Network anomaly detection

Multiclass classification model to predict network anomolies

- Feb 2021
- Domino Valdano and Frank McQuillan

Summary

Our primary focus in this project was preprocessing and feature selection, rather than trying many different supervised learning algorithms. We landed here after working with the data for some time and trying to understand it in depth. Initial trys indicated that, for example, small differences in sampling methodologies resulted in very different accuracies on the validation dataset. This is, we suppose, the nature of this type of severely imbalanced problem.

In the end we found that oversampling minority classes and undersampling majority classes in a relatively simple way resulted in reasonable accuracy. However, oversampling minority classes by duplication tends to result in overfitting, as we typically achieved greater than 90% accuracy on both test/train splits, but only 60-something% on the public leaderboard. SMOTE-type approaches [1,2] were tried at for oversampling but did not figure in our final submission.

Preprocessing and feature selection

Started by plotting histograms to look at the nature of the data skew. Used data summarization methods to look at descriptive statistics like correlation. Tried clustering to see if decision boundaries were clean (they are not). Then use oversampling minority classes and undersampling majority classes to generate the data to feed to the random forest model.

Training methodology

We used a random forest with the following parameters:

```
10::integer,            -- num trees
5::integer,             -- num random features
5::integer,             -- num permutations
10::integer,            -- max tree depth
3::integer,             -- min split
1::integer,             -- min bucket
10::integer,            -- num splits
```

We kept this constant while we experimented with sampling approaches. Random forest is commonly used on anomaly detection problems [3] and also has the benefit of not requiring 1-hot encoding or normalization/standardization. At least, normalization/standardization is less important in tree methods than other supervised learning methods.

Notable aspects

Keep it simple. Don't try to reverse engineer rules-based approaches since that is very time consuming and you will likely miss something important. Many of our laptop based Python methods had memory issues (e.g., with SMOTENC) so we found working in an MPP database very convenient, since ~5M examples is pretty small in MPP land. So we used Greenplum with Apache MADlib for most of the heavy lifting.

But most of all, we had fun trying things out!

References

[1] SMOTE for Imbalanced Classification with Python https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ (https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/)

[2] SMOTE algorithms https://imbalanced-learn.org/stable/references/over_sampling.html#smote-algorithms (https://imbalanced-learn.org/stable/references/over_sampling.html#smote-algorithms)

[3] Anomaly detection using random forest: A performance revisited https://ieeexplore.ieee.org/document/8285847 (https://ieeexplore.ieee.org/document/8285847)

Inline code comments

Please see notebook below.

Setup

Data Loading

Data Exploration

Feature Engineering

Model Development

- **Random Forest (MADlib)**
  - Train model
  - Variable Importance
  - Score Validation Data
  - Confusion Matrix

Write out results

- CSV output

Setup

Create database connection and import libraries

In [1]:
```
%load_ext sql
```

In [2]:
```
# Greenplum Database 5.x on GCP (PM demo machine) - via tunnel
%sql postgresql://gpadmin@localhost:8000/madlib

# PostgreSQL local
#%sql postgresql://fmcquillan@localhost:5432/madlib
```

In [3]:
```
# dependencies
import psycopg2                # Python-PostgreSQL Database Adapter - https://pypi.python.org/pypi/psycopg2
import pandas as pd            # Python Data Analysis Library - https://pandas.pydata.org/
import seaborn as sns          # Statistical data visualization - https://seaborn.pydata.org/
import math                    # Mathematical functions - https://docs.python.org/2/library/math.html
import textwrap as tw          # Text wrapping and filling - https://docs.python.org/2/library/textwrap.html
import ipywidgets as widgets   # Jupyter Widgets - https://ipywidgets.readthedocs.io/en/latest/
import IPython.display as ipd # http://ipython.org/documentation.html
```

In [4]:
```
# package options
# %matplotlib inline
%pylab inline

pylab.rcParams['figure.figsize'] = (12, 8)

pd.options.mode.chained_assignment = None
pd.set_option('display.max_colwidth', -1)

pd.options.display.max_rows = 10000
pd.options.display.max_columns = 10000

sns.set(style="darkgrid")
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [5]:
```
# init to default values
database_host = 'localhost'
database_databasename = 'madlib'
database_username = 'gpadmin'
database_password = ''
database_port = '8000'

# interpret string as markdown
def printmd(string):
    ipd.display(ipd.Markdown(string))

# forms
message = "### Connection Details \n ------"
printmd(message)

printmd("**Host:**")
inputHost = widgets.Text()
ipd.display(inputHost)

printmd("**Port:**")
inputPort = widgets.Text()
ipd.display(inputPort)

printmd("**Database Name:**")
inputDatabaseName = widgets.Text()
ipd.display(inputDatabaseName)

printmd("**Username:**")
inputUsername = widgets.Text()
ipd.display(inputUsername)

printmd("**Password:**")
inputPassword = widgets.Text()
ipd.display(inputPassword)

printmd("*Leave blank for default values*")


def db_connect():
    global conn, cur
    try:
        connString = "host='{}' dbname='{}' user='{}' password='{}' port
={}".format(database_host,database_databasename,database_username,databa
se_password,database_port)
        # print connString
        conn = psycopg2.connect(connString)
        cur = conn.cursor()
        conn.autocommit = True
        message = "<span style='color:green'>**Connection successful!**
</span>"
        printmd(message)
    except:
        message = "<span style='color:red'>**ERROR: Unable to connect to
the database**</span>"
        printmd(message)
```

```
def on_button_click(b):

    global database_host, database_databasename, database_username, data
base_password, database_port

    ipd.clear_output()

    message = "### Connection Details \n ------"
    printmd(message)

    if inputHost.value == "":
        message = "**Host:** {} (default)".format(database_host)
        printmd(message)
    else:
        database_host = inputHost.value
        message = "**Host:** {}".format(database_host)
        printmd(message)

    if inputPort.value == "":
        message = "**Port:** {} (default)".format(database_port)
        printmd(message)
    else:
        database_port = inputPort.value
        message = "**Port:** {}".format(database_port)
        printmd(message)

    if inputDatabaseName.value == "":
        message = "**Database name:** {} (default)".format(database_data
basename)
        printmd(message)
    else:
        database_databasename = inputDatabaseName.value
        message = "**Database name:** {}".format(database_databasename)
        printmd(message)

    if inputUsername.value == "":
        message = "**Username:** {} (default)".format(database_username)
        printmd(message)
    else:
        database_username = inputUsername.value
        message = "**Username:** {}".format(database_username)
        printmd(message)

    if inputPassword.value == "":
        message = "**Password:** {} (default)".format(database_password)
        printmd(message)
    else:
        database_password = inputPassword.value
        message = "**Password:** ##########"
        printmd(message)

    printmd("------")
    db_connect()

button = widgets.Button(description="Connect")
ipd.display(button)
button.on_click(on_button_click)
```

Connection Details

Host: localhost

Port: 8000

Database name: madlib

Username: gpadmin

Password: (default)

Connection successful!

In [7]:
```
def bar_plot(data,title,x,xLabel,y,yLabel,color=None,xAxisRotation=90):

    # Bar plot
    pylab.rcParams['figure.figsize'] = (12, 8)
    seq_col_brew = sns.color_palette("Blues_r", 1)
    sns.color_palette(seq_col_brew)
    if color != None:
        plt = sns.barplot(x=x, y=y, data=data, color=color)
    else:
        plt = sns.barplot(x=x, y=y, data=data)

    # titles
    plt.set_title(title,fontsize=30)
    plt.set_xlabel(xLabel,fontsize=12)
    plt.set_ylabel(yLabel,fontsize=12)

    # rotate x axis labels
    for item in plt.get_xticklabels():
        item.set_rotation(xAxisRotation)

    # remove scientific notation
    plt.ticklabel_format(style='plain', axis='y')
```

In [8]:
```
database_host = 'localhost'
database_databasename = 'madlib'
database_username = 'gpadmin'
database_password = ''
database_port = '8000'
db_connect()
```

Connection successful!

In [9]:
```
# helper function
def query_gpdb(query):

    cur.execute(query)

    colnames = [desc[0] for desc in cur.description]
    return pd.DataFrame(cur.fetchall(), columns=colnames)
```

In [8]:
```
%sql SET search_path=public,madlib
```

```
Done.
```

Out[8]:
```
[]
```

Data Loading

Training dataset

In [ ]:
```
%%sql
DROP TABLE IF EXISTS training_data;
CREATE TABLE training_data (
    a1 TEXT,
    a2 TEXT,
    a3 TEXT,
    a4 FLOAT,
    a5 FLOAT,
    a6 INTEGER,
    a7 INTEGER,
    a8 INTEGER,
    a9 INTEGER,
    a10 INTEGER,
    a11 INTEGER,
    a12 FLOAT,
    a13 FLOAT,
    a14 FLOAT,
    a15 INTEGER,
    a16 FLOAT,
    a17 FLOAT,
    a18 FLOAT,
    a19 FLOAT,
    a20 FLOAT,
    a21 FLOAT,
    a22 FLOAT,
    a23 FLOAT,
    a24 INTEGER,
    a25 FLOAT,
    a26 FLOAT,
    a27 FLOAT,
    a28 FLOAT,
    a29 FLOAT,
    a30 FLOAT,
    a31 FLOAT,
    a32 FLOAT,
    a33 FLOAT,
    a34 FLOAT,
    a35 FLOAT,
    a36 FLOAT,
    a37 FLOAT,
    a38 FLOAT,
    a39 FLOAT,
    a40 FLOAT,
    a41 FLOAT,
    a42 INTEGER,
    y TEXT
);

COPY training_data FROM '/home/gpadmin/network_data/training.csv' CSV DE
LIMITER ',';
```

Add row number to training data

In [13]:
```
%%sql
DROP TABLE IF EXISTS training_data_id;
CREATE TABLE training_data_id AS
SELECT ROW_NUMBER() OVER()-1 AS id, * FROM training_data;
```

```
Done.
4898431 rows affected.
```

Out[13]: []

In [14]:
```
%%sql
DROP TABLE training_data;
ALTER TABLE training_data_id RENAME TO training_data;
```

```
Done.
Done.
```

Out[14]: []

Have a look at some sample data

In [15]:
```
%%sql
SELECT * FROM training_data ORDER BY id LIMIT 5;
```

```
5 rows affected.
```

Out[15]:

id	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16
0	ICMP	ECR_I	STAT10	1032.0	0.0	0	0	0	0	0	0	0.0	0.0	1.0	0	1397.0
1	TCP	PRIVATE	STAT06	0.0	0.0	0	0	0	0	0	0	0.0	0.0	0.08	0	1253.0
2	ICMP	ECR_I	STAT10	1032.0	0.0	0	0	0	0	0	0	0.0	0.0	1.0	0	1052.0
3	TCP	PRIVATE	STAT06	0.0	0.0	0	0	0	0	0	0	0.0	0.0	0.01	0	996.0
4	UDP	NTP_U	STAT10	48.0	48.0	0	0	0	0	0	0	0.0	0.0	1.0	0	1082.0

In [10]:
```
%%sql
SELECT COUNT(*) FROM training_data;
```

```
1 rows affected.
```

Out[10]:

count
4898431

Evaluation dataset

In [ ]:
```
%%sql
DROP TABLE IF EXISTS eval_data;
CREATE TABLE eval_data (
    id INTEGER,
    a1 TEXT,
    a2 TEXT,
    a3 TEXT,
    a4 FLOAT,
    a5 FLOAT,
    a6 INTEGER,
    a7 INTEGER,
    a8 INTEGER,
    a9 INTEGER,
    a10 INTEGER,
    a11 INTEGER,
    a12 FLOAT,
    a13 FLOAT,
    a14 FLOAT,
    a15 INTEGER,
    a16 FLOAT,
    a17 FLOAT,
    a18 FLOAT,
    a19 FLOAT,
    a20 FLOAT,
    a21 FLOAT,
    a22 FLOAT,
    a23 FLOAT,
    a24 INTEGER,
    a25 FLOAT,
    a26 FLOAT,
    a27 FLOAT,
    a28 FLOAT,
    a29 FLOAT,
    a30 FLOAT,
    a31 FLOAT,
    a32 FLOAT,
    a33 FLOAT,
    a34 FLOAT,
    a35 FLOAT,
    a36 FLOAT,
    a37 FLOAT,
    a38 FLOAT,
    a39 FLOAT,
    a40 FLOAT,
    a41 FLOAT,
    a42 INTEGER
);

COPY eval_data FROM '/home/gpadmin/network_data/eval-rev2-1.csv' DELIMIT
ER ',' CSV HEADER;
```

Have a look at some rows from the eval dataset

In [11]:
```
%%sql
SELECT * FROM eval_data ORDER BY id LIMIT 5;
```

5 rows affected.

Out[11]:

id	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16
0	ICMP	ECR_I	STAT10	1032.0	0.0	0	0	0	0	0	0	0.0	0.0	1.0	0	1328.0
1	TCP	HTTP	STAT10	280.0	3413.0	0	0	0	0	1	0	0.0	0.0	1.0	0	674.0
2	TCP	HTTP	STAT10	218.0	1493.0	0	0	0	0	1	0	0.0	0.0	1.0	0	990.0
3	ICMP	ECR_I	STAT10	520.0	0.0	0	0	0	0	0	0	0.0	0.0	1.0	0	1178.0
4	ICMP	ECR_I	STAT10	1032.0	0.0	0	0	0	0	0	0	0.0	0.0	1.0	0	1554.0

In [12]:
```
%%sql
SELECT COUNT(*) FROM eval_data;
```

1 rows affected.

Out[12]:

count
311030

Data Exploration

Frequency histogram

In [48]:
```
%%sql
SELECT y, class_count, (class_count/SUM(class_count) OVER ()) AS percent
FROM
(
SELECT y, COUNT(*) AS class_count FROM training_data GROUP BY y
) q ORDER BY percent DESC;
```

```
23 rows affected.
```

Out[48]:

y	class_count	percent
class18	2807886	0.57322150704991047133
class10	1072017	0.21884905595281427869
normal	972781	0.19859032412623552317
class17	15892	0.00324430414555191244
class06	12481	0.00254795872392608980
class15	10413	0.00212578272512157464
class11	2316	0.00047280445514084000
class01	2203	0.00044973584398759521
class21	1020	0.00020822994138327150
class20	979	0.00019985991432766941
class14	264	0.00005389480858023211922
class04	53	0.00001081979107187587209O
class02	30	0.00000612441004068445590O
class07	21	0.00000428708702847911913O
class22	20	0.00000408294002712297060O
class05	12	0.00000244976401627378236O
class16	10	0.00000204147001356148530O
class08	9	0.00000183732301220533677O
class03	8	0.00000163317601084918824O
class09	7	0.00000142902900949303971O
class13	4	8.16588005424594120E-7
class12	3	6.12441004068445590E-7
class19	2	4.08294002712297060E-7

So, this is an extremely unbalanced dataset!

Summary Deterministic Statistics

https://madlib.apache.org/docs/latest/group__grp__summary.html
(https://madlib.apache.org/docs/latest/group__grp__summary.html)

In [ ]:
```
%%sql
DROP TABLE IF EXISTS training_summary;
SELECT * FROM madlib.summary( 'training_data',   -- Source table
                              'training_summary'    -- Output table
                            );
```

In [17]:
```
%%sql
SELECT * FROM training_summary;
```

