score Model**.
12. Connect the output of **Score Model** to the first input of a new **Evaluate Model** module.

The training section of the experiment should look like this:



13. Save and run the experiment.

14. Visualize the output of **Score Model** to see the predicted BMI.



We can see that the predicted BMI rows are fairly close to the true BMI most of the time.

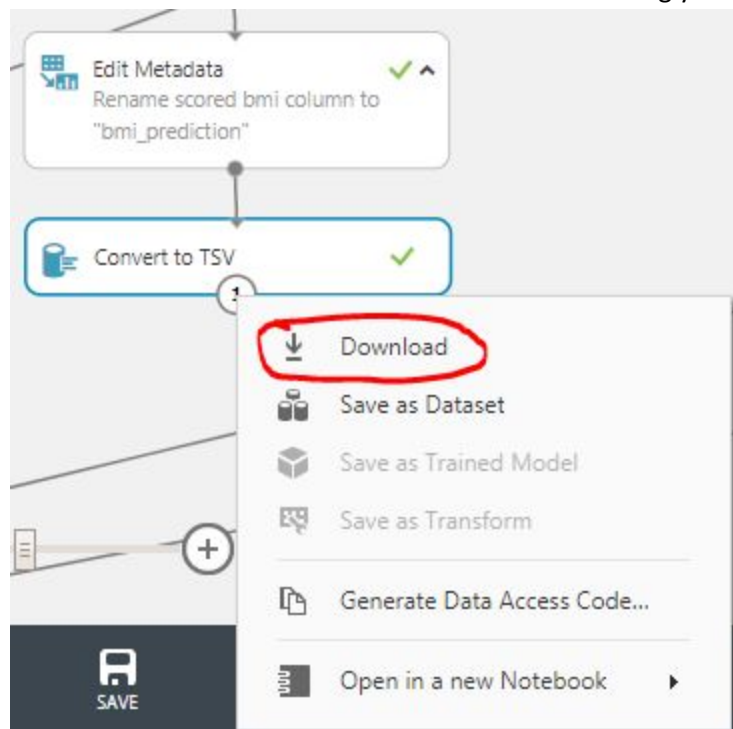
15. Visualize the output of **Evaluate Model** to see the performance of the model.

rows	columns						
		Negative Log Likelihood	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
1	6	11239.575861	2.039555	2.737708	0.566542	0.404126	0.595874

Step 3: Visualization

To better visualize the difference between the true BMI and the predicted BMI, we will export the scored dataset to Elasticsearch and visualize it in a graph.

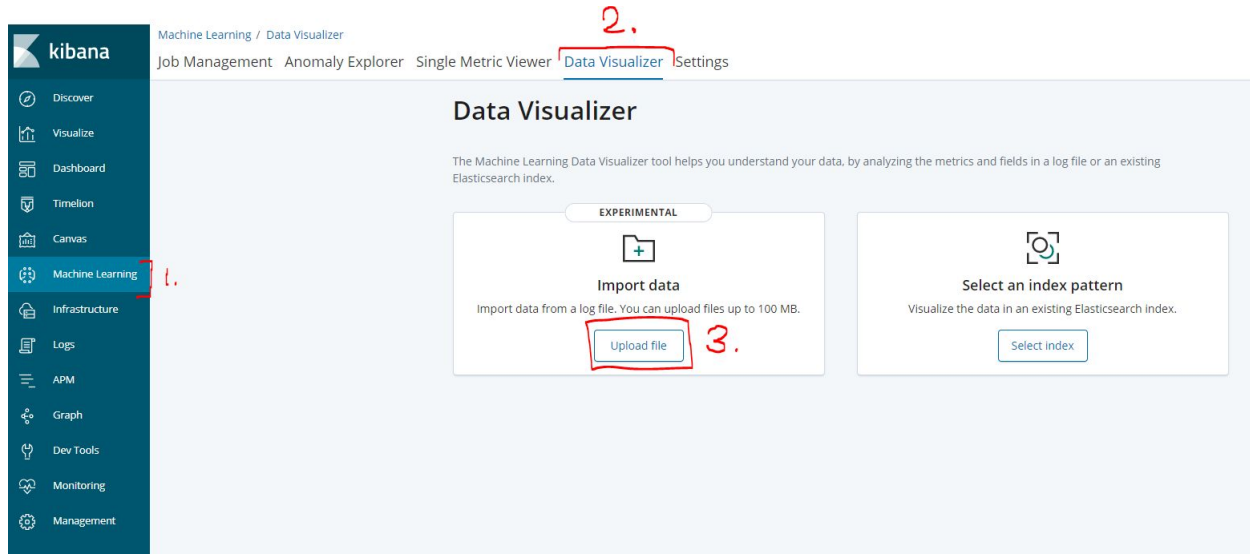
1. Connect the output of **Score Model** to the input of a new **Edit Metadata** module.
2. In the **Edit Metadata** properties, select the **Scored Label Mean** column. Leave everything unchanged except for **New column names**, set it to **bmi_prediction**.
3. Connect the output of the last **Edit Metadata** to the input of a new **Convert to TSV** module.
4. Save and run the experiment.
5. Right-click on the output of **Convert to TSV** and select **Download** to download the scored dataset. Make sure to rename the downloaded file to something you can remember.



