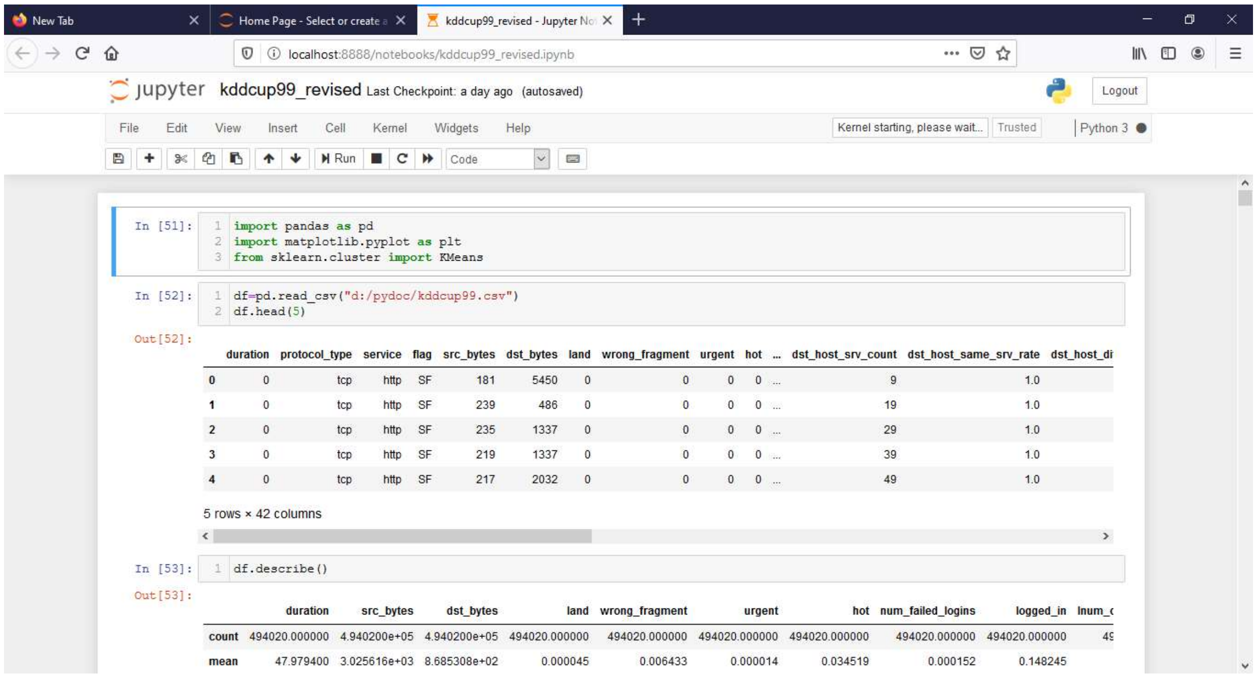
Step :1 :-- Load the csv data kddcup99.csv and results are below:



In [51]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [52]:
```python
df=pd.read_csv("d:/pydoc/kddcup99.csv")
df.head(5)
```

Out[52]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_di
0	0	tcp	http	SF	181	5450	0	0	0	0	...	9	1.0	
1	0	tcp	http	SF	239	486	0	0	0	0	...	19	1.0	
2	0	tcp	http	SF	235	1337	0	0	0	0	...	29	1.0	
3	0	tcp	http	SF	219	1337	0	0	0	0	...	39	1.0	
4	0	tcp	http	SF	217	2032	0	0	0	0	...	49	1.0	