```
43 rows affected.
```

Out[17]:

group_by	group_by_value	target_column	column_number	data_type	row_count	distinct_values
None	None	a1	1	text	4898431	3
None	None	a2	2	text	4898431	70
None	None	a3	3	text	4898431	11
None	None	a4	4	float8	4898431	7195
None	None	a5	5	float8	4898431	22788
None	None	a6	6	int4	4898431	2
None	None	a7	7	int4	4898431	2
None	None	a8	8	int4	4898431	3
None	None	a9	9	int4	4898431	2
None	None	a10	10	int4	4898431	2
None	None	a11	11	int4	4898431	2
None	None	a12	12	float8	4898431	42
None	None	a13	13	float8	4898431	72

None	None	a14	14	float8	4898431	101
None	None	a15	15	int4	4898431	3
None	None	a16	16	float8	4898431	3083
None	None	a17	17	float8	4898431	101
None	None	a18	18	float8	4898431	9883
None	None	a19	19	float8	4898431	101
None	None	a20	20	float8	4898431	512
None	None	a21	21	float8	4898431	6
None	None	a22	22	float8	4898431	90
None	None	a23	23	float8	4898431	256
None	None	a24	24	int4	4898431	3

None	None	a25	25	float8	4898431	101
None	None	a26	26	float8	4898431	101
None	None	a27	27	float8	4898431	76
None	None	a28	28	float8	4898431	93
None	None	a29	29	float8	4898431	98
None	None	a30	30	float8	4898431	101
None	None	a31	31	float8	4898431	101
None	None	a32	32	float8	4898431	95
None	None	a33	33	float8	4898431	101
None	None	a34	34	float8	4898431	10

None	None	a35	35	float8	4898431	256
None	None	a36	36	float8	4898431	30
None	None	a37	37	float8	4898431	6
None	None	a38	38	float8	4898431	95
None	None	a39	39	float8	4898431	101
None	None	a40	40	float8	4898431	512
None	None	a41	41	float8	4898431	100
None	None	a42	42	int4	4898431	1
None	None	y	43	text	4898431	23

In [ ]:
```
%%sql
DROP TABLE IF EXISTS eval_summary;
SELECT * FROM madlib.summary( 'eval_data',   -- Source table
                              'eval_summary'    -- Output table
                            );
```

In [18]:
```
%%sql
SELECT * FROM eval_summary;
```

```
43 rows affected.
```

Out[18]:

group_by	group_by_value	target_column	column_number	data_type	row_count	distinct_values
None	None	id	1	int4	311030	311030
None	None	a1	2	text	311030	3
None	None	a2	3	text	311030	65
None	None	a3	4	text	311030	11
None	None	a4	5	float8	311030	2504
None	None	a5	6	float8	311030	8906
None	None	a6	7	int4	311030	2
None	None	a7	8	int4	311030	2
None	None	a8	9	int4	311030	3
None	None	a9	10	int4	311030	2
None	None	a10	11	int4	311030	2
None	None	a11	12	int4	311030	2
None	None	a12	13	float8	311030	12
None	None	a13	14	float8	311030	87

None	None	a14	15	float8	311030	93
None	None	a15	16	int4	311030	3
None	None	a16	17	float8	311030	2736
None	None	a17	18	float8	311030	101
None	None	a18	19	float8	311030	745
None	None	a19	20	float8	311030	101
None	None	a20	21	float8	311030	442
None	None	a21	22	float8	311030	4
None	None	a22	23	float8	311030	92
None	None	a23	24	float8	311030	256
None	None	a24	25	int4	311030	4
None	None	a25	26	float8	311030	100

None	None	a26	27	float8	311030	101
None	None	a27	28	float8	311030	58
None	None	a28	29	float8	311030	21
None	None	a29	30	float8	311030	24
None	None	a30	31	float8	311030	100
None	None	a31	32	float8	311030	81
None	None	a32	33	float8	311030	79
None	None	a33	34	float8	311030	101
None	None	a34	35	float8	311030	5
None	None	a35	36	float8	311030	256
None	None	a36	37	float8	311030	18
None	None	a37	38	float8	311030	5
None	None	a38	39	float8	311030	100

None	None	a39	40	float8	311030	78
None	None	a40	41	float8	311030	489
None	None	a41	42	float8	311030	100
None	None	a42	43	int4	311030	1

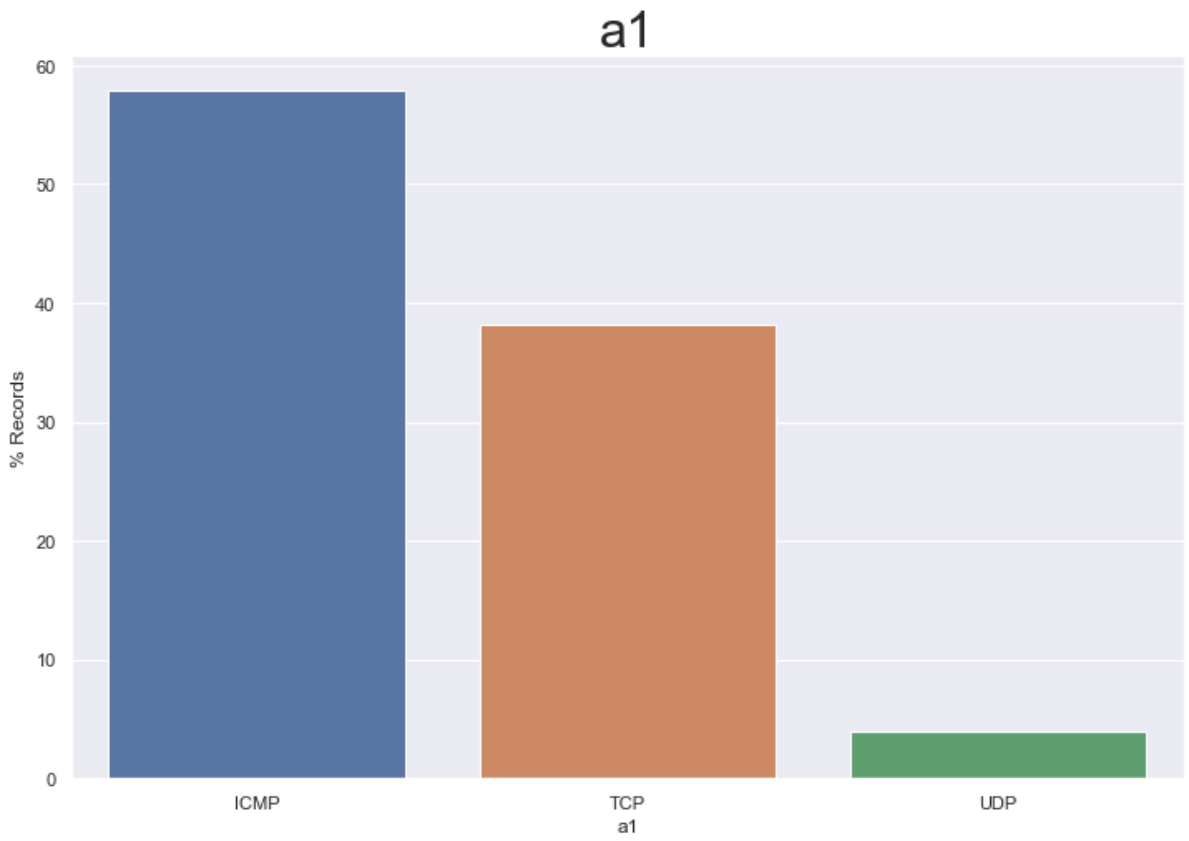
Columns a4 and a5 have a range of 1.3-1.4B, way more than other columns. Column a42 has all the same values so can be ignored. Let's create a widget to look at indidual features:

Categorical Columns

In [13]:
```
catColumns = ['a1','a2','a3', 'a6','a7','a8', 'a9',
              'a10','a11', 'a15','a24','a42',
              'y']

def bar_plot(data,title,x,xLabel,y,yLabel,color=None,xAxisRotation=90):

    # Bar plot
    pylab.rcParams['figure.figsize'] = (12, 8)
    seq_col_brew = sns.color_palette("Blues_r", 1)
    sns.color_palette(seq_col_brew)
    if color != None:
        plt = sns.barplot(x=x, y=y, data=data, color=color)
    else:
        plt = sns.barplot(x=x, y=y, data=data)

    # titles
    plt.set_title(title,fontsize=30)
    plt.set_xlabel(xLabel,fontsize=12)
    plt.set_ylabel(yLabel,fontsize=12)

    # rotate x axis labels
    for item in plt.get_xticklabels():
        item.set_rotation(xAxisRotation)

    # remove scientific notation
    plt.ticklabel_format(style='plain', axis='y')


def get_cat_data_frame(col):
    query = """
        SELECT *
              ,round((record_count * 100.0) / sum(record_count) OVER(),
2) AS perc_records
        FROM (
            SELECT {} AS col
                  ,count(*) AS record_count
            FROM public.training_data
            GROUP BY 1
        ) foo
        ORDER BY perc_records DESC
    """.format(col)
    cur.execute(query)

    colnames = [desc[0] for desc in cur.description]
    return pd.DataFrame(cur.fetchall(), columns=colnames)

def on_cat_selection(res):
    if res['type'] == 'change' and res['name'] == 'value':
        ipd.clear_output()
        printmd("-----\n **Select Column:**")
        ipd.display(catDropdown)
        df = get_cat_data_frame(res['new'])
        bar_plot(df,res['new'],"col",res['new'],"perc_records","% Record
s", None, 0)

catDropdown = widgets.Dropdown(
```

```
    options=catColumns,
    value=catColumns[0],
    description='Column:',
    disabled=False,
)

catDropdown.observe(on_cat_selection)
printmd("-----\n **Select Column:**")
ipd.display(catDropdown)
df = get_cat_data_frame(catColumns[0])
bar_plot(df,catColumns[0],"col",catColumns[0],"perc_records","% Records"
, None, 0)
```

Select Column:

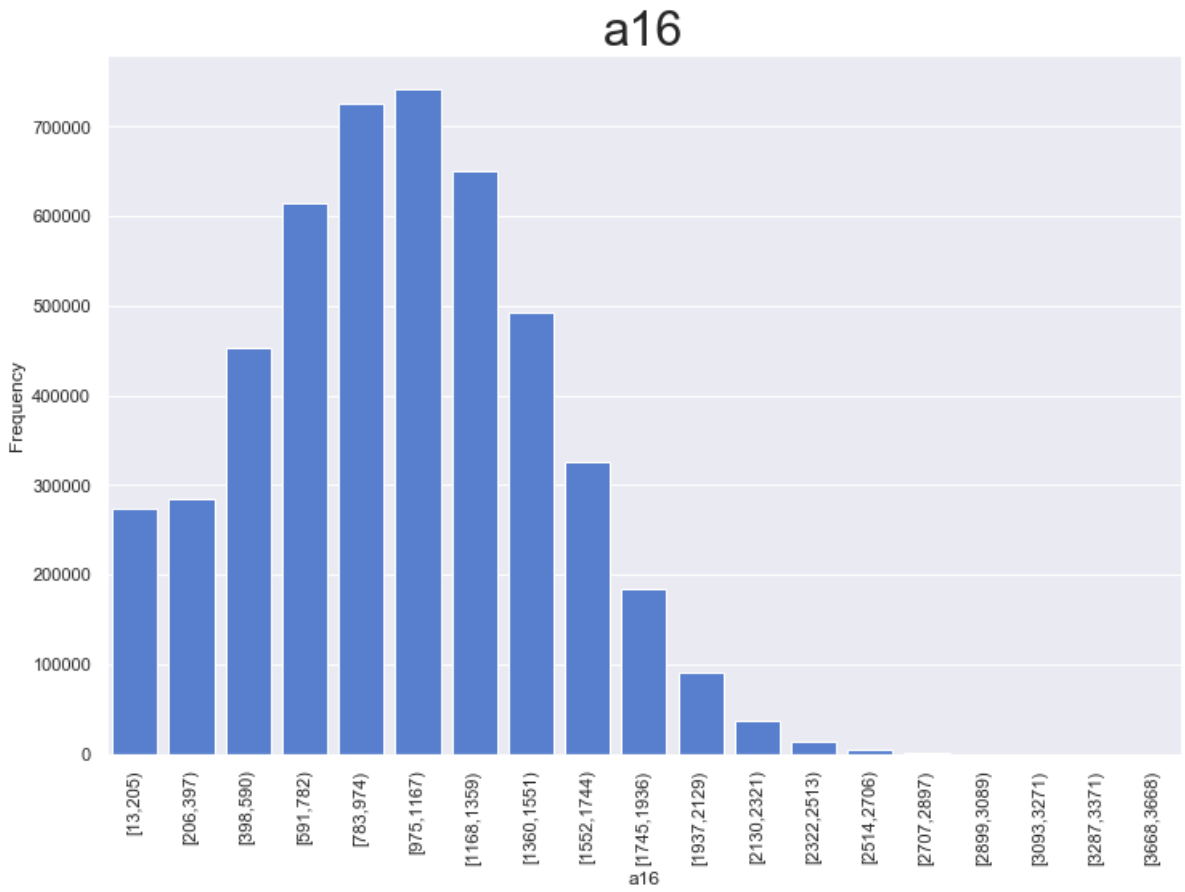


- Low % of values in any one class can skew model results and/or create unstable model.
- Histogram values are being calculated in the database - minimal data movement back to client

Continuous Columns

In [14]:
```
contColumns = ['a4', 'a5','a12', 'a13', 'a14', 'a16',
               'a17', 'a18', 'a19', 'a20', 'a21',
               'a22', 'a23', 'a25', 'a26', 'a27',
               'a28', 'a29', 'a30', 'a31', 'a32',
               'a33', 'a34', 'a35', 'a36', 'a37',
               'a38', 'a39', 'a40', 'a41']
sliderValue = 20
colName = contColumns[0]

def get_cont_data_frame(col, buckets):
    query = """
        WITH aggs AS (
            SELECT min({c}) AS min,
                   max({c}) AS max
              FROM training_data
        )
        SELECT width_bucket({c}, min, max, {b}-1) AS bucket,
               ('[' || min({c}) || ',' || max({c}) || ')')::text as rang
e,
               count(*) as freq
        FROM training_data, aggs
        GROUP BY bucket
        ORDER BY bucket
    """.format(c=col, b=buckets)
    cur.execute(query)

    colnames = [desc[0] for desc in cur.description]
    return pd.DataFrame(cur.fetchall(), columns=colnames)

def graph_reset():
    ipd.clear_output()
    printmd("-----\n")
    ipd.display(widgets.HBox((contDropdown,bucketsSlider)))
    printmd("-----\n")
    df = get_cont_data_frame(colName,sliderValue)
    bar_plot(df,colName,"range",colName,"freq","Frequency", "#4378E2")

def on_cont_selection(res):
    global colName
    if res['type'] == 'change' and res['name'] == 'value':
        colName = res['new']
        graph_reset()

def on_slider_selection(res):
    global sliderValue
    if res['new'] == {} and res['old']:
        sliderValue = res['old']['value']
        graph_reset()