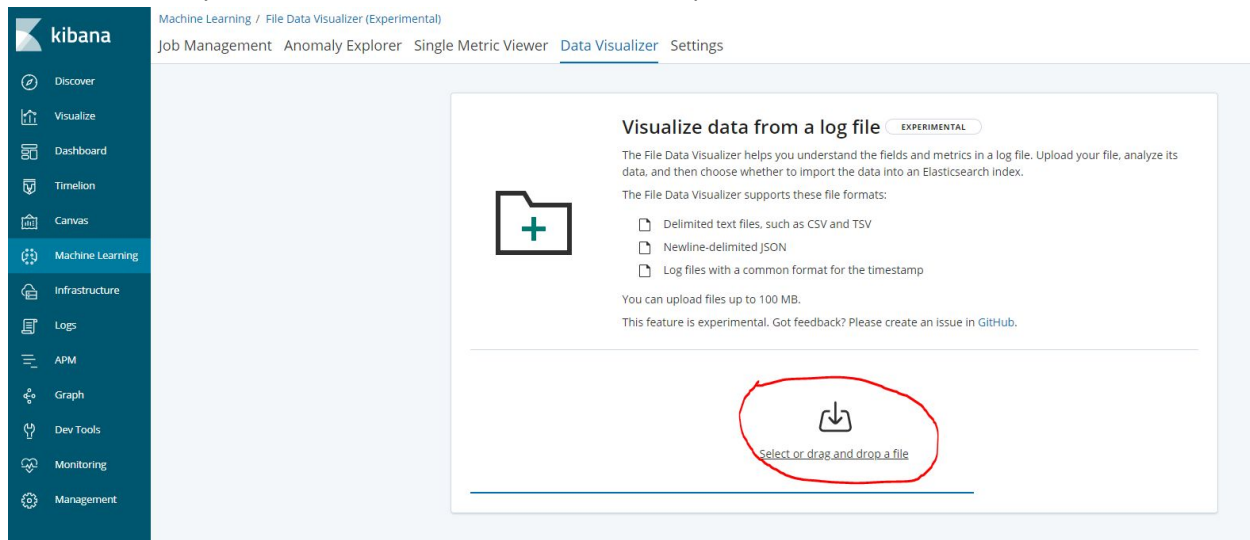
6. Log into your Elasticsearch Service dashboard at <https://cloud.elastic.co>

7. Under your deployments, click **Launch Kibana**, and login to your deployment's Kibana with your credentials.

8. In the **Machine Learning** sidebar, click on the **Data Visualizer** tab and click **Upload File**.



9. Select the file you've downloaded from Azure ML and upload it.



10. It will only read the first 1000 lines at this point, click the **Import** button on the bottom left.

The screenshot shows the Kibana Machine Learning interface. The left sidebar contains navigation links: Discover, Visualize, Dashboard, Timeline, Canvas, Machine Learning (selected), Infrastructure, Logs, APM, Graph, Dev Tools, Monitoring, and Management. The main panel displays the 'Data Visualizer' for a file named 'dist_ss_dist_ss'. It shows the first 1000 lines of the file, a summary of the data (1000 lines analyzed, delimited format, no header row), and file statistics for columns #pid, #snap1, #snap2, and #snap3. The 'Import' button is highlighted in the bottom left corner.

11. Name the index **bmi_prediction** and click **Import**.

The screenshot shows the 'Import' dialog box in Kibana. The 'Index name' field is filled with 'bmi_prediction'. The 'Create index pattern' checkbox is checked. The 'Index pattern name' field is also filled with 'bmi_prediction'. The 'Import' button is highlighted in the bottom left corner.

Wait for the upload to finish, it may take a while. If it freezes or hangs, try again and not switch away from the browser window or tab.

The screenshot shows the 'Import' progress bar and summary in Kibana. The progress bar shows four steps: File processed, Index created, Data uploaded, and Index pattern created, all of which are completed. Below the progress bar, a summary table shows the import details: Index (pid_access_bmi), Index pattern (pid_access_bmi), and Documents ingested (13073). At the bottom, there are four buttons: View index in Discover, Open in Data Visualizer, Index Management, and Index Pattern Management.