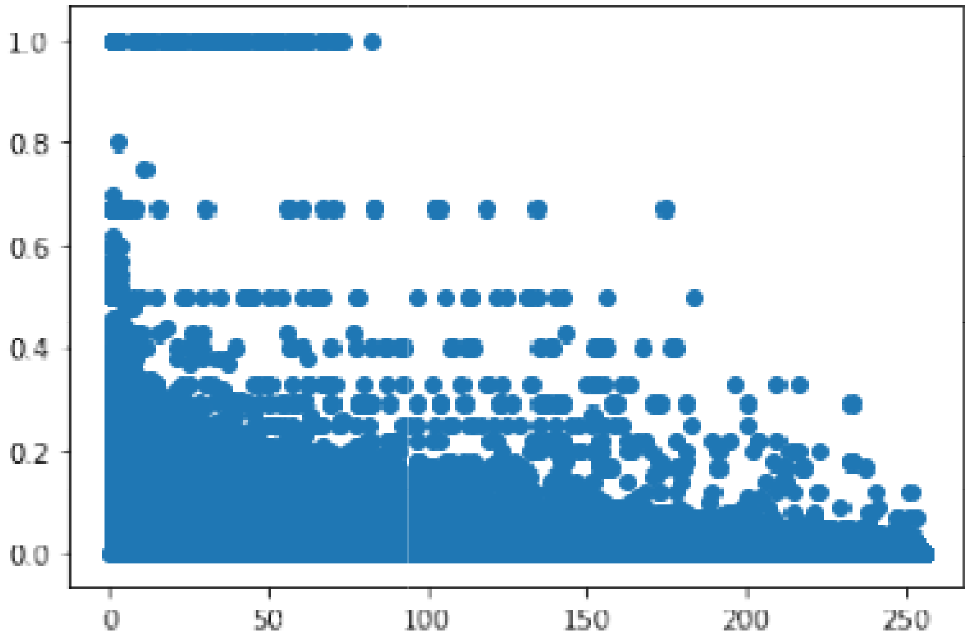
5 rows × 42 columns

In [53]:
```python
df.describe()
```

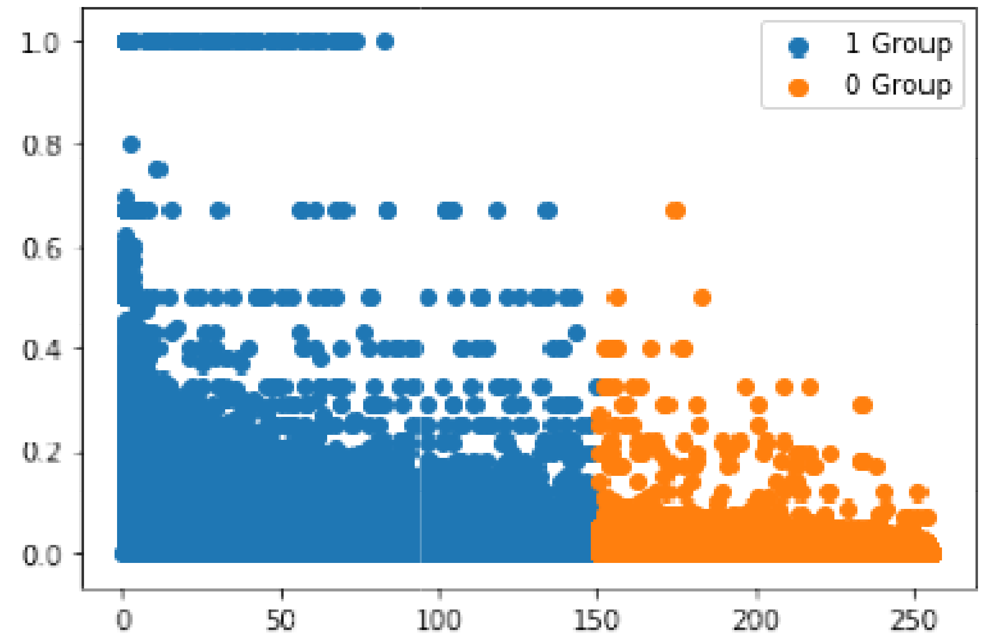
Out[53]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	lnum_c
count	494020.000000	4.940200e+05	4.940200e+05	494020.000000	494020.000000	494020.000000	494020.000000	494020.000000	494020.000000	49
mean	47.979400	3.025616e+03	8.685308e+02	0.000045	0.006433	0.000014	0.034519	0.000152	0.148245	

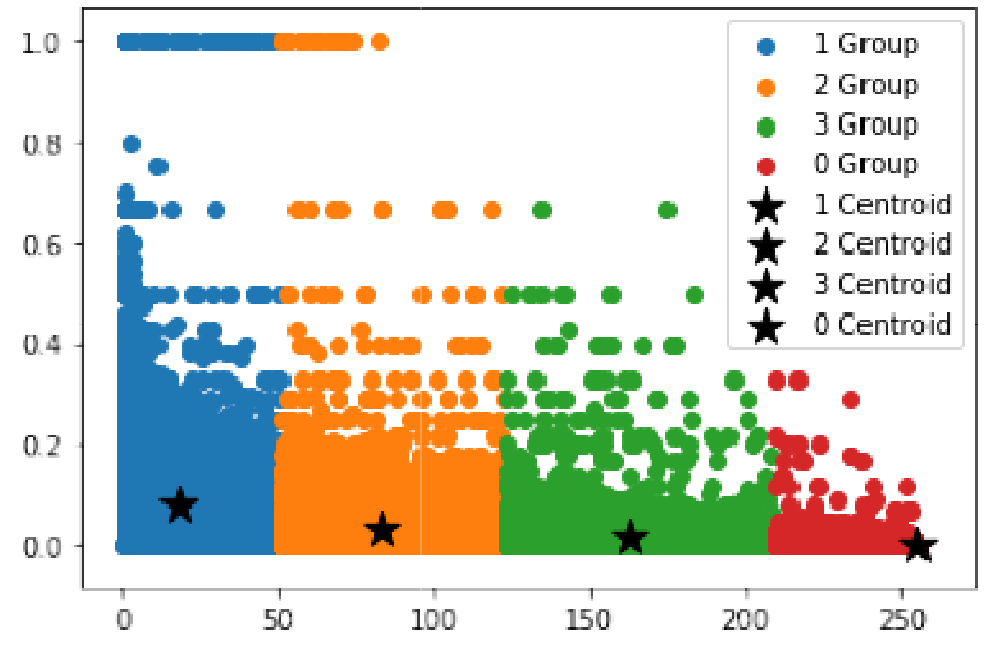
Step 2:-- scattering results of two columns of this selection.



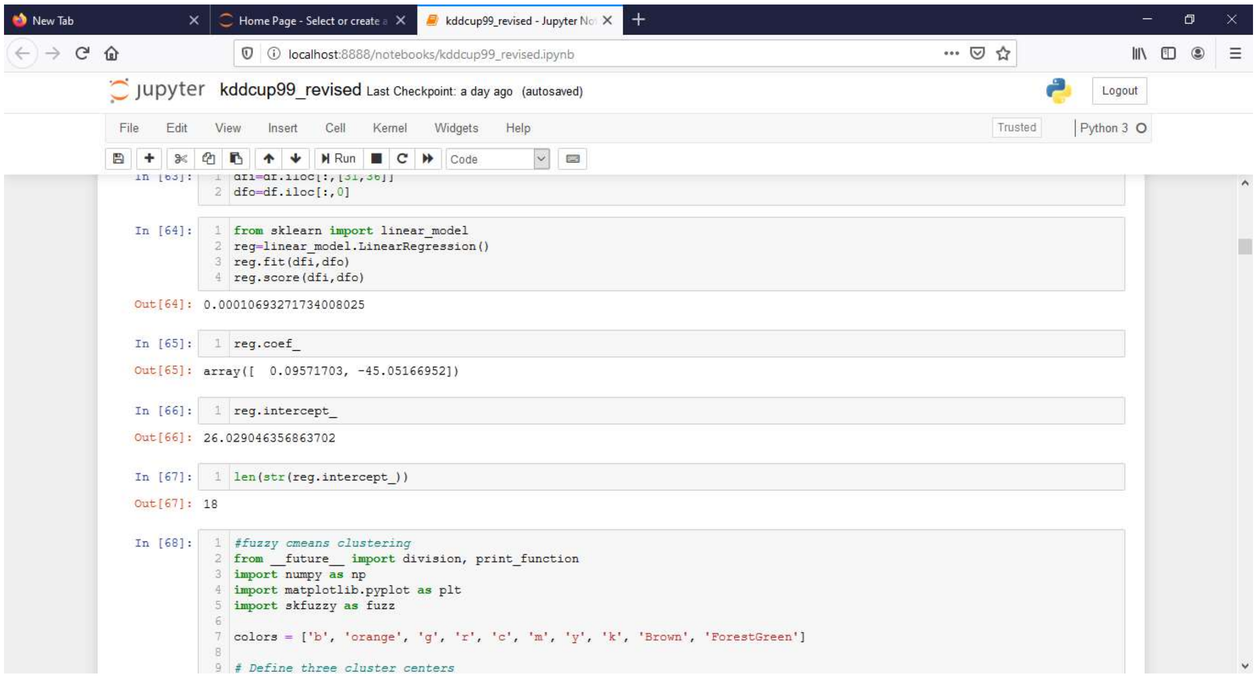
Step 3: -- K means clustering results.



Step 4: -- kmeans clustering results after prediction.