# Look at log transforms
#colsAddLogs = contColumns + ["log({} + 1)".format(c) for c in contColum
ns]
colsAddLogs = contColumns

contDropdown = widgets.Dropdown(
    options=colsAddLogs,
```

```
    value=colsAddLogs[0],
    description='Column:',
    disabled=False,
)

bucketsSlider = widgets.IntSlider(
    value=sliderValue,
    min=5,
    max=50,
    step=1,
    description='# Buckets:',
    disabled=False,
    continuous_update=False,
    orientation='horizontal',
    readout=True,
    readout_format='d'
)

contDropdown.observe(on_cont_selection)
bucketsSlider.observe(on_slider_selection)

graph_reset()
```



- Consider variable transformation if test non-tree based algorithm
- Histogram values are being calculated in the database - minimal data movement back to client

Correlation Testing

https://madlib.apache.org/docs/latest/group__grp__correlation.html
(https://madlib.apache.org/docs/latest/group__grp__correlation.html)

In [23]:
```
%%sql
DROP TABLE IF EXISTS public.feature_correlations, public.feature_correla
tions_summary;
SELECT madlib.correlation(
        'public.training_data',
        'public.feature_correlations'
);
```

```
Done.
1 rows affected.
```

Out[23]:

id,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18,a19,a20,a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,a
Total run time = ('pub

In [24]:
```
corr = %sql SELECT * FROM public.feature_correlations ORDER BY column_po
sition;
corr = corr.DataFrame()
corr.drop('column_position', 'columns', inplace=True)
corr.set_index('variable', True, False, True)
corr
```

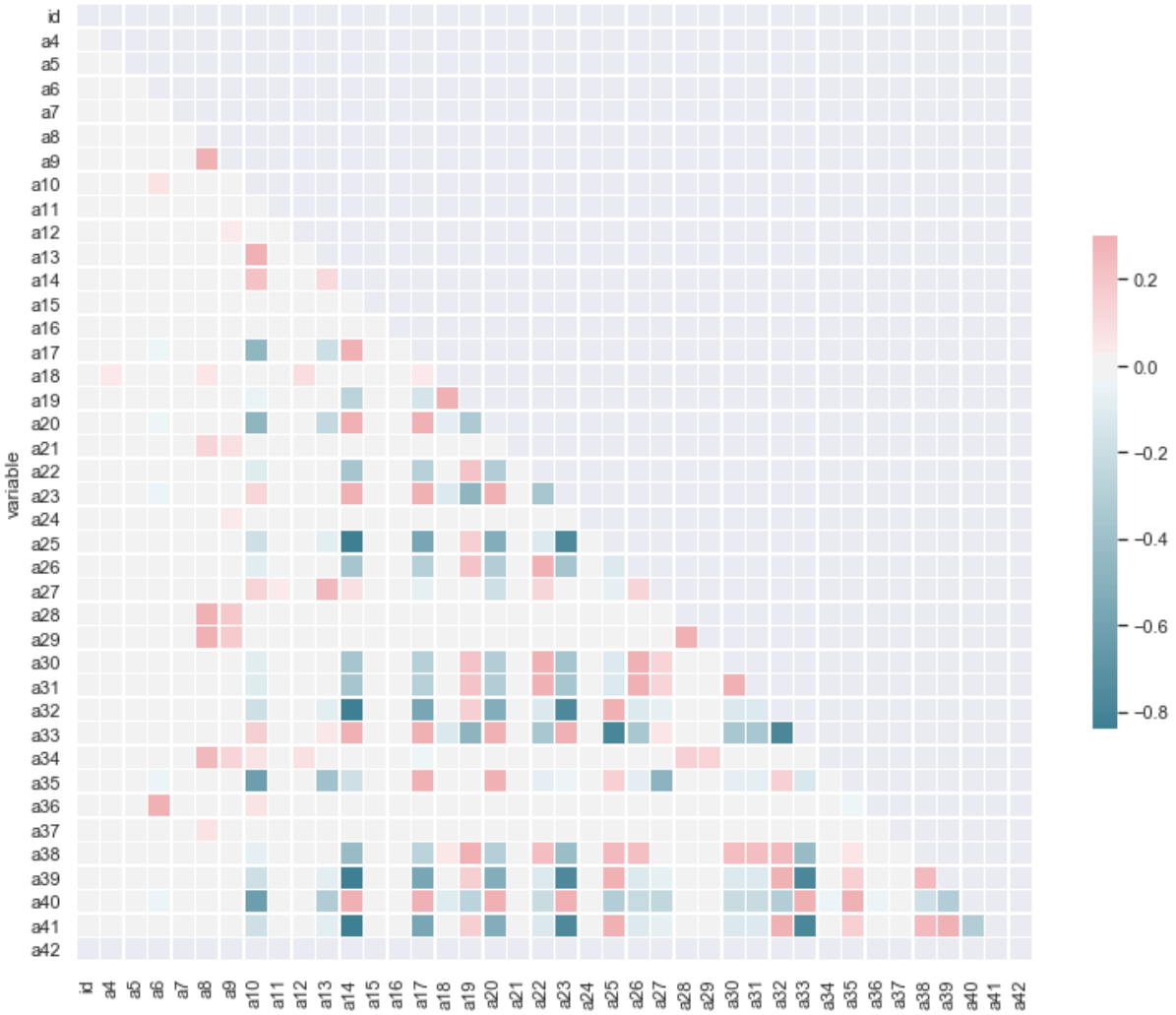
```
40 rows affected.
```

Out[24]:

variable	id	a4	a5	a6	a7	a8	a9	
id	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	
a4	-0.000906	1.000000e+00	NaN	NaN	NaN	NaN	NaN	
a5	0.000414	2.393421e-04	1.000000	NaN	NaN	NaN	NaN	
a6	-0.000261	-3.600888e-05	0.000035	1.000000	NaN	NaN	NaN	
a7	0.000551	-1.082987e-06	0.000004	-0.000018	1.000000e+00	NaN	NaN	
a8	0.000057	-4.110433e-06	0.001204	-0.000131	-2.905093e-06	1.000000	NaN	
a9	0.000189	-5.598723e-06	0.000988	-0.000239	-5.276502e-06	0.455745	1.000000	
a10	0.000052	2.009464e-04	0.002119	0.070524	1.560893e-03	0.011106	0.020172	1.00
a11	0.000436	-4.652790e-06	-0.000004	-0.000069	-1.527701e-06	-0.000011	-0.000020	-0.00
a12	0.000043	3.689192e-05	0.000256	0.001374	-6.116524e-06	0.027821	0.034960	0.02
a13	-0.000568	-1.416911e-04	0.000311	-0.003416	-1.284709e-04	-0.000195	0.000561	0.33
a14	0.000168	6.682568e-04	0.000910	0.013920	-6.541574e-05	0.002389	0.004378	0.21
a15	0.000491	-2.710890e-05	-0.000026	-0.000438	-9.673614e-06	-0.000069	-0.000125	-0.00
a16	-0.000488	2.636534e-04	0.000285	-0.000067	-8.255468e-05	0.000094	0.000241	0.00
a17	0.000154	-7.953748e-04	-0.000558	-0.034743	-8.037939e-04	-0.005091	-0.008877	-0.46
a18	-0.000245	4.122073e-02	0.020392	0.002389	-2.106206e-05	0.052088	0.026378	-0.02
a19	-0.000355	7.194917e-04	0.003306	0.012627	1.584927e-03	0.003444	0.000775	-0.05
a20	0.000531	-1.145882e-03	-0.001983	-0.034332	-7.588268e-04	-0.005398	-0.009475	-0.46
a21	-0.000562	-8.382461e-08	0.000165	-0.000032	-7.051035e-07	0.133023	0.089084	0.00
a22	0.000127	3.090613e-03	0.002313	-0.007417	-1.671306e-04	-0.000671	-0.001494	-0.10
a23	0.000531	-1.716691e-03	-0.001067	-0.042693	-1.084121e-03	-0.006898	-0.002702	0.12
a24	0.000029	2.163766e-05	-0.000011	-0.000246	-5.434247e-06	0.002852	0.036714	0.02
a25	-0.000309	-7.995773e-04	-0.000749	-0.012967	2.000839e-04	-0.000305	-0.002911	-0.18
a26	0.000045	-1.949155e-04	0.002345	-0.006446	-1.688235e-04	-0.000364	-0.001441	-0.09

variable	id	a4	a5	a6	a7	a8	a9	
a27	-0.000724	5.183435e-06	0.000346	-0.003716	-1.001058e-04	0.006156	0.009013	0.13
a28	-0.000293	-1.988572e-06	0.001303	-0.000093	1.539340e-03	0.378266	0.187161	0.00
a29	-0.000289	4.838088e-06	0.001307	-0.000059	1.324177e-03	0.349526	0.172315	0.00
a30	0.000020	2.744574e-03	0.002354	-0.006813	-8.888032e-05	-0.000435	-0.001556	-0.09
a31	0.000083	3.016217e-03	0.002310	-0.007442	-1.672473e-04	-0.000567	-0.001447	-0.10
a32	-0.000317	-6.085324e-04	-0.000757	-0.013001	5.557775e-04	-0.000600	-0.003040	-0.18
a33	0.000384	-1.549505e-03	-0.000968	-0.033523	-1.116773e-03	-0.005393	-0.000362	0.15
a34	0.000117	-2.217914e-05	0.000352	-0.000035	-1.837455e-05	0.259489	0.132047	0.07
a35	0.000503	-2.416744e-03	-0.001534	-0.044079	2.197719e-04	-0.007813	-0.010920	-0.62
a36	0.000026	7.822762e-04	0.000126	0.803830	1.345546e-03	0.001926	0.017916	0.06
a37	0.000159	-6.910167e-06	0.000632	0.004714	-2.805703e-06	0.069186	0.023673	0.00
a38	0.000179	3.295169e-04	-0.000393	0.007031	3.700450e-03	-0.000555	-0.001371	-0.07
a39	-0.000310	-5.615874e-04	-0.000758	-0.013010	1.323950e-04	-0.000667	-0.003076	-0.18
a40	0.000610	-1.652360e-03	-0.002623	-0.045141	-9.963451e-04	-0.007099	-0.012549	-0.62
a41	-0.000316	-6.110693e-04	-0.000747	-0.013078	7.002152e-04	-0.000134	-0.002774	-0.18
a42	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [25]:
```
# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
#f, ax = plt.subplots(figsize=(11, 9))
f, ax = plt.subplots(figsize=(13, 11))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=F
alse);
```



See mix of positive and negative correlations

- means complex interactions between features
- means can't take a rule-based approach or try to data engineer/reverse engineer the feature interactions
- means PCA may be a candidate to try to reduce dimensionality of problem

Clustering

Question: how well do minority classes cluster? This may tell us something about the sharpness of the decision boundary

First encode the text columns

In [ ]:
```
%%sql
DROP TABLE IF EXISTS training_data_encoded, training_data_encoded_dictio
nary;
SELECT madlib.encode_categorical_variables (
        'training_data',            -- Source table
        'training_data_encoded',      -- Output table
        'a1, a2, a3'                        -- Categorical columns
        );
```

In [49]:
```
%%sql
select * from training_data_encoded limit 5;
```

```
5 rows affected.
```

Out[49]:

a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	a17	a18	a19	a20
319.0	3229.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	360.0	0.01	0.0	0.0	15.0
520.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1324.0	1.0	0.0	0.0	434.0
520.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1473.0	1.0	0.0	0.0	454.0
1032.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1903.0	1.0	0.0	0.0	510.0
1032.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	259.0	1.0	0.0	0.0	511.0

Extract training data for minority classes with less than, say, 100 examples:

In [64]:
```
%%sql
drop table if exists training_data_work1;
create table training_data_work1 as
(select * from training_data_encoded where y='class19'
                                            or y='class12'
                                             or y='class13'
                                             or y='class09'
                                             or y='class03'
                                             or y='class08'
                                             or y='class16'
                                             or y='class05'
                                             or y='class22'
                                             or y='class07'
                                             or y='class02'
                                             or y='class04');
```

```
Done.
179 rows affected.
```

Out[64]: []

Prepare for clustering:

In [66]:
```
%%sql
drop table if exists training_data_work2, training_data_work2_summary;
select madlib.cols2vec('training_data_work1',
                       'training_data_work2',
                       '*',
                       'y',
                       'y'
                       );
```

```
Done.
1 rows affected.
```

Out[66]:

cols2vec

In [67]:
```
%%sql
ALTER TABLE training_data_work2 ADD column pid serial;
```

```
Done.
```

Out[67]: []

Call kmeans++ clustering with 15 centroids, which is the number of classes that have less than 100 examples:

In [77]:
```
%%sql
DROP TABLE IF EXISTS training_data_work3;

CREATE TABLE training_data_work3 AS
(SELECT * FROM madlib.kmeanspp(
    'training_data_work2',                  -- points table
    'feature_vector',                      -- column name in point table
    15,
    'madlib.dist_norm1',   -- distance function
    'madlib.avg',                   -- aggregate function
    20,                             -- max iterations
    0.001                          -- minimum fraction of centroids reass
igned to continue iterating
));
```

```
Done.
1 rows affected.
```

Out[77]:
```
[]
```

In [15]:
```
%%sql
select cluster_variance, objective_fn, frac_reassigned, num_iterations f
rom training_data_work3;
```

```
1 rows affected.
```

Out[15]:

cluster_variance	objective_fn	frac_reassigned	num_iterations
[15794.6516666667, 3145.33333333333, 0.0, 23800.7533333333, 0.0, 0.0, 0.0, 8784.16666666915, 6645.45, 32274.5410169492, 12993.9846153846, 0.0, 24033.3133333333, 32017.36, 7709.195]	167198.748966	0.0	6

So it converges in 6 iterations (pretty quick). Let's look at mapping of points to clusters but class:

In [17]:
```
%%sql
DROP TABLE IF EXISTS point_cluster_map;
CREATE TABLE point_cluster_map AS
SELECT data.*, (madlib.closest_column(centroids, feature_vector, 'madli
b.squared_dist_norm2')).*
FROM training_data_work2 as data, training_data_work3;
ALTER TABLE point_cluster_map RENAME column_id to cluster_id; -- change
 column name
SELECT y, pid, cluster_id, distance FROM point_cluster_map ORDER BY y, d
istance desc;
```