12. Verify that the data was successfully imported by going to **Discover** and selecting the **bmi_prediction** index.

The screenshot shows the Kibana Discover interface. The left sidebar contains navigation links: Discover, Visualize, Dashboard, Timelion, Canvas, Machine Learning, Infrastructure, Logs, APM, Graph, Dev Tools, Monitoring, and Management. The 'Discover' tab is active. The main area shows a search bar with the text '>_ Search... (e.g. status:200 AND extension:PHP)'. Below the search bar, the 'bmi_prediction' index is selected and highlighted with a red underline. The 'Selected fields' section lists: ? _source, t _id, t _index, # _score, t _type, # bmi, # bmi_prediction, # bmicat, # co1, # co2, # co3, # co4, and # co5. The 'Available fields' section lists: pid, ss6, co5, mlg4, ff6, pid, ss7, co7, mlg6, ff8, pid, ss6, co6, mlg5, ff7, pid, ss6, and co6. The right side of the interface displays a table of data for the selected index, showing fields like pid, ss6, co5, mlg4, ff6, pid, ss7, co7, mlg6, ff8, pid, ss6, co6, mlg5, ff7, pid, ss6, and co6.

13. Go to **Visualize** and create a new visualization

The screenshot shows the Kibana Visualize interface. The left sidebar contains navigation links: Discover, Visualize, Dashboard, Timelion, Canvas, Machine Learning, Infrastructure, Logs, APM, Graph, Dev Tools, Monitoring, and Management. The 'Visualize' tab is active and highlighted with a red underline. The main area shows a search bar with the text 'Search...'. Below the search bar, there is a table of visualization types. The table has two columns: 'Type' and 'Visualize'. The rows are: Title 1, Bar Example, Map Example, mid2BarChart, mid2PieChart, Pie Example, and 0 items selected. The 'Visualize' column for each row shows the corresponding visualization type: Vertical Bar, Coordinate Map, Vertical Bar, Pie, and Pie. A red circle highlights the '+' button in the top right corner of the search bar.

14. For the visualization type, select **Area** and select the **bmi_prediction** index.

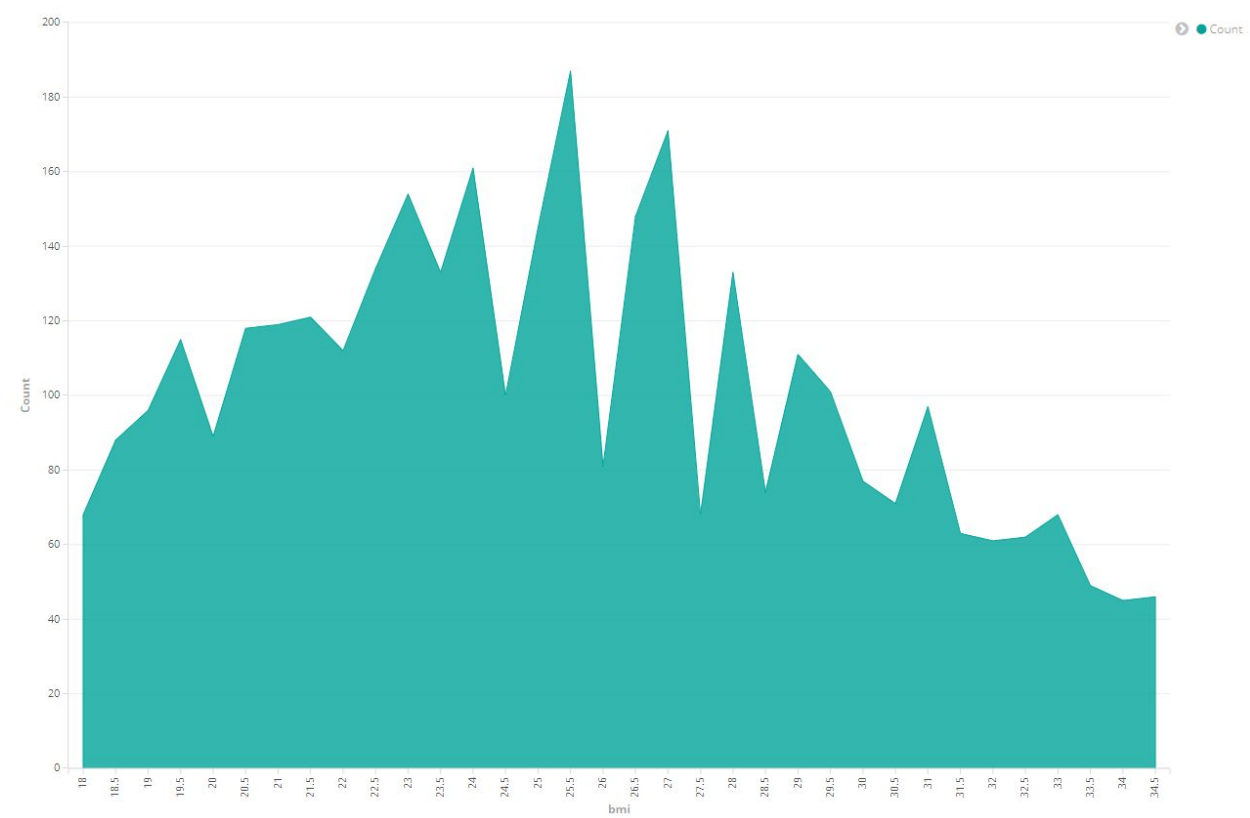
15. Make sure the **Y-Axis** is set to Count. Add a bucket for **X-Axis** and set the aggregation to **Histogram** and field **bmi**. Set the minimum interval to **0.5**. Click the play button to update the graph.