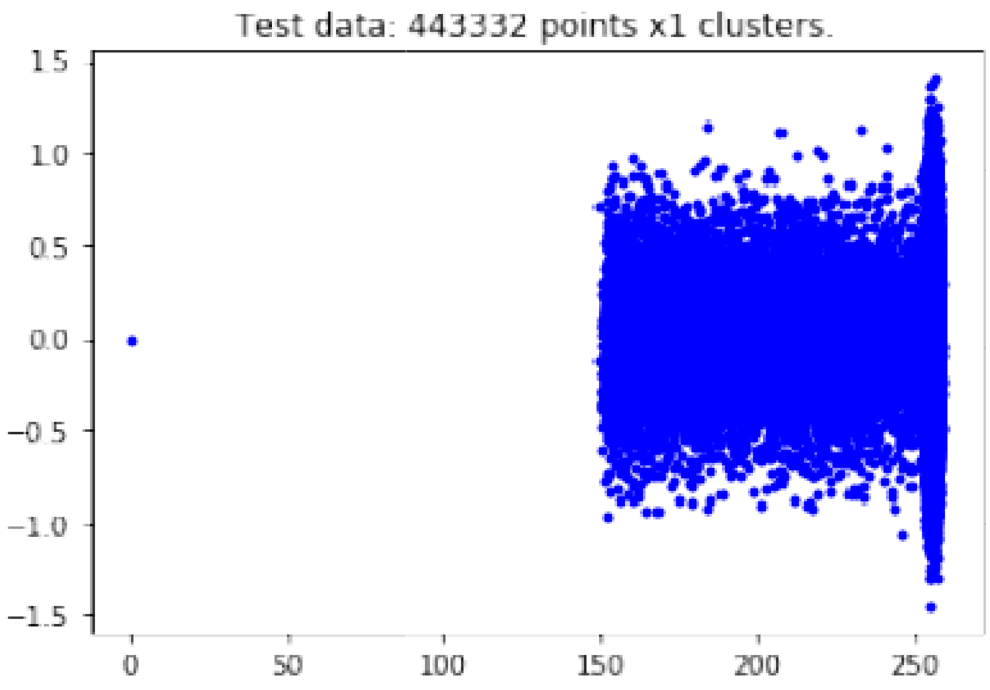


Step 5: -- k means score by linear regression(hybrid k means(k means + linear regression))

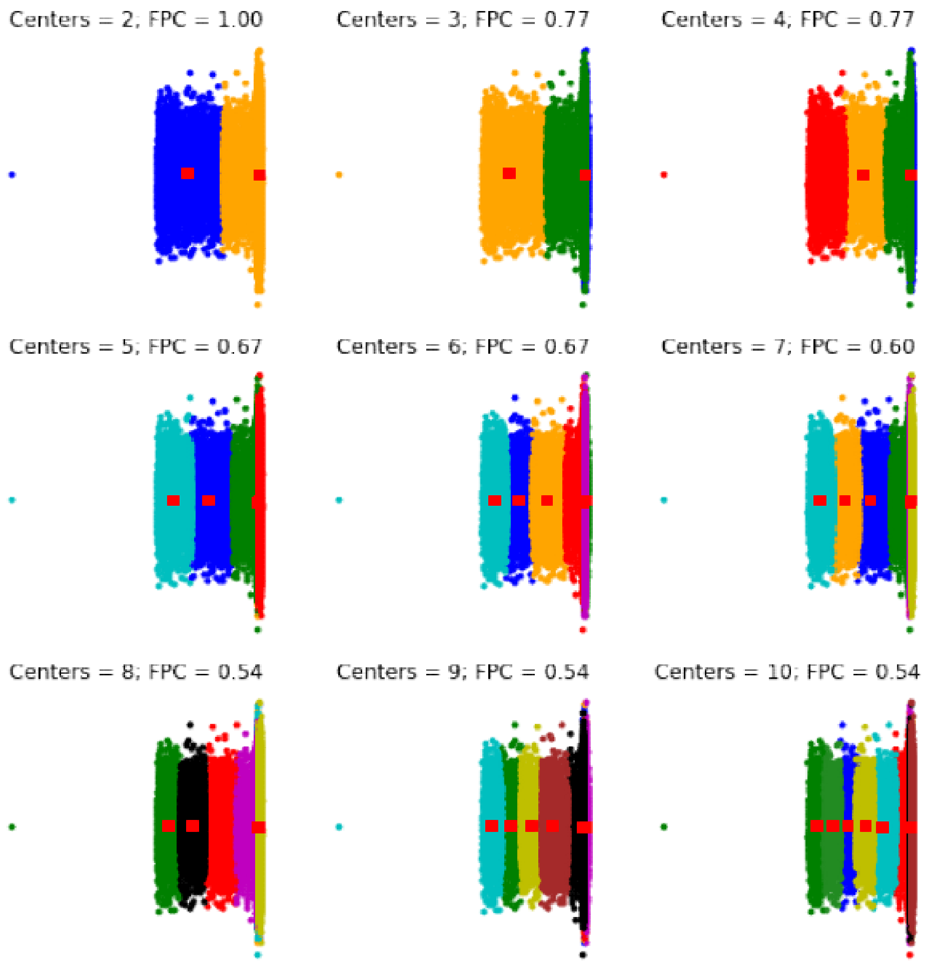

New Tab
Home Page - Select or create a
kddcup99_revised - Jupyter No
localhost:8888/notebooks/kddcup99_revised.ipynb
jupyter kddcup99_revised Last Checkpoint: a day ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [63]:  dfi=df.iloc[:,[31,36]]
          dfo=df.iloc[:,0]

In [64]:  from sklearn import linear_model
          reg=linear_model.LinearRegression()
          reg.fit(dfi,dfo)
          reg.score(dfi,dfo)

Out[64]: 0.00010693271734008025

In [65]:  reg.coef_

Out[65]: array([  0.09571703, -45.05166952])

In [66]:  reg.intercept_

Out[66]: 26.029046356863702

In [67]:  len(str(reg.intercept_))

Out[67]: 18

In [68]:  #fuzzy cmeans clustering
          from __future__ import division, print_function
          import numpy as np
          import matplotlib.pyplot as plt
          import skfuzzy as fuzz

          colors = ['b', 'orange', 'g', 'r', 'c', 'm', 'y', 'k', 'Brown', 'ForestGreen']