```
Done.
179 rows affected.
Done.
179 rows affected.
```

Out[17]:

	y	pid	cluster_id	distance
	class02	17	13	80066368.3904
	class02	164	0	3541781.34629
	class02	146	14	1162567.47512
	class02	178	14	1057157.68512
	class02	50	14	840084.560966
	class02	97	0	826395.679621
	class02	156	8	790317.402777
	class02	143	10	591068.118111
	class02	28	0	571682.008921
	class02	71	14	547151.315616
	class02	35	14	509772.225116
	class02	39	14	508212.935116
	class02	101	0	436604.342254
	class02	81	8	428263.330217
	class02	121	0	360492.846287
	class02	24	14	331115.675616
	class02	119	3	315612.257602
	class02	22	8	168780.548337
	class02	139	10	142488.066296
	class02	116	0	116908.175587
	class02	6	8	104659.263217
	class02	112	0	93380.0129542
	class02	154	10	85657.2988805
	class02	77	8	66884.930217
	class02	75	8	56973.856217
	class02	99	14	43841.2004156
	class02	158	8	23743.754897
	class02	111	8	7906.354717
	class02	107	8	5670.030057
	class02	69	6	0.0
	class03	20	12	98446937.1023
	class03	171	10	732082.796573
	class03	166	3	511925.944464
	class03	2	9	398933.411413
	class03	84	3	152978.613158
	class03	170	9	115296.630227

class03	114	9	93671.8125522
class03	34	9	27822.6999217
class04	130	9	1100587.53498
class04	163	9	509820.212944
class04	3	9	448517.726267
class04	15	9	320515.003956
class04	123	3	305536.315124
class04	96	3	292535.265549
class04	103	3	273305.826235
class04	127	3	184128.509993
class04	51	3	174362.867086
class04	38	3	167971.468689
class04	128	10	159708.878511
class04	137	9	152888.920803
class04	157	9	138288.187527
class04	30	3	127495.679824
class04	85	9	112905.944057
class04	179	9	89604.3784302
class04	7	9	89480.3934725
class04	8	3	84954.5256842
class04	76	3	79623.1140465
class04	100	3	75762.103462
class04	83	3	75004.3529909
class04	19	9	74106.2510319
class04	86	3	71441.7307687
class04	23	9	70828.9885997
class04	49	3	70196.9812109
class04	32	9	69704.527681
class04	10	3	68976.2023665
class04	26	3	61254.258082
class04	25	9	48708.8051929
class04	131	3	47622.7892198
class04	4	9	46715.8002522
class04	90	9	44062.1102776
class04	82	9	35143.0246997
class04	62	9	34952.9547014
class04	133	9	32727.3226132

class04	153	9	32379.8764844
class04	16	3	28266.4520665
class04	176	3	26817.2812398
class04	147	3	21237.6877709
class04	14	3	17025.8145731
class04	126	9	15051.1072437
class04	117	9	11853.483159
class04	140	3	10950.7575776
class04	9	9	8694.06837593
class04	78	9	7662.85109966
class04	173	9	6408.91583355
class04	172	3	5635.77012869
class04	161	9	5202.45789288
class04	57	9	5129.39347254
class04	149	9	4522.63203864
class04	89	3	4157.1892198
class04	144	3	1053.72875758
class04	105	3	894.226253136
class05	125	3	1438258.83467
class05	120	13	473387.371069
class05	94	9	252347.810642
class05	27	3	183476.563818
class05	44	3	133095.459626
class05	95	9	119594.120025
class05	29	9	86624.9082861
class05	66	3	75903.6158176
class05	33	9	58177.183498
class05	21	9	43973.0558302
class05	141	9	40113.9766675
class05	54	5	0.0
class07	160	9	655396.49582
class07	87	3	225307.772269
class07	72	9	192845.242769
class07	104	3	167002.654282
class07	138	9	145682.727684
class07	175	3	137888.084591
class07	152	9	129814.50243

class07	136	9	97626.8463285
class07	88	3	85828.6509354
class07	134	9	84221.0294539
class07	11	3	69028.3242687
class07	48	3	65984.8137087
class07	145	3	59329.1620465
class07	46	3	59171.6729909
class07	65	3	54212.1919131
class07	93	3	53380.1978242
class07	31	9	48999.6156505
class07	53	9	40041.0194047
class07	174	9	31374.8737861
class07	113	9	28648.2676793
class07	13	9	28243.0029471
class08	167	9	1151342.81951
class08	43	13	1007360.27929
class08	91	10	573105.31888
class08	148	10	539632.965034
class08	165	0	438240.144454
class08	41	10	437567.490327
class08	108	3	292574.83538
class08	5	0	179139.382504
class08	70	10	118438.602073
class09	132	12	19197859.2789
class09	110	3	624011.23538
class09	73	13	473893.945958
class09	59	9	259906.231132
class09	142	3	174939.534369
class09	155	2	0.0
class09	67	4	0.0
class12	150	10	695877.374819
class12	80	10	547615.740196
class12	36	10	259164.797204
class13	177	13	3481635.25151
class13	1	13	3447830.81409
class13	63	13	2993114.49259
class13	18	13	2969575.4845

class16	129	12	30901833.2624
class16	115	0	3093296.86187
class16	68	13	1548677.67356
class16	151	0	292053.334121
class16	47	9	233497.449203
class16	60	9	76597.6864912
class16	79	9	70468.0253776
class16	106	9	50279.7542878
class16	124	9	38698.1217522
class16	118	9	35061.1444641
class19	159	9	691394.713284
class19	42	10	332736.13405
class22	37	0	2434722.84629
class22	122	7	1851227.77778
class22	109	1	1525286.22222
class22	74	7	906586.277778
class22	12	1	701883.888889
class22	64	7	638810.444445
class22	135	7	632929.944445
class22	162	3	600954.902046
class22	58	7	583093.444444
class22	52	8	449902.104877
class22	168	7	391944.777777
class22	56	1	313365.222222
class22	61	7	268381.611111
class22	169	9	239240.931381
class22	92	7	216626.111111
class22	45	7	173480.611111
class22	40	7	20504.7777778
class22	98	7	6303.27777781
class22	102	7	489.611111116
class22	55	11	0.0

In [90]:
```
%%sql
SELECT y, pid, cluster_id, distance FROM point_cluster_map WHERE y='clas
s04';
```

```
53 rows affected.
```

Out[90]:

	y	pid	cluster_id	distance
	class04	19	9	74106.2510319
	class04	26	3	61254.258082
	class04	30	3	127495.679824
	class04	3	9	448517.726267
	class04	4	9	46715.8002522
	class04	7	9	89480.3934725
	class04	8	3	84954.5256842
	class04	32	9	69704.527681
	class04	38	3	167971.468689
	class04	51	3	174362.867086
	class04	9	9	8694.06837593
	class04	10	3	68976.2023665
	class04	14	3	17025.8145731
	class04	15	9	320515.003956
	class04	16	3	28266.4520665
	class04	23	9	70828.9885997
	class04	25	9	48708.8051929
	class04	62	9	34952.9547014
	class04	76	3	79623.1140465
	class04	82	9	35143.0246997
	class04	89	3	4157.1892198
	class04	103	3	273305.826235
	class04	105	3	894.226253136
	class04	49	3	70196.9812109
	class04	57	9	5129.39347254
	class04	78	9	7662.85109966
	class04	83	3	75004.3529909
	class04	117	9	11853.483159
	class04	123	3	305536.315124
	class04	85	9	112905.944057
	class04	86	3	71441.7307687
	class04	90	9	44062.1102776
	class04	96	3	292535.265549
	class04	100	3	75762.103462
	class04	127	3	184128.509993
	class04	131	3	47622.7892198

class04	133	9	32727.3226132
class04	137	9	152888.920803
class04	163	9	509820.212944
class04	126	9	15051.1072437
class04	128	10	159708.878511
class04	172	3	5635.77012869
class04	173	9	6408.91583355
class04	176	3	26817.2812398
class04	179	9	89604.3784302
class04	130	9	1100587.53498
class04	140	3	10950.7575776
class04	144	3	1053.72875758
class04	147	3	21237.6877709
class04	149	9	4522.63203864
class04	153	9	32379.8764844
class04	157	9	138288.187527
class04	161	9	5202.45789288

Not seeing a super clear decision boundary since given classes are mapping to multiple centroids. Probably need to normalize features since they have very different ranges in order to do this properly.

Check sihlhouette value:

In [91]:
```
%%sql
SELECT * FROM madlib.simple_silhouette( 'training_data_work2',
 -- Input points table
                                                'feature_vector',
-- Points column in input table
                                                (select centroids from tra
ining_data_work3),           -- Column in centroids table containing cen
troids
                                                'madlib.squared_dist_norm
2'   -- Distance function
                                            );
```

```
1 rows affected.
```

Out[91]:

simple_silhouette
0.818159540208

This is a pretty low silhouette value, so not a great decision boudary. Let's look at silhouette by point:

In [92]:
```
%%sql
DROP TABLE IF EXISTS km_points_silh;
SELECT * FROM madlib.simple_silhouette_points( 'training_data_work2',
-- Input points table
                                               'km_points_silh',      --
 Output table
                                               'pid',                 --
 Point ID column in input table
                                               'feature_vector',
-- Points column in input table
                                               'training_data_work3',
-- Centroids table
                                               'centroids',           --
 Column in centroids table containing centroids
                                               'madlib.squared_dist_norm
2'   -- Distance function
                                           );
SELECT * FROM km_points_silh ORDER BY centroid_id;
```

```
Done.
1 rows affected.
179 rows affected.
```

Out[92]:

pid	centroid_id	neighbor_centroid_id	silh
165	0	14	0.935808256274
101	0	14	0.93625464132
121	0	14	0.961721329119
97	0	14	0.886829404236
37	0	13	0.631878539057
151	0	14	0.96597102327
5	0	14	0.981133477231
115	0	8	0.0533889996169
28	0	14	0.936659712261
116	0	14	0.987592846725
164	0	14	0.563960453975
112	0	14	0.989223745617
12	1	7	0.908629047708
56	1	7	0.965324555962
109	1	7	0.931354905229
155	2	11	1.0
125	3	9	0.456257974465
51	3	9	0.169233121146
49	3	9	0.945042006946
27	3	9	0.875607862266
87	3	9	0.0226586445892
103	3	9	0.858241432486
175	3	9	0.611939302388
83	3	9	0.789954163633
119	3	9	0.781116483353
65	3	9	0.935156569057
123	3	9	0.848145882248
145	3	9	0.935441571694
147	3	9	0.959581383253
11	3	9	0.879767856284
127	3	9	0.890129505526
131	3	9	0.889167849502
105	3	9	0.998790324126
93	3	9	0.928706120481
89	3	9	0.993949061371
142	3	9	0.853312710236

110	3	9	0.678144421048
84	3	9	0.660789282205
76	3	9	0.796116657838
86	3	9	0.944135479978
88	3	9	0.826885013155
66	3	9	0.904283374477
176	3	9	0.946486690137
96	3	9	0.852108894436
172	3	9	0.991228151725
100	3	9	0.941679555819
166	3	9	0.766833134909
104	3	9	0.451978223416
48	3	9	0.888507795269
108	3	9	0.722195588166
46	3	9	0.935480452168
44	3	9	0.671792038605
38	3	9	0.209801660831
8	3	9	0.936401167169
30	3	9	0.529883009128
16	3	9	0.942914455433
14	3	9	0.982734528476
10	3	9	0.813040686399
26	3	9	0.842060427617
144	3	9	0.998638419267
140	3	9	0.981383826671
162	3	9	0.226853637986
67	4	5	1.0
54	5	4	1.0
69	6	12	1.0
45	7	1	0.981760107023
135	7	1	0.922261037626
61	7	1	0.977541257494
102	7	1	0.999959525809
98	7	1	0.999464472901
92	7	1	0.982635512602
168	7	1	0.976154780416
74	7	1	0.922219957461

64	7	1	0.942272818685
58	7	1	0.95029286846
40	7	1	0.998224155849
122	7	1	0.919167134734
156	8	10	0.590586986887
158	8	14	0.990440297162
52	8	10	0.574635453916
6	8	14	0.953939421749
22	8	14	0.914167831513
75	8	14	0.972603085203
77	8	14	0.970761004966
107	8	14	0.997705761441
81	8	14	0.800338263684
111	8	14	0.996705399629
79	9	3	0.886813411447
53	9	3	0.935718363641
21	9	3	0.953078822192
19	9	3	0.796800768985
161	9	3	0.993273499725
159	9	3	0.296928045246
157	9	3	0.908819238326
153	9	3	0.934039713218
149	9	3	0.993703986806
141	9	3	0.95348827964
85	9	3	0.609855367426
59	9	3	0.217110982292
57	9	3	0.992684454911
47	9	3	0.27576005115
33	9	3	0.913758617495
31	9	3	0.9142347384
29	9	3	0.927356736414
25	9	3	0.957919366172
23	9	3	0.809639344968
15	9	3	0.842461600973
13	9	3	0.963658195144
9	9	3	0.989932915948
7	9	3	0.728570163203

3	9	3	0.808693137334
179	9	3	0.73022287036
173	9	3	0.990500002267
169	9	3	0.0596669341311
167	9	10	0.646004634942
163	9	3	0.79479606889
137	9	3	0.34297870699
133	9	3	0.932717389759
117	9	3	0.980522768381
113	9	3	0.962145539413
95	9	3	0.913528417893
114	9	3	0.927884904466
106	9	3	0.941598573002
170	9	3	0.888789170611
34	9	3	0.967691239785
32	9	3	0.812476801711
124	9	3	0.937483813595
126	9	3	0.983938625716
136	9	3	0.76425560899
138	9	3	0.90365387684
134	9	3	0.820188677538
130	9	3	0.69899935564
160	9	3	0.765796066841
152	9	3	0.630342923769
78	9	3	0.991002755392
82	9	3	0.926992353063
2	9	3	0.500306895413
72	9	3	0.268769269845
90	9	3	0.90107169752
174	9	3	0.965873820871
62	9	3	0.927429054928
60	9	3	0.887869865315
4	9	3	0.892587685052
94	9	3	0.86326781737
118	9	3	0.946513279748
70	10	8	0.956707890382
150	10	8	0.628416893535

42	10	9	0.845344961622
36	10	8	0.853665598597
80	10	8	0.691812956729
154	10	8	0.97133675931
148	10	3	0.59288369722
128	10	8	0.933594509808
91	10	9	0.562842837393
139	10	8	0.954308553244
171	10	8	0.682480843222
41	10	3	0.684946497101
143	10	9	0.608601364911
55	11	2	1.0
129	12	13	0.854878504143
20	12	13	0.890192321999
132	12	13	0.922830705682
18	13	0	0.537046645618
120	13	0	0.977480253272
68	13	0	0.799350132586
63	13	0	0.530013909127
1	13	0	0.4885040702
177	13	0	0.504040227792
43	13	0	0.904620210277
17	13	0	0.504370214577
73	13	0	0.962159315565
99	14	8	0.978954500426
39	14	8	0.772704403851
35	14	8	0.876110468883
71	14	8	0.811398595337
178	14	8	0.665921108206
24	14	8	0.889975291064
146	14	8	0.709497094899
50	14	8	0.797274623372

Try different numbers of centroids:

In [19]:
```
%%sql
DROP TABLE IF EXISTS k_auto, k_auto_summary;