The image displays two screenshots of the 'bmi_prediction' dashboard configuration interface, showing the 'Metrics & Axes' tab.

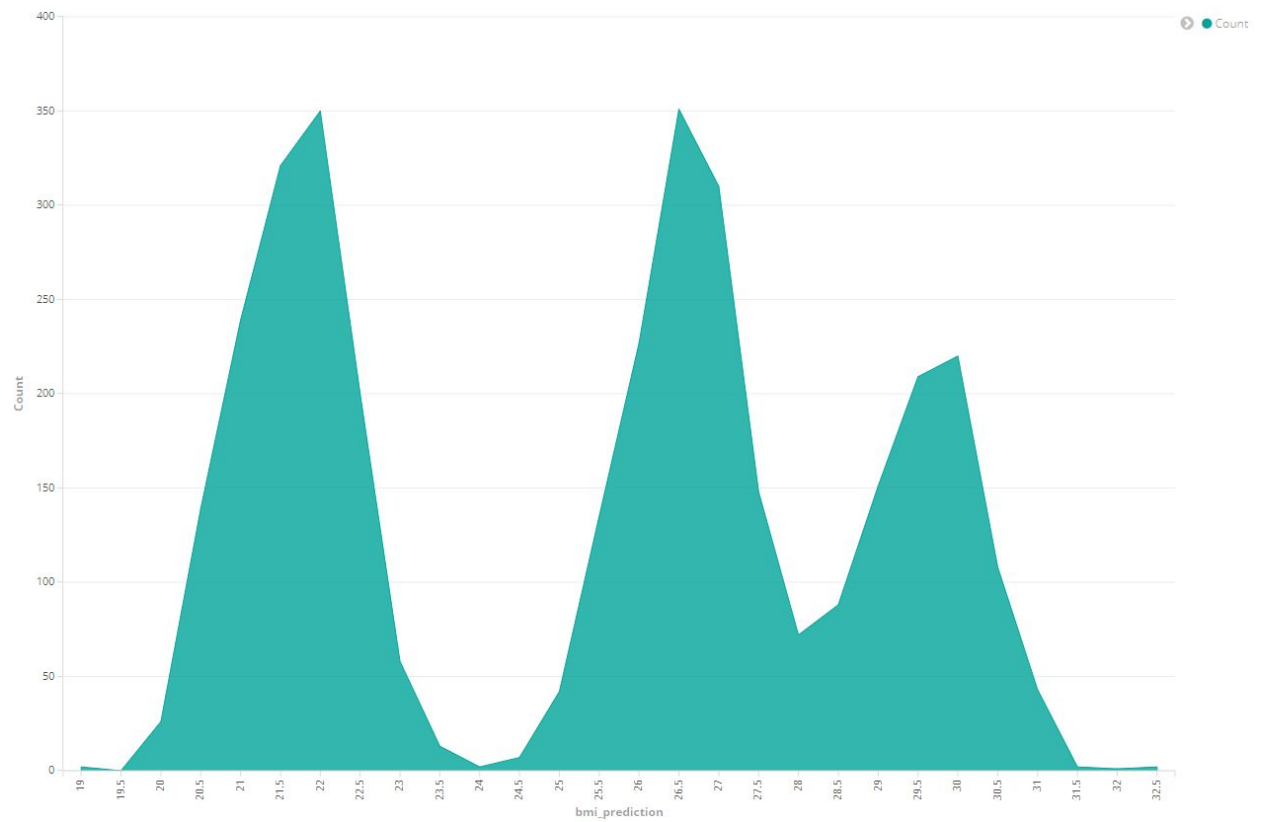
Left Screenshot: The 'Metrics' section shows 'Y-Axis Count' selected. The 'Buckets' section shows 'Select buckets type' with 'X-Axis' selected (highlighted with a red circle). The 'Add metrics' button is visible.