          # Define three cluster centers
```

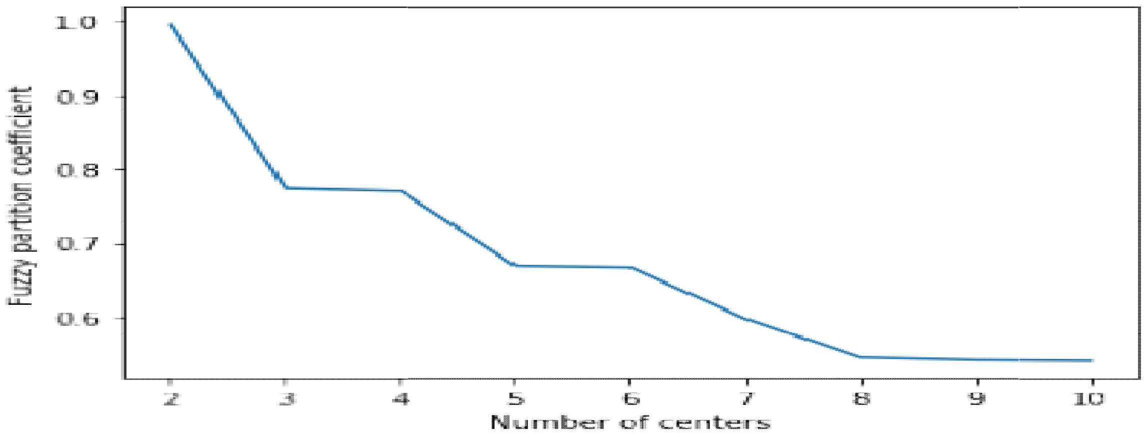
Step 6: -- fuzzy c means clustering of two columns.



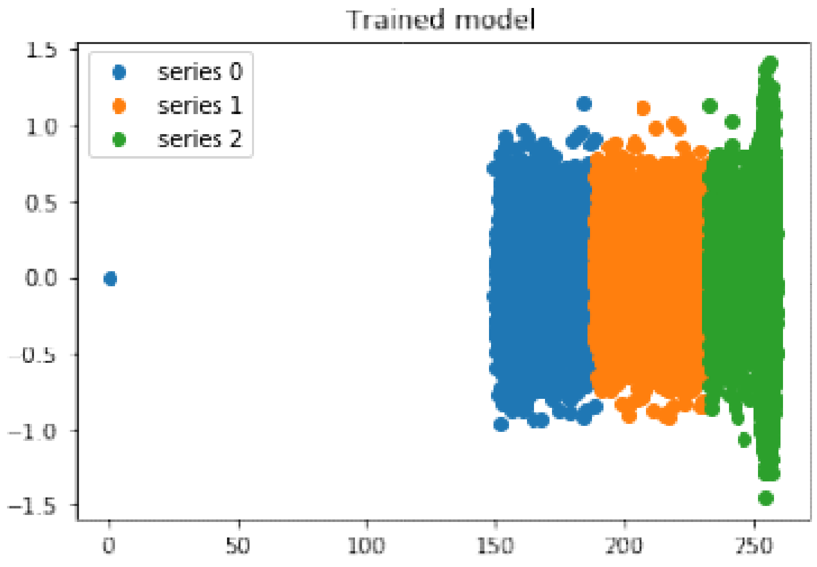
Step 7: -- fuzzy c means point centers representation.



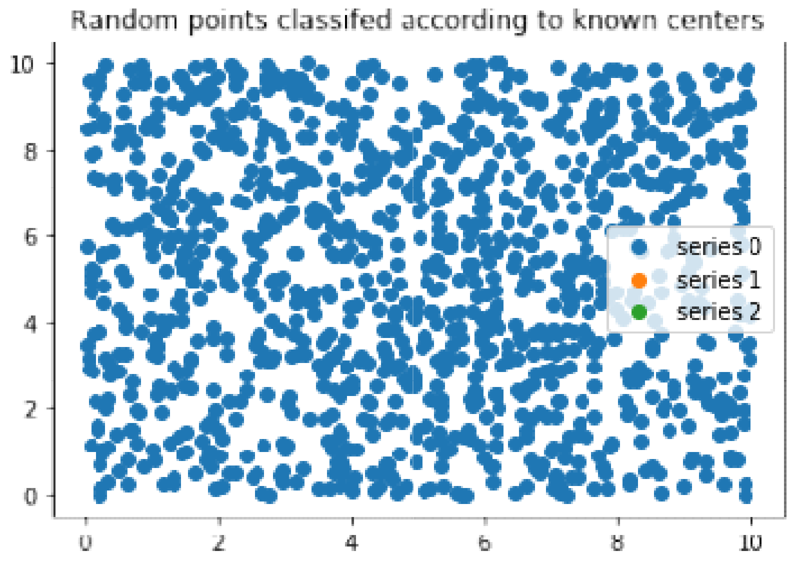
Step8: -- fuzzy partition coefficient.



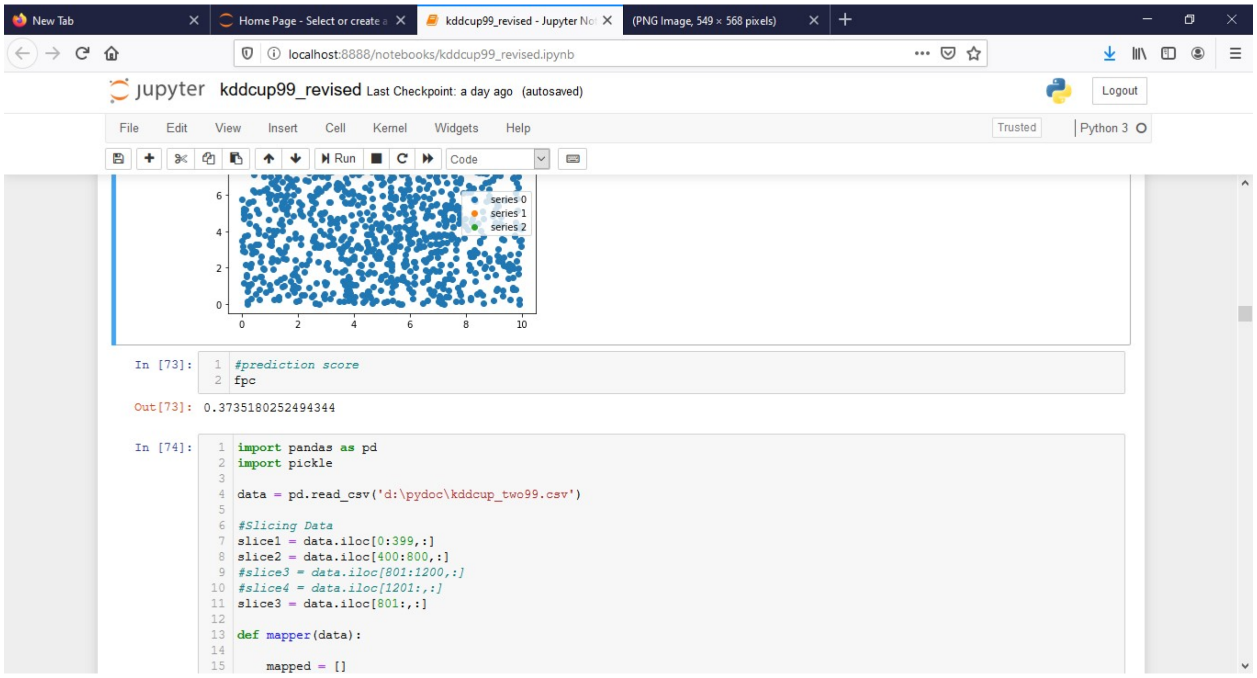
Step 9: -- regenerate fuzzy model with 3 cluster centers



Step 10: -- Random points classification according to known centers.



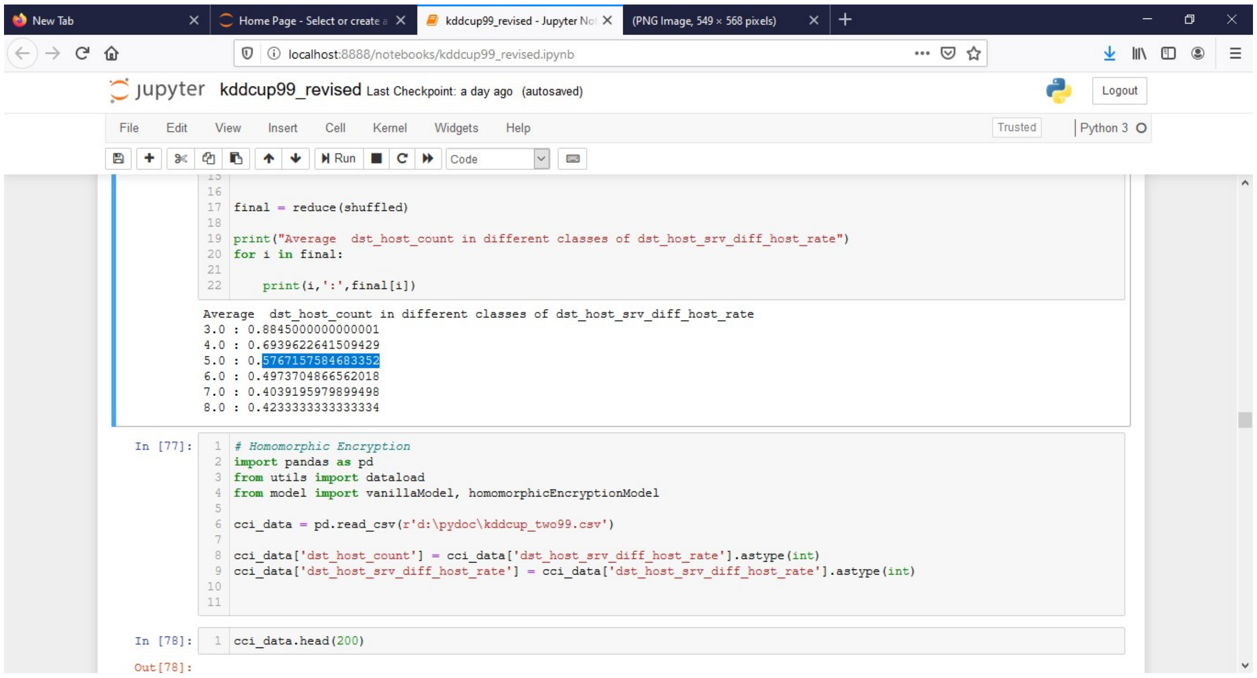
Step 4: -- fuzzy c means prediction score.


localhost:8888/notebooks/kddcup99_revised.ipynb
jupyter kddcup99_revised Last Checkpoint: a day ago (autosaved)
series 0
series 1
series 2