SELECT madlib.kmeanspp_auto(
    'training_data_work2',                  -- points table
    'k_auto',                          -- output table
    'feature_vector',                       -- column name in point table
    ARRAY[12, 13, 14, 15, 16, 17, 18],              -- k values to try
    'madlib.squared_dist_norm2',    -- distance function
    'madlib.avg',                   -- aggregate function
    20,                             -- max iterations
    0.001,                          -- minimum fraction of centroids reas
signed to continue iterating
    1.0,                            -- centroid seed
    'both'                          -- k selection algorithm (silhouette
 or elbow or both)
);
```

```
Done.
1 rows affected.
```

Out[19]:

kmeanspp_auto

In [22]:
```
%%sql
SELECT cluster_variance, objective_fn, frac_reassigned, num_iterations,
 silhouette, elbow, selection_algorithm FROM k_auto_summary;
```

```
1 rows affected.
```

Out[22]:

cluster_variance	objective_fn	frac_reassigned	num_iterations	silhouette	elbow
[36094131.4341202, 37349426.5333333, 0.0, 0.0, 0.0, 0.0, 0.0, 38483830.7682562, 7861154.54450667, 0.0, 33259529.652835, 57617814.6536667]	210665887.587	0.0	6	0.86935801181	31936211.4258

In [23]:
```
%%sql
SELECT cluster_variance, objective_fn, frac_reassigned, num_iterations,
 silhouette, elbow FROM k_auto ORDER BY k;
```

```
7 rows affected.
```

Out[23]:

cluster_variance	objective_fn	frac_reassigned	num_iterations	silhouette	elbo
[36094131.4341202, 37349426.5333333, 0.0, 0.0, 0.0, 0.0, 0.0, 38483830.7682562, 7861154.54450667, 0.0, 33259529.652835, 57617814.6536667]	210665887.587	0.0	6	0.86935801181	31936211.42
[13547850.1317583, 37349426.5333333, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 10595018.3640367, 876223.99025, 52995357.0571684, 45596303.9899481, 0.0]	160960180.066	0.0	8	0.798808476983	10713657.26
[31580457.7297298, 37349426.5333333, 0.0, 0.0, 0.0, 0.0, 0.0, 82739299.8503419, 10617528.3301614, 0.0, 0.0, 0.0, 12840182.95434, 0.0]	175126895.398	0.0	6	0.754280855921	-5807471.022
[8999116.26861961, 5690378.66666667, 0.0, 0.0, 0.0, 0.0, 0.0, 7883394.42190189, 0.0, 52995357.0571684, 0.0, 17557233.8039177, 2540535.33333333, 876223.99025, 7861154.54450667]	104403394.086	0.0	3	0.813210035641	9322629.929
[349817.306933333, 37349426.5333333, 0.0, 0.0, 0.0, 0.0, 2913377.04604444, 0.0, 876223.99025, 45596303.9899481, 0.0, 7983310.83413192, 13314059.6529424, 1588320.5, 0.0, 6387179.263775]	116358019.117	0.0	7	0.788313135166	19175850.03

[5530556.05006, 37349426.5333333, 0.0, 0.0, 0.0, 0.0, 0.0, 12840182.95434, 0.0, 6196840.223915, 1467675.53907692, 876223.99025, 0.0, 7861154.54450667, 8558832.19872308, 4312149.24188627, 144086.548866667]	85137127.825	0.0	6	0.801506774799	29394274.14
[2292795.37918788, 37349426.5333333, 0.0, 0.0, 0.0, 0.0, 57617814.6536667, 0.0, 12840182.95434, 120819.4565, 626350.292509091, 4999903.073075, 2103101.57553, 0.0, 2737496.38614667, 2858953.04728837, 5530556.05006, 5215143.58095385]	134292542.983	0.0	7	0.78306291267	40188153.2

Plot the silhouette profiles

In [98]:
```
# get range of k values tested
k_range = %sql SELECT k FROM k_auto ORDER BY k;

# outer loop on k
# plot clusters for each k value
for n_clusters in k_range:

    # create table mapping each point to its centroid
    kval = n_clusters[0]
    %sql DROP TABLE IF EXISTS k_plot1;
    %sql CREATE TABLE k_plot1 AS (SELECT data.*, (madlib.closest_column
(centroids, feature_vector, 'madlib.squared_dist_norm2')).column_id as c
luster_id FROM training_data_work2 as data, k_auto WHERE k=$kval);

    # get info from tables and reshape to np arrays
    # number of points
    num_points_proxy= %sql SELECT COUNT(*) FROM k_plot1;
    num_points= num_points_proxy[0][0]

    # points
    points_proxy = %sql SELECT feature_vector FROM k_plot1 ORDER BY pid;
    points = np.array(points_proxy).reshape(num_points,123)

    # cluster id
    cluster_id_proxy = %sql SELECT cluster_id FROM k_plot1 ORDER BY pid;
    cluster_id = np.array(cluster_id_proxy).reshape(num_points)

    # centroids
    centroids_proxy = %sql SELECT centroids FROM k_auto WHERE k=$kval;
    centers = np.array(centroids_proxy[0][0]).reshape(kval,123)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the fo
rmed
    # clusters
    silhouette_avg = %sql SELECT silhouette FROM k_auto WHERE k=$kval;
    print("For n_clusters =", kval,
          "The average silhouette_score is :", silhouette_avg)

    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
    # The silhouette coefficient can range from -1, 1 but in this exampl
e all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhoue
tte
    # plots of individual clusters, to demarcate them clearly.
    len_X = %sql select count(*) from training_data_work2;
    len_X = len_X[0][0]
    ax1.set_ylim([0, len_X + (kval + 1) * 10])

    y_lower = 10
```

```
    # inner loop on number of centroids
    for i in range(kval):

        %sql DROP TABLE IF EXISTS points_distr1;
        %sql SELECT * FROM madlib.simple_silhouette_points( 'training_da
ta_work2', 'points_distr1', 'pid', 'feature_vector', (SELECT centroids F
ROM k_auto WHERE k=$kval), 'madlib.squared_dist_norm2');
        ith_cluster_silhouette_values_proxy = %sql SELECT silh from poin
ts_distr1 WHERE centroid_id=$i ORDER BY silh;
        ith_cluster_silhouette_values = np.array(ith_cluster_silhouette_
values_proxy).reshape(len(ith_cluster_silhouette_values_proxy))

        size_cluster_i_proxy = %sql SELECT COUNT(*) from points_distr1 W
HERE centroid_id=$i;
        size_cluster_i = size_cluster_i_proxy[0][0]

        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / kval)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                  0, ith_cluster_silhouette_values,
                  facecolor=color, edgecolor=color, alpha=0.7);

        # Label the silhouette plots with their cluster numbers at the m
iddle
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i));

        # Compute the new y_lower for next plot
        y_lower = y_upper + 10  # 10 for the 0 samples

    ax1.set_title("Silhouette plot for the various clusters.")
    ax1.set_xlabel("Silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # The vertical line for average silhouette score of all the values
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 2nd Plot showing the actual clusters formed
    centroids = %sql SELECT centroid_id FROM points_distr1 ORDER BY pid;
    cluster_labels = np.array(centroids).reshape(len(centroids))

    colors = cm.nipy_spectral(cluster_labels.astype(float) / kval)
    #ax2.scatter(X[:, 0], X[:, 1], marker='.', s=30, lw=0, alpha=0.7,
    #            c=colors, edgecolor='k')

    # Labeling the clusters
    # Draw white circles at cluster centers
    #ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
    #            c="white", alpha=1, s=200, edgecolor='k')

    #for i, c in enumerate(centers):
    #    ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
    #                s=50, edgecolor='k')
```

```
    #ax2.set_title("Visualization of the clustered data.")
    #ax2.set_xlabel("Feature space for the 1st feature")
    #ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(("Silhouette analysis for KMeans clustering on sample d
ata "
                  "with n_clusters = %d" % kval),
                 fontsize=14, fontweight='bold')

plt.show();
```

```
7 rows affected.
Done.
179 rows affected.
1 rows affected.
179 rows affected.
179 rows affected.
1 rows affected.
1 rows affected.
('For n_clusters =', 12, 'The average silhouette_score is :', [(0.84790
4829929578,)])
1 rows affected.
Done.
1 rows affected.
15 rows affected.
1 rows affected.
Done.
1 rows affected.
15 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
10 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
26 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
1 rows affected.
Done.
1 rows affected.
```

```
104 rows affected.
1 rows affected.
179 rows affected.
Done.
179 rows affected.
1 rows affected.
179 rows affected.
179 rows affected.
1 rows affected.
1 rows affected.
('For n_clusters =', 13, 'The average silhouette_score is :', [(0.86713
9583945893,)])
1 rows affected.
Done.
1 rows affected.
108 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
12 rows affected.
1 rows affected.
Done.
1 rows affected.
16 rows affected.
1 rows affected.
```

```
Done.
1 rows affected.
12 rows affected.
1 rows affected.
Done.
1 rows affected.
19 rows affected.
1 rows affected.
179 rows affected.
Done.
179 rows affected.
1 rows affected.
179 rows affected.
179 rows affected.
1 rows affected.
1 rows affected.
('For n_clusters =', 14, 'The average silhouette_score is :', [(0.86803
1309555745,)])
1 rows affected.
Done.
1 rows affected.
20 rows affected.
1 rows affected.
Done.
1 rows affected.
12 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
2 rows affected.
1 rows affected.
Done.
1 rows affected.
104 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
```