Right Screenshot: The 'Buckets' configuration panel is shown. The 'X-Axis' bucket is selected (highlighted with a red circle). The 'Aggregation' is set to 'Histogram' (highlighted with a red underline). The 'Field' is set to 'bmi' (highlighted with a red underline). The 'Minimum Interval' is set to '0.5' (highlighted with a red underline). The 'Show empty buckets' checkbox is unchecked. The 'Custom Label' field is empty. The 'Add sub-buckets' button is visible. The 'Play' button in the top right corner is highlighted with a red circle.

This results in a graph of the true BMI values in the dataset:

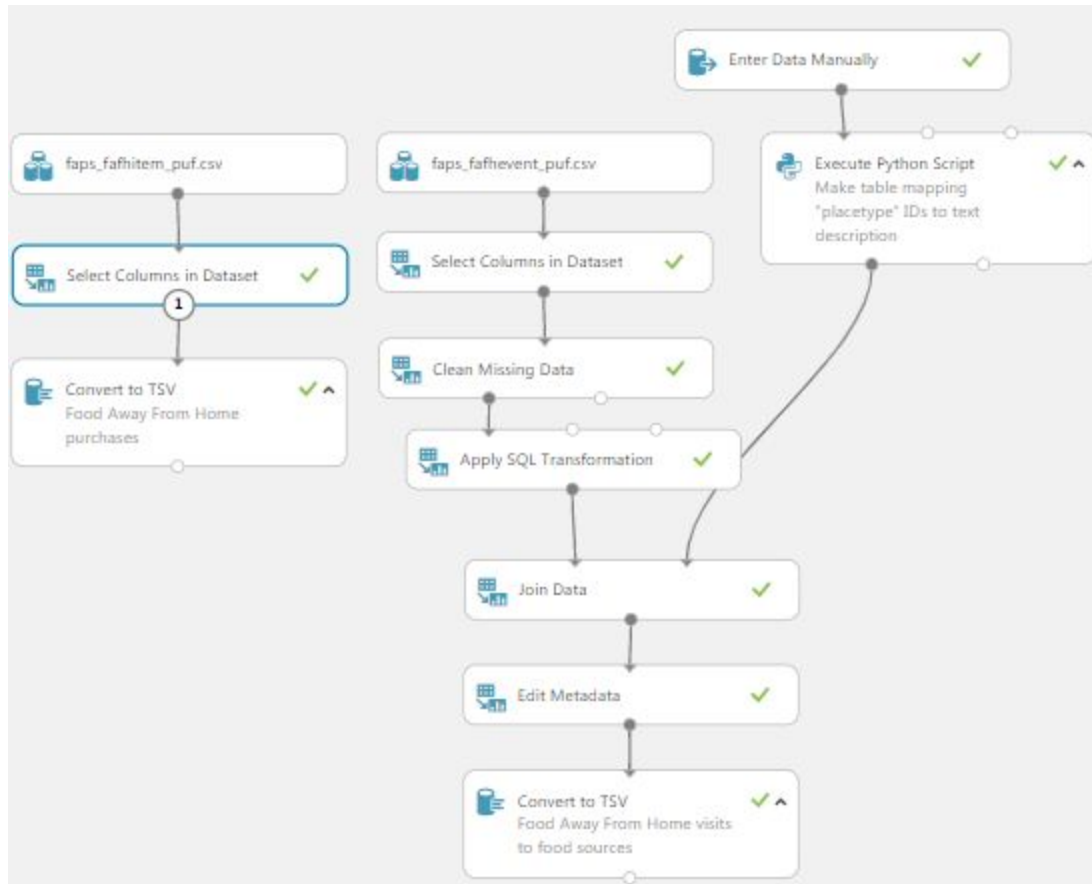


16. Change the field to **bmi_prediction** to see the graph of the predicted bmi:



Step 4 (Optional): Other Visualizations

1. If you want to make a word cloud of restaurants visited and food bought, or a timeline, you can make a new mini-experiment to transform the **faps_fafhitem_puf** and **faps_fafhevent_puf** datasets we uploaded into something we can export to Elasticsearch as well. Set up the modules like so:



7. In the **Select Columns in Dataset** for **faps_fafhitem_puf**, select columns: hhnum,eventid,itemnum,itemdesc

8. In the **Select Columns in Dataset** for **faps_fafhevent_puf**, select columns: eventid,hhnum,date,placetype,placedist_s,totalpaid,breakfast,lunch,dinner_supper,snack_drink,whogotpnnum

9. In the **Clean Missing Data** module, select the columns: breakfast,lunch,dinner_supper,snack_drink and set the **Cleaning mode** to **Custom substitution value**. Set the **Replacement value** to **0**.