```
In [73]:  #prediction score
          fpc

Out[73]:  0.3735180252494344
```

```
In [74]:  import pandas as pd
          import pickle

          data = pd.read_csv('d:\pydoc\kddcup_two99.csv')

          #Slicing Data
          slice1 = data.iloc[0:399,:]
          slice2 = data.iloc[400:800,:]
          #slice3 = data.iloc[801:1200,:]
          #slice4 = data.iloc[1201:,:]
          slice3 = data.iloc[801:,:]

          def mapper(data):

              mapped = []
```

Step 9: -- average score up to eight columns after reducer technique.


localhost:8888/notebooks/kddcup99_revised.ipynb
jupyter kddcup99_revised Last Checkpoint: a day ago (autosaved)

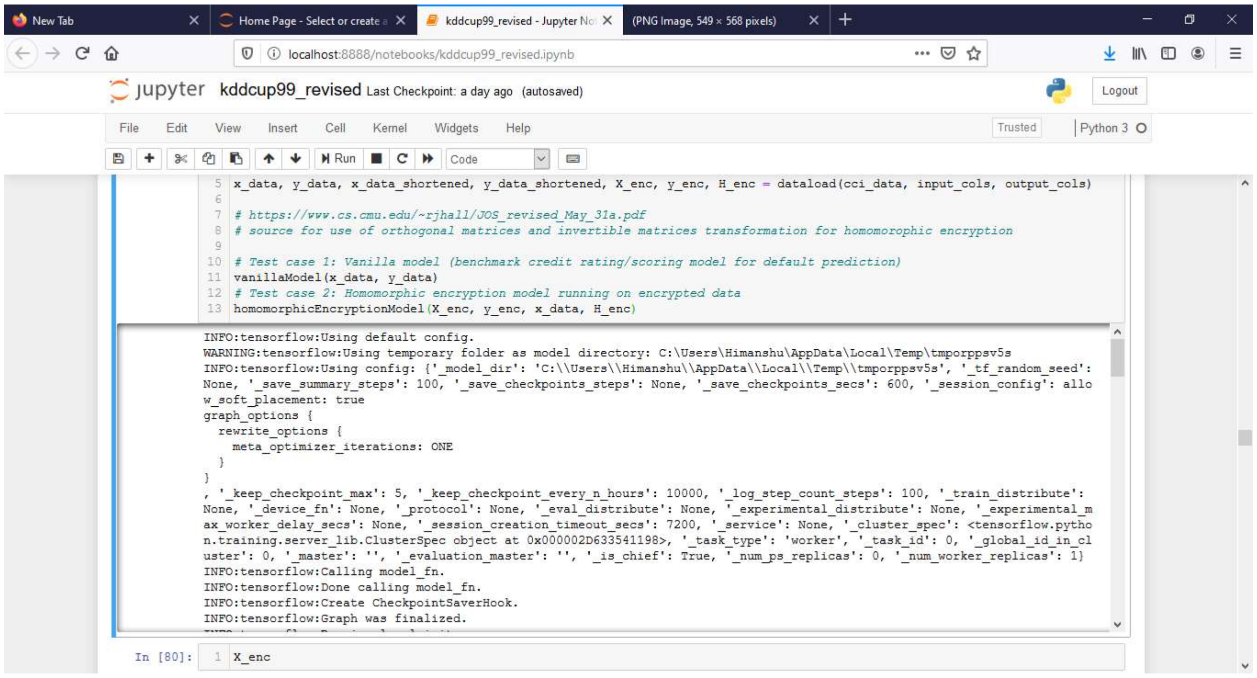
```
final = reduce(shuffled)