```
15 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
1 rows affected.
Done.
1 rows affected.
16 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
179 rows affected.
Done.
179 rows affected.
1 rows affected.
179 rows affected.
179 rows affected.
1 rows affected.
1 rows affected.
('For n_clusters =', 15, 'The average silhouette_score is :', [(0.79200
9469731866,)])
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
```

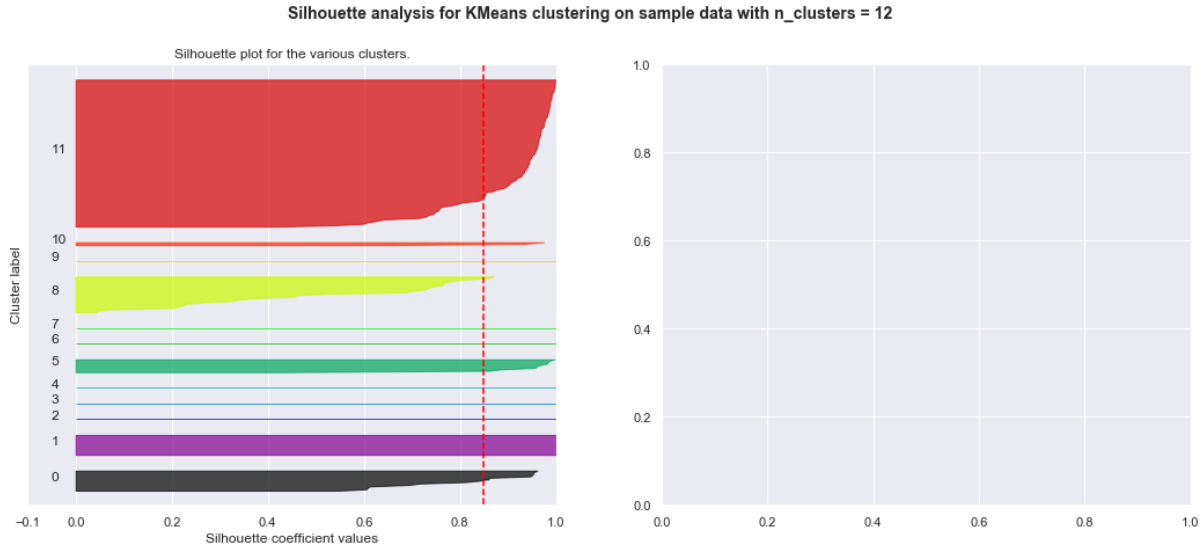
```
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
8 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
1 rows affected.
Done.
1 rows affected.
12 rows affected.
1 rows affected.
Done.
1 rows affected.
59 rows affected.
1 rows affected.
Done.
1 rows affected.
20 rows affected.
1 rows affected.
Done.
1 rows affected.
19 rows affected.
1 rows affected.
Done.
1 rows affected.
46 rows affected.
1 rows affected.
179 rows affected.
Done.
179 rows affected.
1 rows affected.
179 rows affected.
179 rows affected.
1 rows affected.
1 rows affected.
('For n_clusters =', 16, 'The average silhouette_score is :', [(0.78399
5460343156,)])
1 rows affected.
Done.
1 rows affected.
8 rows affected.
1 rows affected.
Done.
1 rows affected.
15 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
```

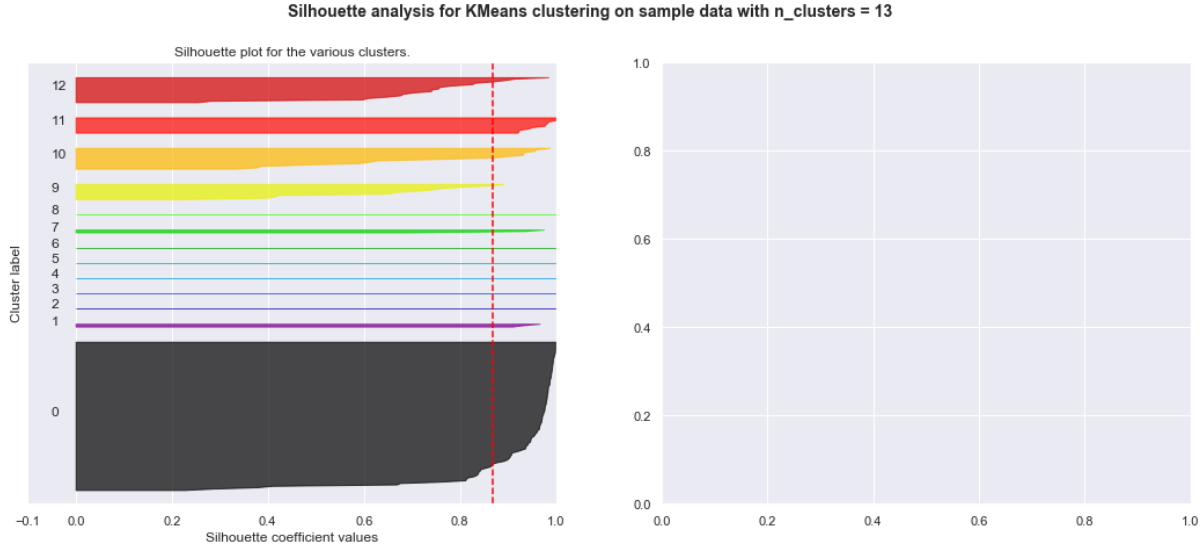
```
1 rows affected.
1 rows affected.
Done.
1 rows affected.
51 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
40 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
2 rows affected.
1 rows affected.
Done.
1 rows affected.
4 rows affected.
1 rows affected.
Done.
1 rows affected.
16 rows affected.
1 rows affected.
Done.
1 rows affected.
13 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
15 rows affected.
1 rows affected.
Done.
1 rows affected.
9 rows affected.
1 rows affected.
179 rows affected.
Done.
179 rows affected.
1 rows affected.
179 rows affected.
179 rows affected.
1 rows affected.
```

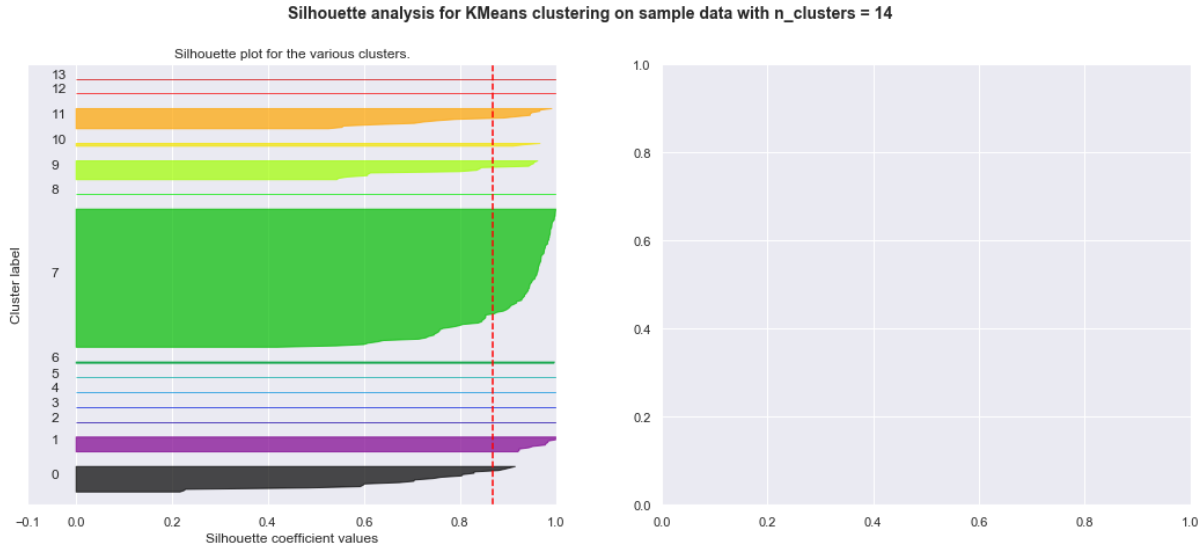
```
1 rows affected.
('For n_clusters =', 17, 'The average silhouette_score is :', [(0.82974
5366314852,)])
1 rows affected.
Done.
1 rows affected.
46 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
2 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
9 rows affected.
1 rows affected.
Done.
1 rows affected.
8 rows affected.
1 rows affected.
Done.
1 rows affected.
10 rows affected.
1 rows affected.
Done.
1 rows affected.
12 rows affected.
1 rows affected.
Done.
```

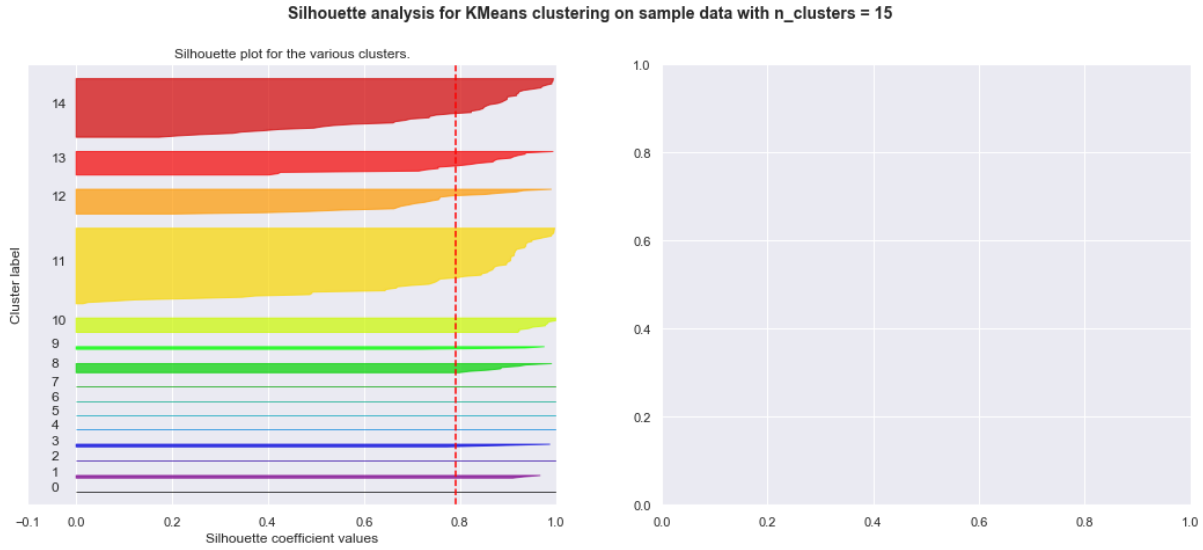
```
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
11 rows affected.
1 rows affected.
Done.
1 rows affected.
58 rows affected.
1 rows affected.
Done.
1 rows affected.
13 rows affected.
1 rows affected.
179 rows affected.
Done.
179 rows affected.
1 rows affected.
179 rows affected.
179 rows affected.
1 rows affected.
1 rows affected.
('For n_clusters =', 18, 'The average silhouette_score is :', [(0.77131
9343876484,)])
1 rows affected.
Done.
1 rows affected.
45 rows affected.
1 rows affected.
Done.
1 rows affected.
7 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
3 rows affected.
```

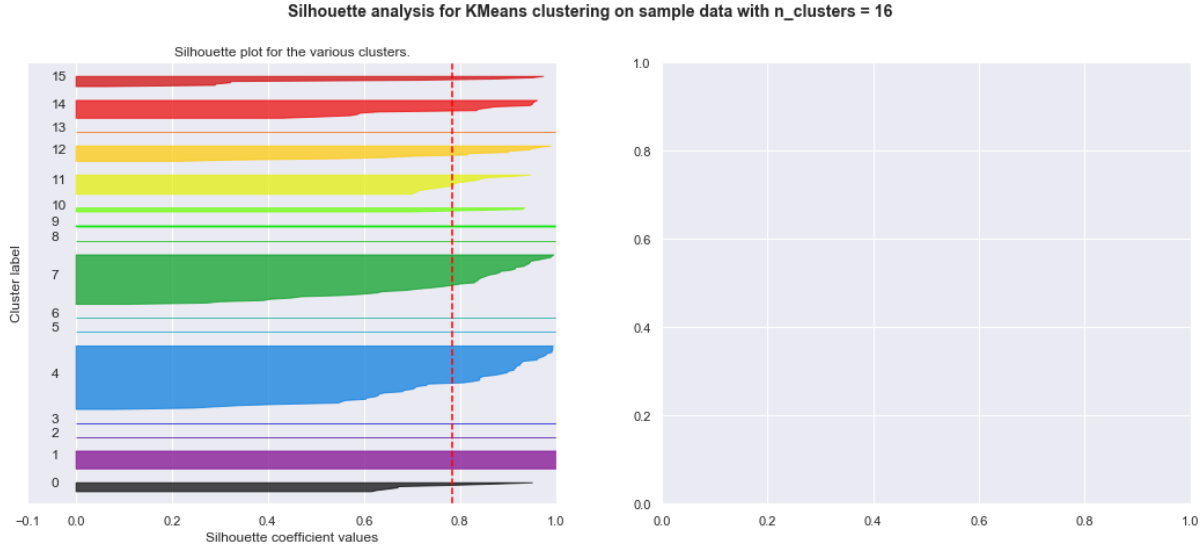
```
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
1 rows affected.
1 rows affected.
Done.
1 rows affected.
20 rows affected.
1 rows affected.
Done.
1 rows affected.
8 rows affected.
1 rows affected.
Done.
1 rows affected.
11 rows affected.
1 rows affected.
Done.
1 rows affected.
8 rows affected.
1 rows affected.
Done.
1 rows affected.
2 rows affected.
1 rows affected.
Done.
1 rows affected.
11 rows affected.
1 rows affected.
Done.
1 rows affected.
5 rows affected.
1 rows affected.
Done.
1 rows affected.
52 rows affected.
1 rows affected.
179 rows affected.
```

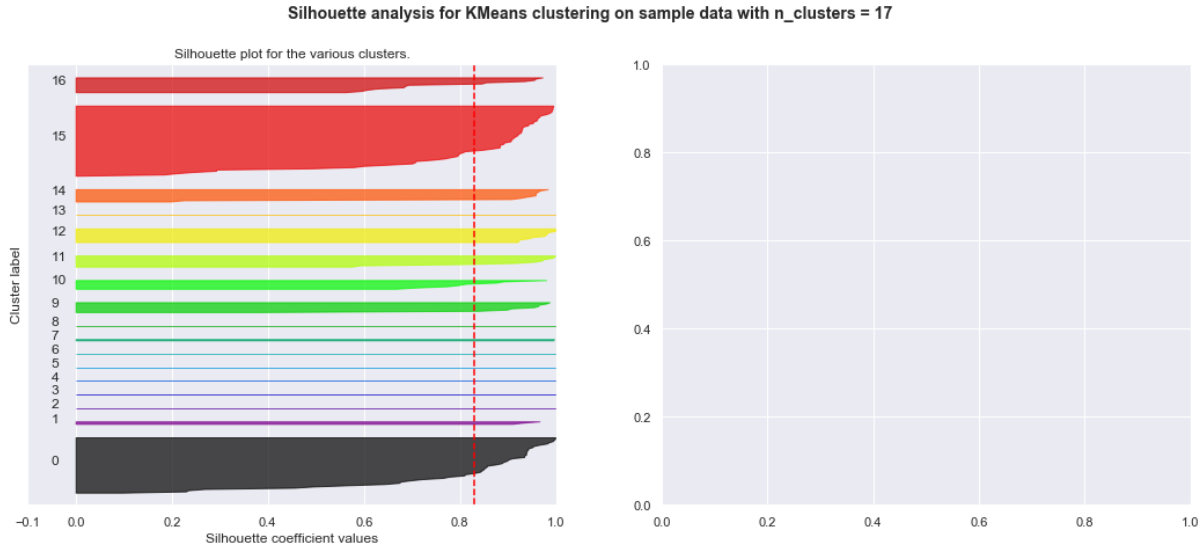

Silhouette analysis for KMeans clustering on sample data with n_clusters = 12
Silhouette plot for the various clusters.
Cluster label
Silhouette coefficient values

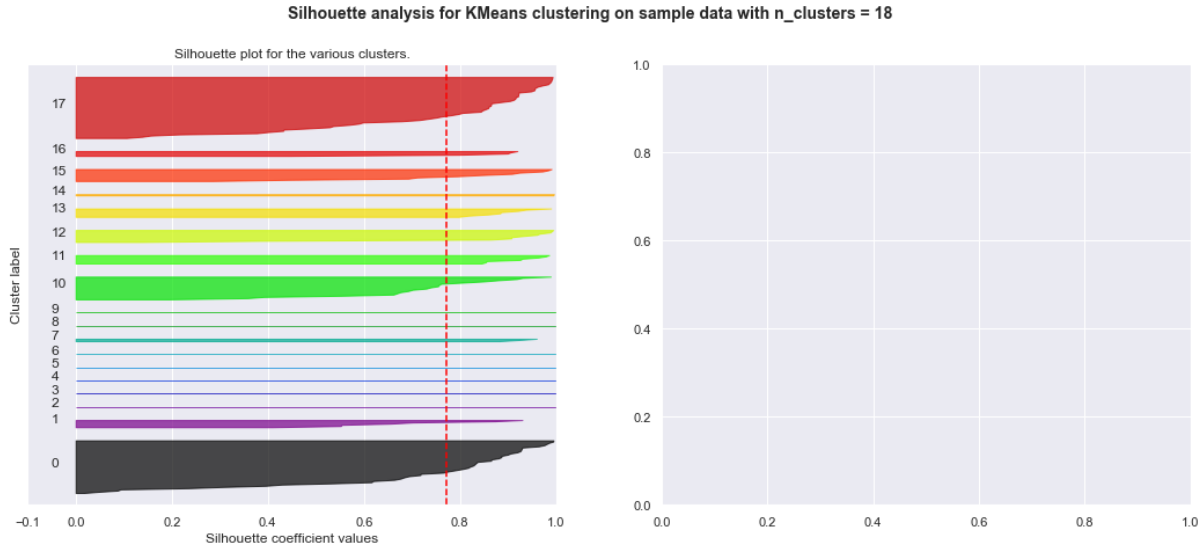

Silhouette analysis for KMeans clustering on sample data with n_clusters = 13
Silhouette plot for the various clusters.
Cluster label
Silhouette coefficient values


Silhouette analysis for KMeans clustering on sample data with n_clusters = 14
Silhouette plot for the various clusters.
Cluster label
Silhouette coefficient values


Silhouette analysis for KMeans clustering on sample data with n_clusters = 15
Silhouette plot for the various clusters.
Cluster label
Silhouette coefficient values


Silhouette analysis for KMeans clustering on sample data with n_clusters = 16
Silhouette plot for the various clusters.
Cluster label
Silhouette coefficient values

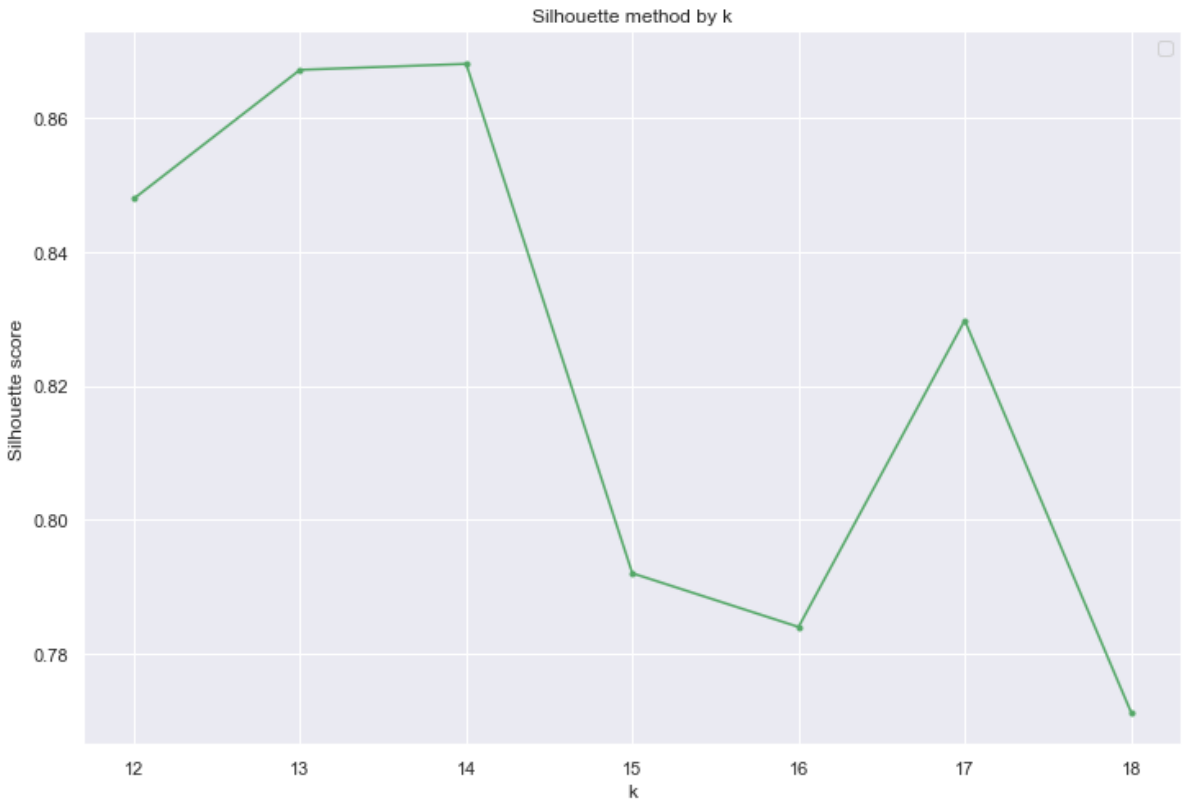

Silhouette analysis for KMeans clustering on sample data with n_clusters = 17
Silhouette plot for the various clusters.
Cluster label
Silhouette coefficient values



Profiles above show decision boundary to be overlapping and centroid=15 is not better than others really. We can aggregate the view to confirm (higher silh value is better):

In [100]:
```
# get silhouette values for each k
k = %sql SELECT k FROM k_auto ORDER BY k;
silhouette = %sql SELECT silhouette FROM k_auto ORDER BY k;