10. In the **Apply SQL Transformation** module, use the SQL script from Github:

https://github.com/shozonu/usda-foodaps-bmi-model/blob/master/sql/fafh_event_clean.sql

```
update t1 set breakfast = 0 where breakfast < 0;
update t1 set lunch = 0 where lunch < 0;
update t1 set dinner_supper = 0 where dinner_supper < 0;
update t1 set snack_drink = 0 where snack_drink < 0;
update t1 set placedist_s = null where placedist_s < 0;
select * from t1;
```

11. In **faps_fafhevent_puf**, the **placetype** column is an integer coded to a specific text description. The code table is in the official data handbook at the USDA ERS dataset link. We can copy the entire table from the handbook but it is not formatted correctly and there are some irrelevant lines. Copy/paste the text from the Github into the **Enter Data Manually** node:

https://github.com/shozonu/usda-foodaps-bmi-model/blob/master/Data/event_placetype_code_raw.txt

And set the **DataFormat** to **TSV**. Check the **HasHeader** box.

Enter Data Manually

DataFormat

TSV

☒ HasHeader

Data

```
198 325
199 51
200 0.32
201 Place of worship
202 326
203 1
204 0.01
205 Preschool
206 327
207 4
208 0.03
209 School
210 328
211 16
212 0.10
213 Work
214 402
215 48
216 0.30
217 Unknown
```

12. In the **Execute Python Script** module, use the code from Github to construct a table of the descriptions and the placetype key that we can later join with the data:
https://github.com/shozonu/usda-foodaps-bmi-model/blob/master/py/fafh_event_placeid_desc.py

```
import pandas as pd

def azureml_main(df = None):
    # Create dictionary to hold values
    arr = {"desc" : [], "id" : []}
    for index, row in df.iterrows():
        # Place description occurs every nth row,
        # Corresponding place id occurs every n-3th row
        if((index + 1) % 4 == 0):
            arr["id"].append(int(df["raw"].iat[index - 3]))
            arr["desc"].append(df["raw"].iat[index])

    # Construct new dataframe from dictionary
    data = pd.DataFrame.from_dict(arr)

    # Rearrange columns so that id is first
    cols = data.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    data = data[cols]
    return data
```

13. In the **Join Data** module, set the **Join key columns for L** to **placetype** and **Join key columns for R** to **id**. Make sure the **Match case** box is **checked**. Set the **Join type** to **Inner Join**. Make sure the **Keep right key column** is **unchecked**.

14. In the **Edit Metadata** module, select **whogotpnun** and type **pnum** for the **New column names**.

15. After saving and running the experiment, download the two datasets by right-clicking on the two **Convert to TSV** modules' output and clicking **download**. Make sure to rename the downloaded files to something you can remember.

16. Follow the same basic steps in [Step 3: Visualization](#) for uploading the data with Kibana:

Machine Learning > Data Visualizer > Import

But for these two datasets, you need to change the import settings when you get to the **Import** stage.

17. For **fafh_item**, click the **Advanced** tab and edit the mappings. Under **itemdesc**, add a comma to the end of the existing line, and add a line:

```
8  "itemdesc": {  
9    "type": "text",  
10   "fielddata": true  
11 },
```

This allows the text field to be aggregatable for the purposes of making the word cloud.

18. We need to change **fafh_event** as well. Change the **desc** field and **date** field like so:

Mappings

```
1  {  
2    "breakfast": {  
3      "type": "long"  
4    },  
5    "date": {  
6      "type": "date",  
7      "format": "MM/dd/yyyy hh:mm:ss a"  
8    },  
9    "desc": {  
10     "type": "text",  
11     "fielddata": true  
12   },  
13   "dinner_supper": {  
14     "type": "long"  
15   },  
16   "eventid": {  
17     "type": "long"  
18   },  
19   "hhnum": {  
20     "type": "long"  
21   },
```

In order to properly import the date for our time visualization, we needed to change the type to "date" and specify the date format that our dataset used.