print("Average  dst_host_count in different classes of dst_host_srv_diff_host_rate")
for i in final:

    print(i,':',final[i])
```

```
Average  dst_host_count in different classes of dst_host_srv_diff_host_rate
3.0 : 0.8845000000000001
4.0 : 0.6939622641509429
5.0 : 0.5767157584683352
6.0 : 0.4973704866562018
7.0 : 0.4039195979899498
8.0 : 0.4233333333333334
```

```
In [77]:  # Homomorphic Encryption
          import pandas as pd
          from utils import dataload
          from model import vanillaModel, homomorphicEncryptionModel

          cci_data = pd.read_csv(r'd:\pydoc\kddcup_two99.csv')

          cci_data['dst_host_count'] = cci_data['dst_host_srv_diff_host_rate'].astype(int)
          cci_data['dst_host_srv_diff_host_rate'] = cci_data['dst_host_srv_diff_host_rate'].astype(int)
```

```
In [78]:  cci_data.head(200)

Out[78]:
```

Step 10: -- Homomorphic encryption model output representation.



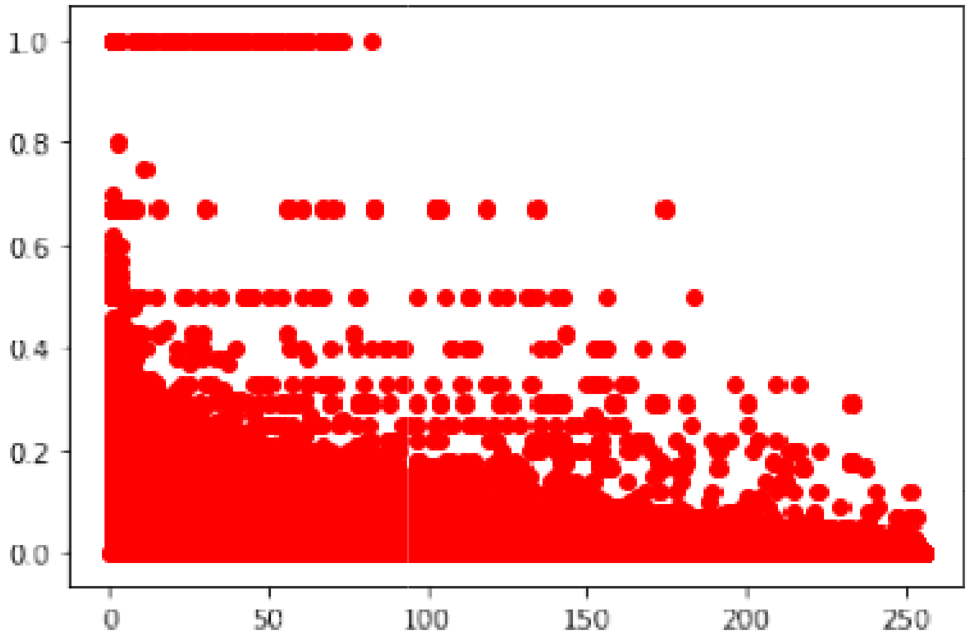
```
x_data, y_data, x_data_shortened, y_data_shortened, X_enc, y_enc, H_enc = dataload(cci_data, input_cols, output_cols)

# https://www.cs.cmu.edu/~rjhall/JOS_revised_May_31a.pdf
# source for use of orthogonal matrices and invertible matrices transformation for homomorophic encryption