#plot
plt.title('Silhouette method by k');
plt.xlabel('k');
plt.ylabel('Silhouette score');
plt.grid(True,);
plt.plot(k, silhouette, 'g.-');
plt.legend();
```

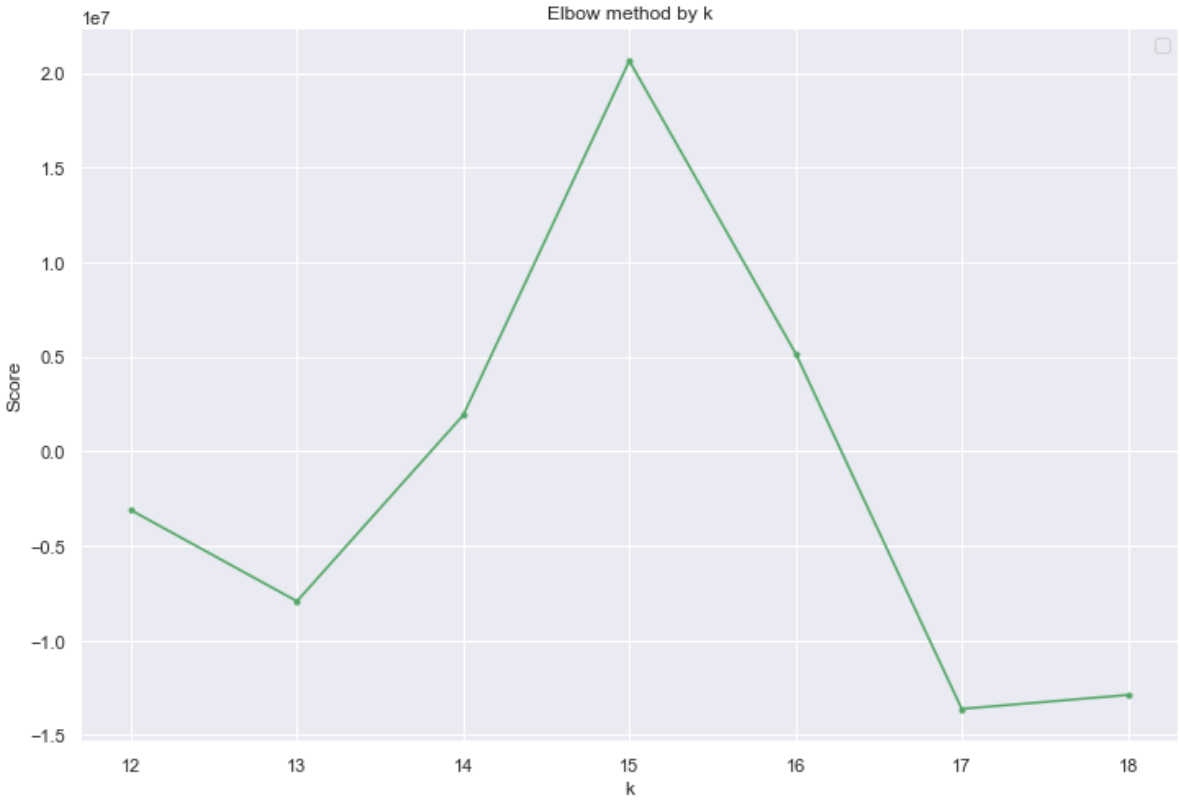
```
7 rows affected.
7 rows affected.
```



But the elbow method does show 15 to be the best number of centroids (bigger elbow value is better)

In [105]:
```
# get elbow values for each k
elbow = %sql SELECT elbow FROM k_auto ORDER BY k;

#plot
plt.title('Elbow method by k');
plt.xlabel('k');
plt.ylabel('Score');
plt.grid(True,);
plt.plot(k, elbow, 'g.-');
plt.legend();
```

```
7 rows affected.
```



Feature Engineering

We tried several variations on the sampling theme but only 1 is show below: undersample classes with more than 1000 examples down to 1000, and sample the rest to 100. Other more complex schemes like stratified sampling did not seem to yield better results.

In [24]:
```
%sql SET search_path=network_anomaly_run2,madlib
```

```
Done.
```

Out[24]:
```
[]
```

Extract classes with > 1000 examples:

In [9]:
```
%%sql
DROP TABLE IF EXISTS y;
CREATE TABLE y AS SELECT y, (COUNT(*)>=1000) AS is_top FROM public.train
ing_data GROUP BY y;
```

```
Done.
23 rows affected.
```

Out[9]:
```
[]
```

In [10]:
```
%%sql DROP TABLE IF EXISTS training_data_top;
CREATE TABLE training_data_top AS
    SELECT *
FROM public.training_data JOIN y USING(y) WHERE is_top=TRUE;
SELECT COUNT(*) FROM training_data_top
```

```
Done.
4897009 rows affected.
1 rows affected.
```

Out[10]:

count
4897009

Extract classes with < 1000 examples:

In [11]:
```
%%sql DROP TABLE IF EXISTS training_data_bottom;
CREATE TABLE training_data_bottom AS
    SELECT *
FROM public.training_data JOIN y USING(y) WHERE is_top=FALSE;
SELECT COUNT(*) FROM training_data_bottom
```

```
Done.
1422 rows affected.
1 rows affected.
```

Out[11]:

count
1422

Undersample majority 9 classes to 1000 each:

In [12]:
```
%%sql
DROP TABLE IF EXISTS training_data_top_sample2;
SELECT madlib.balance_sample(
                                'training_data_top',              -- Sourc
e table
                                'training_data_top_sample2',      -- Outpu
t table
                                'y',                  -- Class column
                                'uniform',            -- Uniform sample
                                 9000);               -- Desired output tab
le size
```

```
Done.
1 rows affected.
```

Out[12]:

balance_sample

In [13]:
```
%sql SELECT y, COUNT(*) FROM training_data_top_sample2 GROUP BY y;
```

```
9 rows affected.
```

Out[13]:

y	count
class06	1000
class11	1000
class15	1000
class17	1000
class01	1000
class10	1000
class18	1000
class21	1000
normal	1000

In [14]:
```
%%sql
DROP TABLE IF EXISTS training_data_bottom_sample2;
SELECT madlib.balance_sample(
                                'training_data_bottom',              -- So
urce table
                                'training_data_bottom_sample2',      -- Ou
tput table
                                'y',                  -- Class column
                                'uniform',            -- Uniform sample
                                 1400);               -- Desired output tab
le size
```

```
Done.
1 rows affected.
```

Out[14]:

balance_sample

In [15]:
```
%sql SELECT y, COUNT(*) FROM training_data_bottom_sample2 GROUP BY y;
```

```
14 rows affected.
```

Out[15]:

y	count
class03	100
class05	100
class07	100
class09	100
class12	100
class14	100
class16	100
class02	100
class04	100
class08	100
class13	100
class19	100
class20	100
class22	100

Now UNION them and check results:

In [16]:
```
%%sql
    DROP TABLE IF EXISTS training_data_balanced2;
    CREATE TABLE training_data_balanced2 AS
    SELECT * FROM training_data_top_sample2 UNION SELECT * FROM training
_data_bottom_sample2;
```

```
Done.
10400 rows affected.
```

Out[16]:
```
[]
```

In [18]:
```
%sql SELECT y,count(*) FROM training_data_balanced2 GROUP BY y ORDER BY
 y;
```

```
23 rows affected.
```

Out[18]:

y	count
class01	1000
class02	100
class03	100
class04	100
class05	100
class06	1000
class07	100
class08	100
class09	100
class10	1000
class11	1000
class12	100
class13	100
class14	100
class15	1000
class16	100
class17	1000
class18	1000
class19	100
class20	100
class21	1000
class22	100
normal	1000

In [19]:
```
%sql ALTER TABLE training_data_balanced2 DROP COLUMN is_top;
```

```
Done.
```

Out[19]:
```
[]
```

Create test/train split on 80/20 basis:

In [20]:
```
%%sql
DROP TABLE IF EXISTS balanced2_train, balanced2_test;
SELECT madlib.train_test_split(
                                'training_data_balanced2',    -- Source
 table
                                'balanced2',            -- Output table
                                0.8,       -- Sample proportion
                                NULL,       -- Sample proportion
                                NULL,       -- Strata definition
                                NULL,       -- Columns to output
                                FALSE,     -- Sample without replacement
                                TRUE);    -- Do not separate output tabl
es
SELECT y,COUNT(*) FROM balanced2_train GROUP BY y;
```

```
Done.
1 rows affected.
23 rows affected.
```

Out[20]:

y	count
class01	788
class03	83
class05	87
class07	83
class09	85
class10	801
class12	84
class14	83
class16	83
class18	821
class21	816
normal	804
class02	73
class04	85
class06	784
class08	85
class11	786
class13	74
class15	771
class17	820
class19	77
class20	75
class22	72

In [21]:
```
%sql SELECT y,COUNT(*) FROM balanced2_test GROUP BY y;
```

```
23 rows affected.
```

Out[21]:

y	count
class02	27
class04	15
class06	216
class08	15
class11	214
class13	26
class15	229
class17	180
class19	23
class20	25
class22	28
class01	212
class03	17
class05	13
class07	17
class09	15
class10	199
class12	16
class14	17
class16	17
class18	179
class21	184
normal	196

Train model with Random Forest

https://madlib.apache.org/docs/latest/group__grp__random__forest.html (https://madlib.apache.org/docs/latest/group__grp__random__forest.html)

In [22]:
```
%sql ALTER TABLE balanced2_train DROP COLUMN __madlib_id__;
```

```
Done.
```

Out[22]:
```
[]
```

In [23]:
```
%sql ALTER TABLE balanced2_test DROP COLUMN __madlib_id__;
```

```
Done.
```

Out[23]:
```
[]
```

In [24]:
```
%%sql
DROP TABLE IF EXISTS rf_model2, rf_model2_summary, rf_model2_group;
SELECT madlib.forest_train(
            'balanced2_train',  -- source table
            'rf_model2',              -- output table
            'id',                     -- unique row id
            'y',                      -- dependent var
            '*',                      -- indep var
            null,                     -- cols to exclude
            null,                     -- grouping
            10::integer,              -- num trees
            5::integer,               -- num random features
            true::boolean,            -- importance
            5::integer,               -- num permutations
            10::integer,              -- max tree depth
            3::integer,               -- min split
            1::integer,               -- min bucket
            10::integer,               -- num splits
            NULL,                     -- null handling
            TRUE                      -- verbose
        );
```