19. Once you successfully uploaded both of them with the modified mappings, check them in **Discover**.

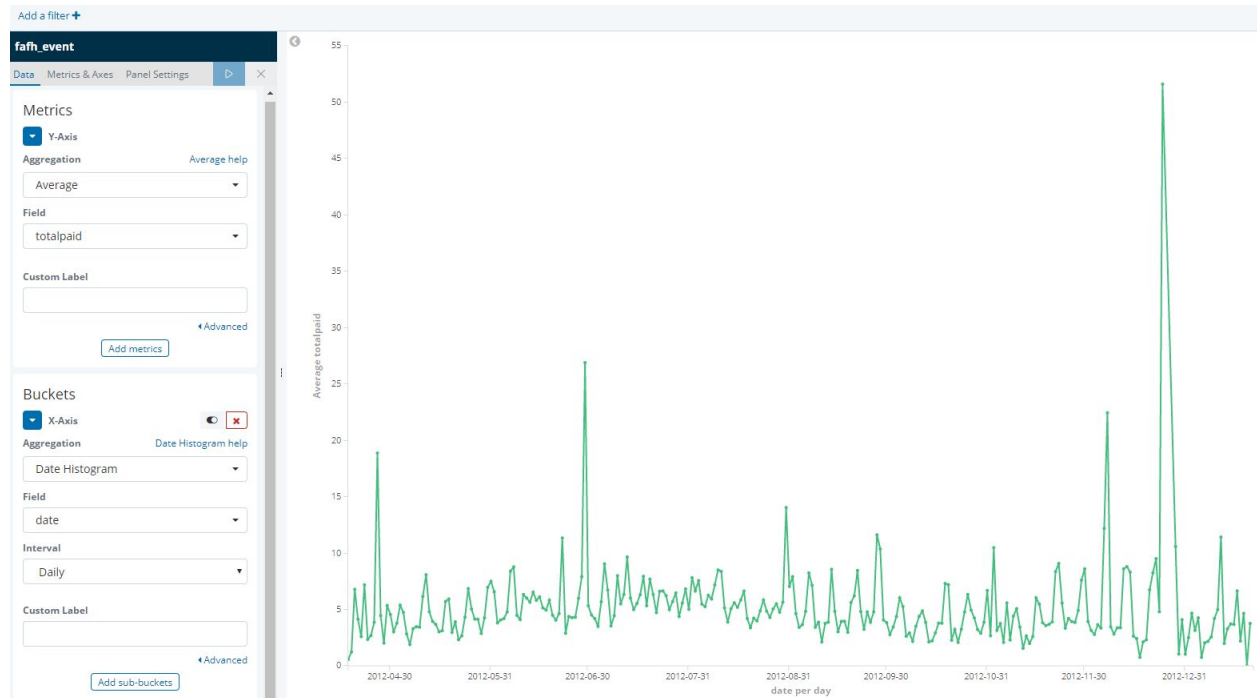
The screenshot shows the Kibana Discover interface. The left sidebar contains navigation links: Discover, Visualize, Dashboard, Timelion, Canvas, Machine Learning, Infrastructure, Logs, APM, Graph, Dev Tools, Monitoring, and Management. The main panel displays search results for the query '>_ search... (e.g. status:200 AND extension:PHP)'. The selected index is 'fafh_item'. The 'Selected fields' section shows '_source'. The 'Available fields' section lists: t _id, t _index, # _score, t _type, # eventid, # hhnum, t itemdesc, and # itemnum. The search results table shows 116,074 hits. The first few rows of data are:

eventid	hhnum	itemnum	itemdesc
66,824	101,599	1	GROCERY
66,826	101,599	2	BISCUIT
66,826	101,599	3	APPLE C
66,826	101,599	6	FRIED S
66,826	101,599	4	SAUSAGE
66,826	101,599	5	GRAVY
66,826	101,599	1	EGGS (I
66,832	101,599	2	REGULAR
66,832	101,599	1	French
21,737	101,601	5	WINE, R

The screenshot shows the Kibana Discover interface. The left sidebar contains navigation links: Discover, Visualize, Dashboard, Timelion, Canvas, Machine Learning, Infrastructure, Logs, APM, Graph, Dev Tools, Monitoring, and Management. The main panel displays search results for the query '>_ search... (e.g. status:200 AND extension:PHP)'. The selected index is 'fafh_event'. The 'Selected fields' section shows '_source'. The 'Available fields' section lists: t _id, t _index, # _score, t _type, # breakfast, @ date, t desc, # dinner_supper, # eventid, # hhnum, # lunch, # placedist_s, # placetype, # pnum, # snack_drink, and # totalpaid. The search results table shows 30,030 hits. The first few rows of data are:

eventid	hhnum	pnum	date	place
63,198	100,214	1	June 29th 2012, 17:00:00.000	place
69,123	101,187	3	December 4th 2012, 16:00:00.000	pl
69,124	101,187	3	December 5th 2012, 16:00:00.000	pl
69,126	101,187	4	December 4th 2012, 16:00:00.000	pl
57,499	101,323	10	August 13th 2012, 17:00:00.000	pl
62,570	102,167	1	June 10th 2012, 17:00:00.000	place
7,996	102,269	2	May 13th 2012, 17:00:00.000	placety
52,823	102,862	1	October 2nd 2012, 17:00:00.000	pla
70,417	103,023	2	December 13th 2012, 16:00:00.000	p

20. In **Visualize**, make a new **Line Graph** with the **fafh_event** index and set their settings as follows to create the time-series graph of average money spent on food during the period. You can set the time filter to **Absolute**, between **April 2012** and **January 2013**.