# Test case 1: Vanilla model (benchmark credit rating/scoring model for default prediction)
vanillaModel(x_data, y_data)
# Test case 2: Homomorphic encryption model running on encrypted data
homomorphicEncryptionModel(X_enc, y_enc, x_data, H_enc)
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\Himanshu\AppData\Local\Temp\tmporppsv5s
INFO:tensorflow:Using config: {'_model_dir': 'C:\\Users\\Himanshu\\AppData\\Local\\Temp\\tmporppsv5s', '_tf_random_seed':
None, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_save_checkpoints_secs': 600, '_session_config': allo
w_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_train_distribute':
None, '_device_fn': None, '_protocol': None, '_eval_distribute': None, '_experimental_distribute': None, '_experimental_m
ax_worker_delay_secs': None, '_session_creation_timeout_secs': 7200, '_service': None, '_cluster_spec': <tensorflow.pytho
n.training.server_lib.ClusterSpec object at 0x000002D633541198>, '_task_type': 'worker', '_task_id': 0, '_global_id_in_cl
uster': 0, '_master': '', '_evaluation_master': '', '_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
```

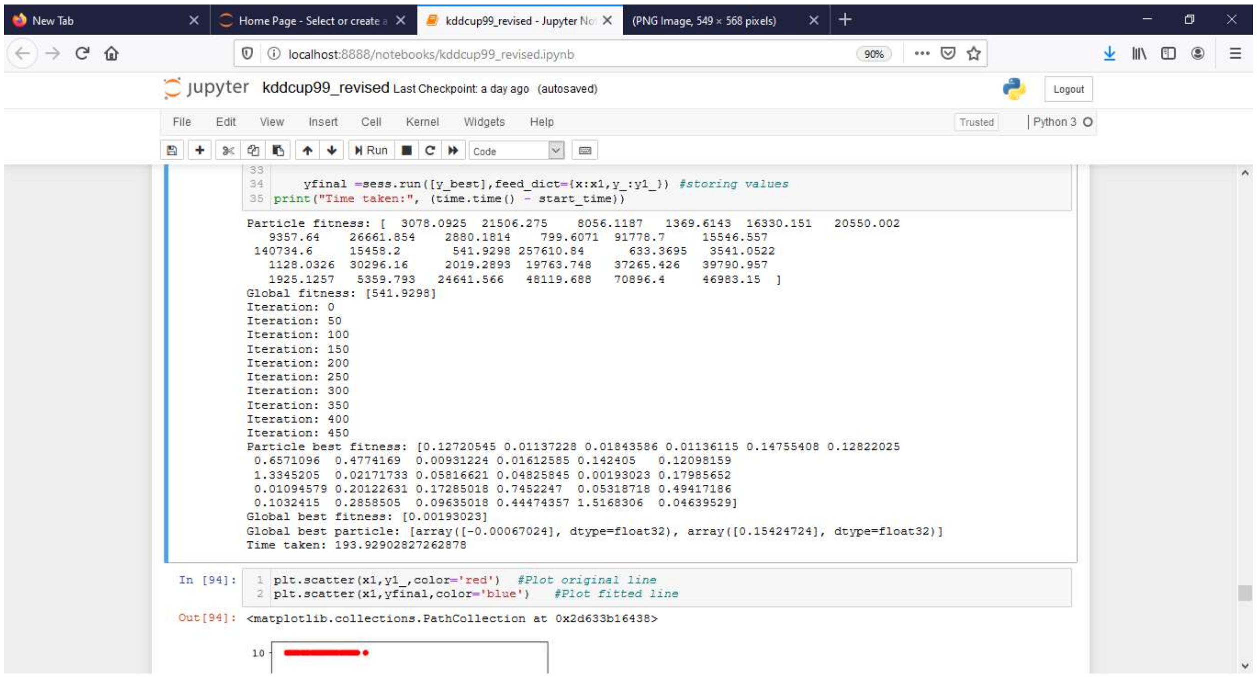
```
In [80]: X_enc
```

jupyter kddcup99_revised Last Checkpoint: a day ago (autosaved)
localhost:8888/notebooks/kddcup99_revised.ipynb
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Logout
Run
Code

Step 11: -- particle swarm optimization algorithm with optimization after homomorphic encryption model.



Step 12 : -- pso fitness score representation.



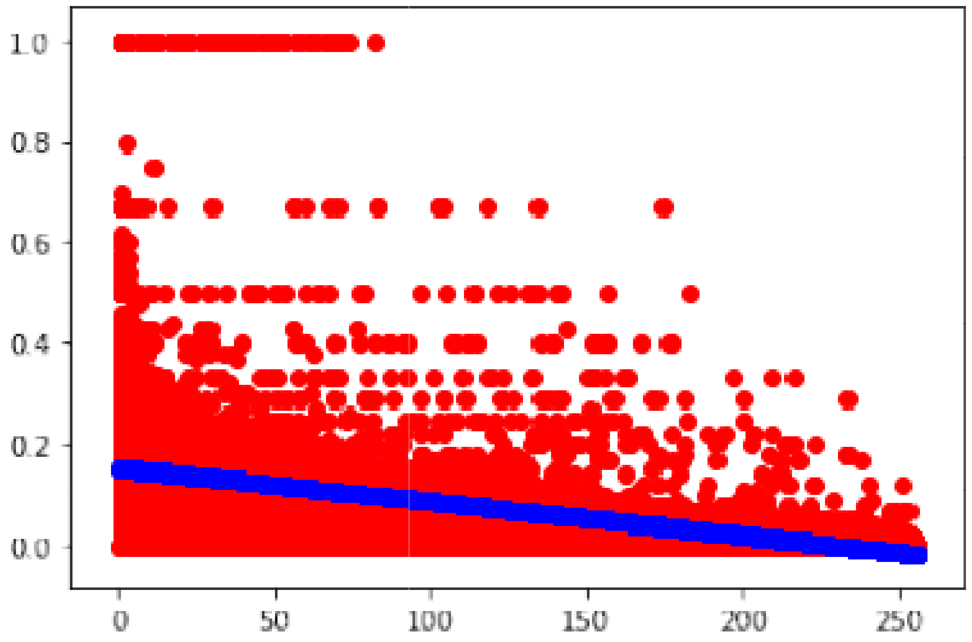
```
    yfinal =sess.run([y_best],feed_dict={x:x1,y_:y1_}) #storing values
print("Time taken:", (time.time() - start_time))
```

```
Particle fitness: [  3078.0925   21506.275    8056.1187   1369.6143  16330.151   20550.002
   9357.64    26661.854    2880.1814    799.6071  91778.7     15546.557
 140734.6     15458.2      541.9298 257610.84      633.3695   3541.0522
   1128.0326  30296.16     2019.2893  19763.748   37265.426   39790.957
   1925.1257   5359.793   24641.566   48119.688   70896.4     46983.15   ]
Global fitness: [541.9298]
Iteration: 0
Iteration: 50
Iteration: 100
Iteration: 150
Iteration: 200
Iteration: 250
Iteration: 300
Iteration: 350
Iteration: 400
Iteration: 450
Particle best fitness: [0.12720545 0.01137228 0.01843586 0.01136115 0.14755408 0.12822025
 0.6571096  0.4774169  0.00931224 0.01612585 0.142405   0.12098159
 1.3345205  0.02171733 0.05816621 0.04825845 0.00193023 0.17985652
 0.01094579 0.20122631 0.17285018 0.7452247  0.05318718 0.49417186
 0.1032415  0.2858505  0.09635018 0.44474357 1.5168306  0.04639529]
Global best fitness: [0.00193023]
Global best particle: [array([-0.00067024], dtype=float32), array([0.15424724], dtype=float32)]
Time taken: 193.92902827262878
```

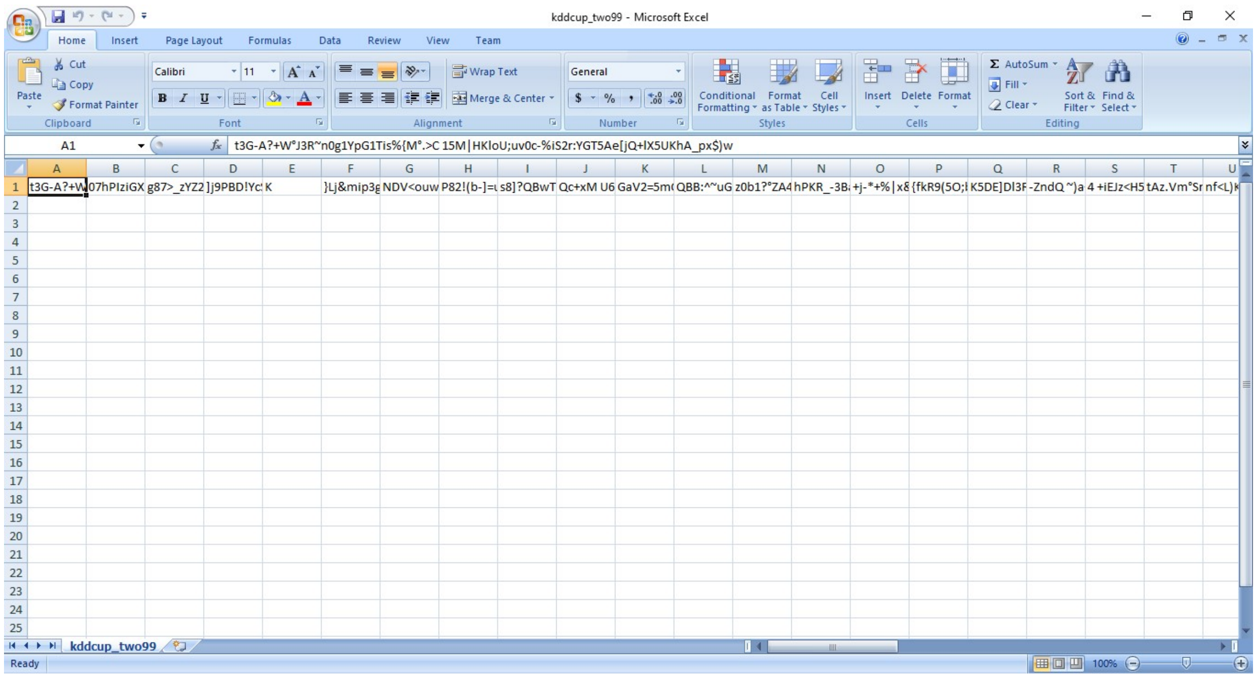
```
In [94]:
plt.scatter(x1,y1_,color='red')  #Plot original line
plt.scatter(x1,yfinal,color='blue')   #Plot fitted line
```