```
Done.
1 rows affected.
```

Out[24]:

forest_train

In [25]:
```
%%sql
    SELECT gid, sample_id
    FROM rf_model2;
```

10 rows affected.

Out[25]:

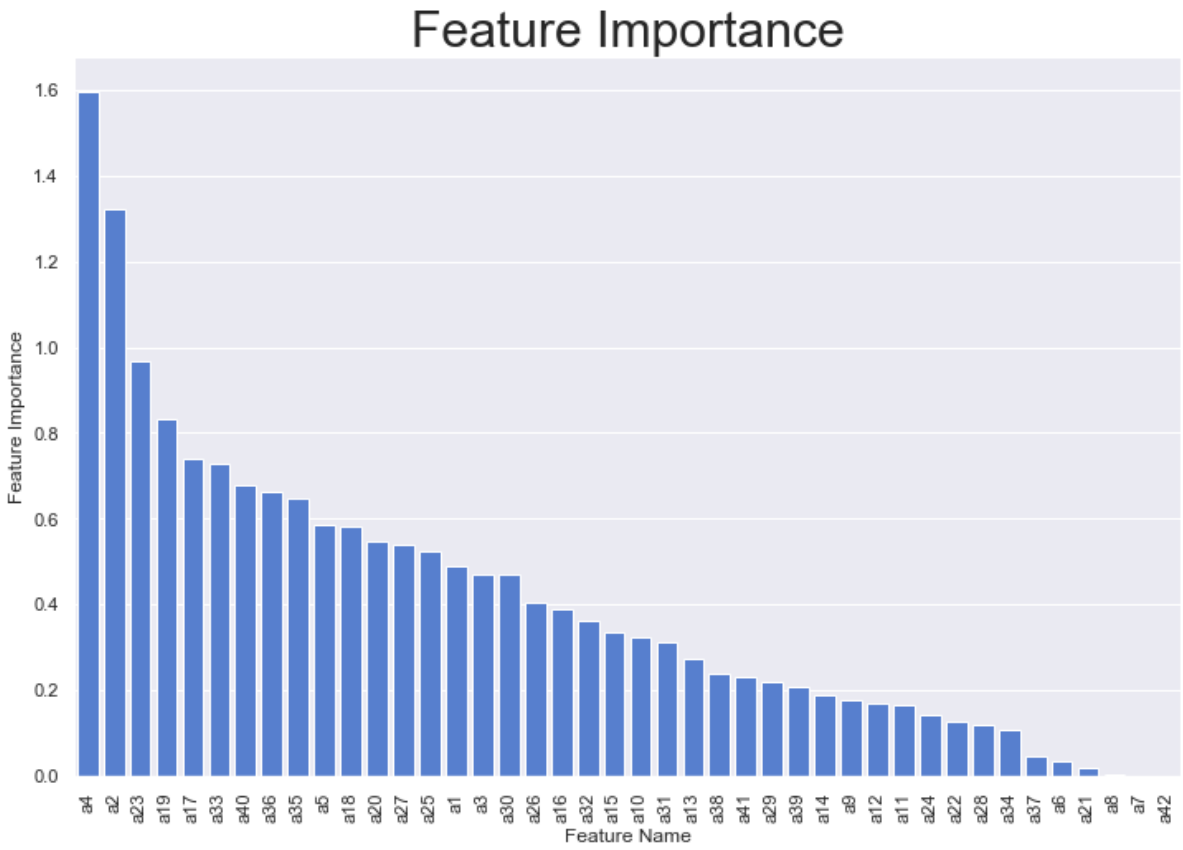
gid	sample_id
1	1
1	2
1	3
1	4
1	5
1	6
1	7
1	8
1	9
1	10

Variable Importance

Impurity importance:

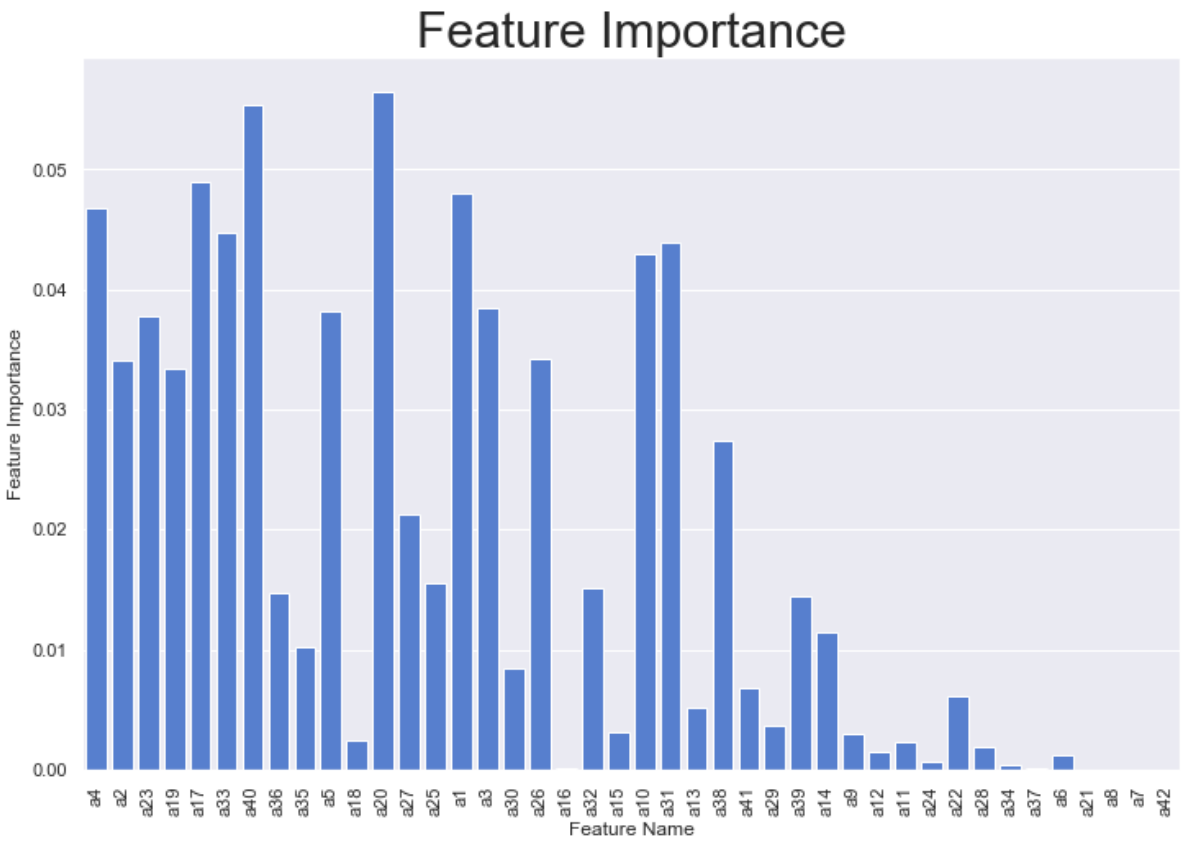
In [25]:
```
query = """SET search_path=network_anomaly_run2, madlib;
    SELECT unnest(string_to_array(independent_varnames,',')) AS feature_
name
          ,unnest(impurity_var_importance) AS impurity_feature_importanc
e
          ,unnest(oob_var_importance) AS oob_feature_importance
    FROM rf_model2_group l
        ,rf_model2_summary r
    ORDER BY 2 DESC
"""

df = query_gpdb(query)
ipd.display(df.head(10))
bar_plot(df,"Feature Importance","feature_name",'Feature Name',"impurity
_feature_importance","Feature Importance", "#4378E2")
```

	feature_name	impurity_feature_importance	oob_feature_importance
0	a4	1.597629	0.046803
1	a2	1.321560	0.034117
2	a23	0.968556	0.037731
3	a19	0.831987	0.033349
4	a17	0.741452	0.048955
5	a33	0.726509	0.044693
6	a40	0.679008	0.055408
7	a36	0.663073	0.014740
8	a35	0.648912	0.010236
9	a5	0.586486	0.038223



out of bag importance:

In [26]:
```
bar_plot(df,"Feature Importance","feature_name",'Feature Name',"oob_feat
ure_importance","Feature Importance", "#4378E2")
```



Score Validation/Training Data

https://madlib.apache.org/docs/latest/group__grp__random__forest.html (https://madlib.apache.org/docs/latest/group__grp__random__forest.html)

In [27]:
```
# Evaluate validation dataset
query = """
    SET search_path=network_anomaly_run2,madlib;
    DROP TABLE IF EXISTS rf_model2_scored;
    SELECT madlib.forest_predict('rf_model2',
                                 'balanced2_test',
                                 'rf_model2_scored',
                                 'response');

    DROP TABLE IF EXISTS rf_model2_scored_tmp;
    CREATE TABLE rf_model2_scored_tmp AS
    SELECT *
    FROM rf_model2_scored
    JOIN balanced2_test
    USING (id);
    DROP TABLE rf_model2_scored;
    ALTER TABLE rf_model2_scored_tmp RENAME TO rf_model2_scored;
    SELECT * FROM rf_model2_scored LIMIT 10;

"""
cur.execute(query)
```

In [ ]:
```
# Evaluate training dataset
query = """
    DROP TABLE IF EXISTS rf_model4_scored;
    SELECT madlib.forest_predict('rf_model4',
                                 'balanced4_train',
                                 'rf_model4_scored',
                                 'response');

    DROP TABLE IF EXISTS rf_model4_scored_tmp;
    CREATE TABLE rf_model4_scored_tmp AS
    SELECT *
    FROM rf_model4_scored
    JOIN balanced4_train
    USING (id);
    DROP TABLE rf_model4_scored;
    ALTER TABLE rf_model4_scored_tmp RENAME TO rf_model4_scored;
    SELECT * FROM rf_model4_scored LIMIT 10;

"""
cur.execute(query)
```

In [27]:
```
# Evaluate on entire 4.9-million "training" dataset
query = """
    DROP TABLE IF EXISTS rf_model2_scored;
    SELECT madlib.forest_predict('rf_model2',
                                 'public.training_data',
                                 'rf_model2_scored',
                                 'response');

    DROP TABLE IF EXISTS rf_model2_scored_tmp;
    CREATE TABLE rf_model2_scored_tmp AS
    SELECT *
    FROM rf_model2_scored
    JOIN public.training_data
    USING (id);
    DROP TABLE rf_model2_scored;
    ALTER TABLE rf_model2_scored_tmp RENAME TO rf_model2_scored;
    SELECT * FROM rf_model2_scored LIMIT 10;

"""
cur.execute(query)
```

In [31]:
```
# Evaluate on 311K "eval" dataset
query = """
    SET search_path=network_anomaly_run2,madlib;
    DROP TABLE IF EXISTS rf_model2_scored;
    SELECT madlib.forest_predict('rf_model2',
                                 'public.eval_data',
                                 'rf_model2_scored',
                                 'response');

    DROP TABLE IF EXISTS rf_model2_scored_tmp;
    CREATE TABLE rf_model2_scored_tmp AS
    SELECT *
    FROM rf_model2_scored
    JOIN public.eval_data
    USING (id);
    DROP TABLE rf_model2_scored;
    ALTER TABLE rf_model2_scored_tmp RENAME TO rf_model2_scored;
    SELECT * FROM rf_model2_scored LIMIT 10;

"""
cur.execute(query)
```

In [36]:
```
%config SqlMagic.autopandas=True
```

In [34]:
```
%%sql
select * from rf_model2_scored order by id limit 10;
```

```
10 rows affected.
```

Out[34]:

	id	estimated_y	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13
0	0	class18	ICMP	ECR_I	STAT10	1032.0	0.0	0	0	0	0	0	0	0.0	0.0
1	1	normal	TCP	HTTP	STAT10	280.0	3413.0	0	0	0	0	1	0	0.0	0.0
2	2	normal	TCP	HTTP	STAT10	218.0	1493.0	0	0	0	0	1	0	0.0	0.0
3	3	class18	ICMP	ECR_I	STAT10	520.0	0.0	0	0	0	0	0	0	0.0	0.0
4	4	class18	ICMP	ECR_I	STAT10	1032.0	0.0	0	0	0	0	0	0	0.0	0.0
5	5	class18	ICMP	ECR_I	STAT10	1032.0	0.0	0	0	0	0	0	0	0.0	0.0
6	6	normal	UDP	PRIVATE	STAT10	42.0	0.0	0	0	0	0	0	0	0.0	0.0
7	7	normal	TCP	HTTP	STAT10	305.0	1846.0	0	0	0	0	1	0	0.0	0.4
8	8	class18	ICMP	ECR_I	STAT10	1032.0	0.0	0	0	0	0	0	0	0.0	0.0
9	9	class18	ICMP	ECR_I	STAT10	520.0	0.0	0	0	0	0	0	0	0.0	0.0

In [ ]:
```
%%sql
SET search_path=network_anomaly_run2, madlib;
```

Confusion matrix:

In [30]:
```
%%sql
DROP TABLE IF EXISTS confusion;
SELECT madlib.confusion_matrix( 'rf_model2_scored', 'confusion', 'estima
ted_y', 'y');
SELECT * FROM confusion ORDER BY "class";
```

```
Done.
1 rows affected.
23 rows affected.
```

Out[30]:

	row_id	class	confusion_arr
0	1	class01	[2202, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
1	2	class02	[0, 25, 1, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
2	3	class03	[0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
3	4	class04	[0, 0, 0, 52, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
4	5	class05	[0, 0, 0, 0, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
5	6	class06	[0, 0, 0, 0, 0, 12447, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 5, 0, 0, 0, 0, 0, 21]
6	7	class07	[0, 0, 0, 0, 0, 0, 21, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
7	8	class08	[0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
8	9	class09	[0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0]
9	10	class10	[0, 0, 0, 0, 108, 0, 108, 0, 0, 1060156, 11, 0, 0, 0, 11247, 0, 256, 0, 1, 0, 0, 0, 130]
10	11	class11	[0, 0, 0, 0, 0, 832, 0, 0, 0, 5, 1476, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3]
11	12	class12	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
12	13	class13	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
13	14	class14	[0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 259, 0, 0, 0, 0, 0, 0, 0, 0, 0]
14	15	class15	[1, 0, 0, 0, 0, 6, 4, 0, 0, 3, 1, 1, 0, 0, 10352, 0, 38, 0, 0, 0, 0, 0, 7]
15	16	class16	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 1]
16	17	class17	[0, 0, 1, 0, 0, 10, 0, 1, 0, 19, 10, 0, 0, 0, 38, 0, 15749, 0, 0, 0, 3, 0, 61]
17	18	class18	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 2807744, 0, 0, 0, 0, 135]
18	19	class19	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0]
19	20	class20	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 977, 0, 0, 0]
20	21	class21	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1019, 0, 1]
21	22	class22	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 18, 0]
22	23	normal	[399, 221, 527, 227, 134, 4090, 189, 729, 37, 29, 12428, 222, 151, 6, 89, 330, 631, 4, 210, 336, 16682, 0, 935110]

In [28]:
```
%%sql DROP TABLE IF EXISTS accuracy_by_class;
CREATE TABLE accuracy_by_class AS
    SELECT y,correct, total,correct::FLOAT/total as accuracy
FROM (SELECT y, COUNT(*) AS correct FROM rf_model2_scored WHERE y=estima
ted_y GROUP BY y) c JOIN
     (SELECT y,count(*) AS total FROM rf_model2_scored GROUP BY y) t USI
NG(y);
SELECT * FROM accuracy_by_class ORDER BY y;
```

```
Done.
23 rows affected.
23 rows affected.
```

Out[28]:

y	correct	total	accuracy
class01	212	212	1.0
class02	20	27	0.740740740741
class03	7	17	0.411764705882
class04	15	15	1.0
class05	11	13	0.846153846154
class06	211	216	0.976851851852
class07	17	17	1.0
class08	15	15	1.0
class09	1	15	0.0666666666667
class10	192	199	0.964824120603
class11	213	214	0.995327102804
class12	16	16	1.0
class13	26	26	1.0
class14	17	17	1.0
class15	228	229	0.995633187773
class16	14	17	0.823529411765
class17	177	180	0.983333333333
class18	179	179	1.0
class19	23	23	1.0
class20	24	25	0.96
class21	184	184	1.0
class22	27	28	0.964285714286
normal	192	196	0.979591836735

In [29]:
```
%sql SELECT AVG(accuracy) AS balanced_accuracy FROM accuracy_by_class;
```

```
1 rows affected.
```

Out[29]:

balanced_accuracy
0.900378370374

This is overfitting ^^^ likely

CSV output for submission

In [37]:
```
eval_result = %sql select id as "ID", estimated_y as "ANOMALY" from rf_model2_scored order by ID;
```

```
311030 rows affected.
```

In [38]:
```
eval_result[:5]
```

Out[38]:

	ID	ANOMALY
0	0	class18
1	1	normal
2	2	normal
3	3	class18
4	4	class18

Write out eval dataset results to CSV to submit:

In [39]:
```
eval_result.to_csv("/Users/fmcquillan/Downloads/eval_result_run2.csv")
```

Done for now!