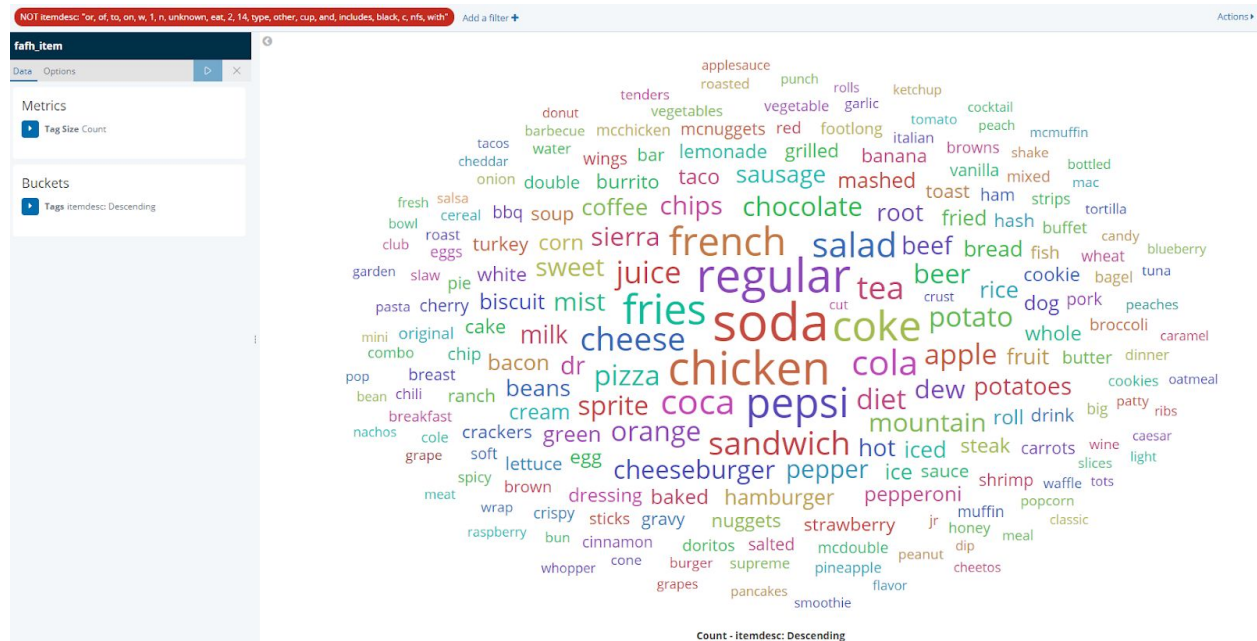
The screenshot displays the Kibana visualization editor for a visualization named 'fahf_event'. The interface is divided into several sections:

- Left Sidebar:**
 - Metrics:** A dropdown menu showing 'Tag Size: Count'.
 - Buckets:** A dropdown menu showing 'Tags'.
 - Aggregation:** A dropdown menu showing 'Terms'.
 - Field:** A dropdown menu showing 'desc'.
 - Order By:** A dropdown menu showing 'metric: Count'.
 - Order:** A dropdown menu showing 'Descendir'.
 - Size:** A dropdown menu showing '156'.
 - Custom Label:** An empty text input field.
- Main Panel:**
 - Text Scale:** A dropdown menu showing 'square root'.
 - Orientations:** A dropdown menu showing 'single'.
 - Font Size:** A slider control with values '18' and '72'.
 - Show Label:** A checkbox that is checked.
- Right Sidebar:**
 - Add filter:** A dialog box with a dropdown menu showing 'desc', a dropdown menu showing 'is not one of', and a text input field showing 'Values...'.
- Bottom Status Bar:**
 - NOT desc:** A list of terms: '*of, on, including, not, co, other, non, op, stop, small, unknown, wcores, misc, school, home*'.
 - of:** 1
 - on:** 1
 - including:** 1
 - not:** 1
 - co:** 1
 - other:** 1
 - non:** 1
 - op:** 1
 - stop:** 1
 - small:** 1
 - unknown:** 1
 - wcores:** 1
 - misc:** 1
 - school:** 1
 - home:** 1



Count - desc: Descending

22. Repeat the same basic steps as step 21 for **fafh_item** to create a word cloud describing food bought away from home. Set the filter words to any words you see in the word cloud that are not helpful.



References

1. Data Source:
<https://www.ers.usda.gov/data-products/foodaps-national-household-food-acquisition-and-purchase-survey/>
2. Project Github:
<https://github.com/shozonu/usda-foodaps-bmi-model>
3. Azure ML Studio Specifications:
<https://azure.microsoft.com/en-us/pricing/details/machine-learning-studio/>
4. Elasticsearch Service Specifications:
<https://www.elastic.co/guide/en/cloud/current/ec-reference-hardware.html>
5. Azure ML Project Gallery:
<https://gallery.cortanaintelligence.com/Experiment/USDA-FoodAPS-BMI-Modeling>