```
Out[94]: <matplotlib.collections.PathCollection at 0x2d633b16438>
```

Step13: -- clustering representation of fitness of pso.



Step 14: -- encrypted data in csv file representation by implementation of cipher text encryption.

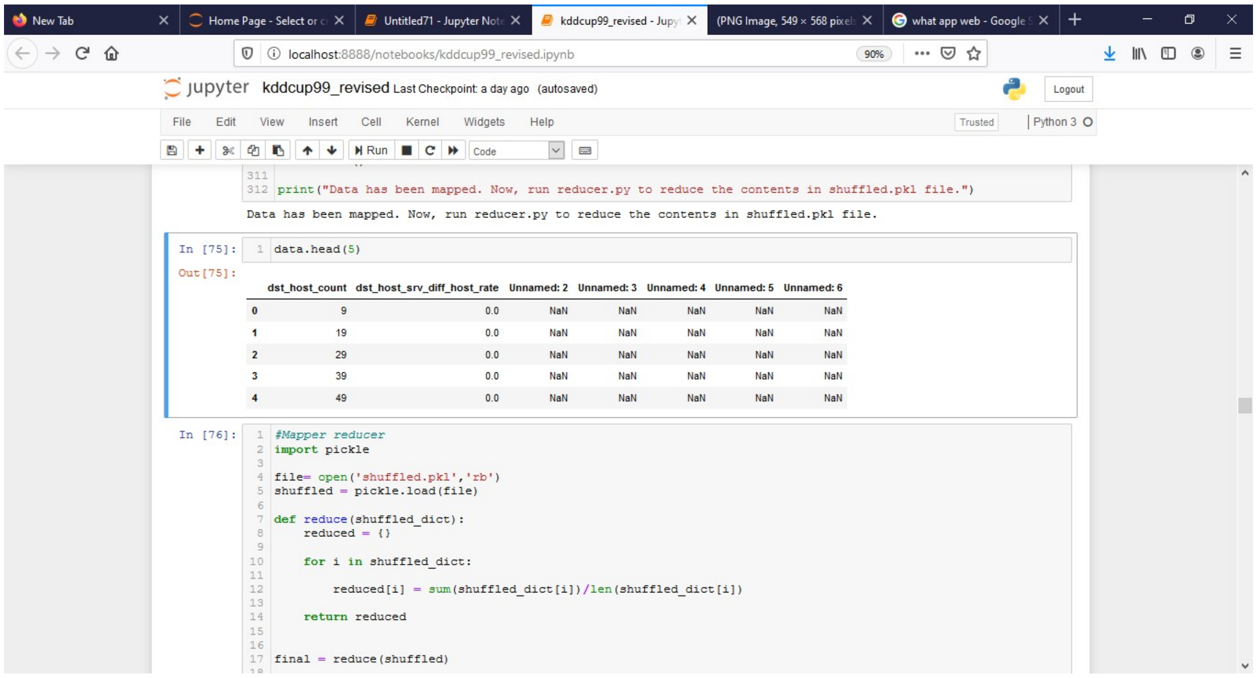


kddcup_two99 - Microsoft Excel

A1: t3G-A?+W°J3R~n0g1YpG1Tis%{M°.>C 15M|HKloU;uv0c-%iS2r:YGT5Ae[jQ+lX5UKhA_px$)w

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	t3G-A?+W	07hPIziGX	g87>_zYZ2	]j9PBD!Yc	K	}Lj&mip3g	NDV<ouw	P82!(b-]=ι	s8]?QBwT	Qc+xM U6	GaV2=5m(	QBB:^~uG	z0b1?°ZA4	hPKR_-3B	+j-*+%|x8	{fkR9(5O;I	K5DE]DI3F	-ZndQ ~)a	4 +iEJz<H5	tAz.Vm°Sr	nf<L)K

Step 15: -- simple data in csv file representation without implementation of cipher text encryption.



jupyter kddcup99_revised Last Checkpoint: a day ago (autosaved)

```
print("Data has been mapped. Now, run reducer.py to reduce the contents in shuffled.pkl file.")
```

```
Data has been mapped. Now, run reducer.py to reduce the contents in shuffled.pkl file.
```

In [75]:

```
data.head(5)
```

Out[75]:

	dst_host_count	dst_host_srv_diff_host_rate	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6
0	9	0.0	NaN	NaN	NaN	NaN	NaN
1	19	0.0	NaN	NaN	NaN	NaN	NaN
2	29	0.0	NaN	NaN	NaN	NaN	NaN
3	39	0.0	NaN	NaN	NaN	NaN	NaN
4	49	0.0	NaN	NaN	NaN	NaN	NaN

In [76]:

```
#Mapper reducer
import pickle

file= open('shuffled.pkl','rb')
shuffled = pickle.load(file)

def reduce(shuffled_dict):
    reduced = {}

    for i in shuffled_dict:

        reduced[i] = sum(shuffled_dict[i])/len(shuffled_dict[i])

    return reduced


final = reduce(shuffled)
```