# Mental Arts Project

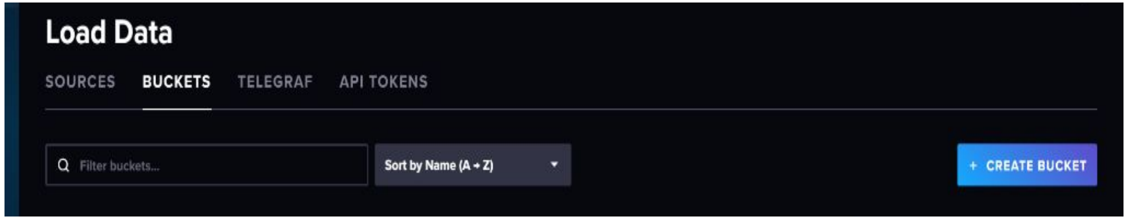
Time series studying using InfluxDB

Zehra Göl

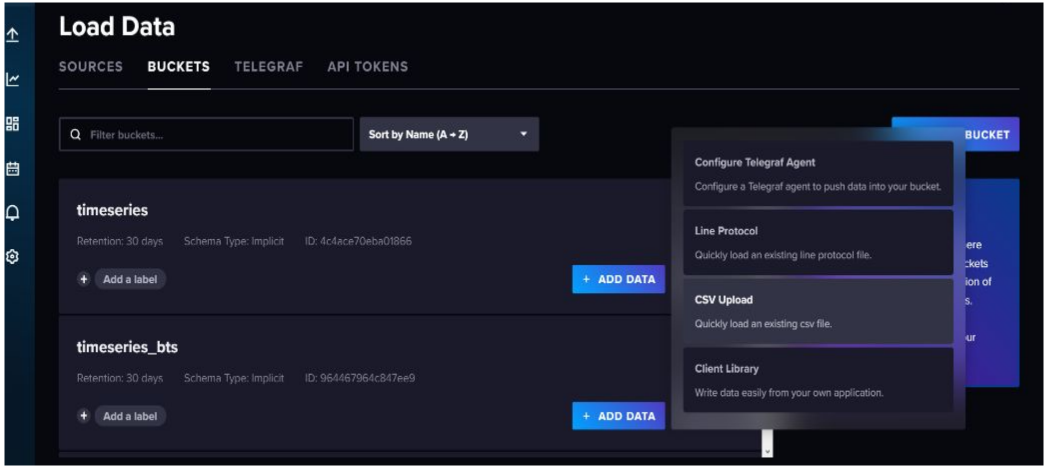
## What Is InfluxDB

InfluxDB is a high performance Time Series Database. It can store hundreds of thousands of points per second. The InfluxDB SQL-like query language was built specifically for time series.

# Creating a bucket and Load Data



bucket is where the data stored



timeseries

Retention: 30 days Schema Type: Implicit ID: 4c4ace70eba01866

timeseries_bts

Retention: 30 days Schema Type: Implicit ID: 964467964c847ee9

Configure Telegraf Agent — Configure a Telegraf agent to push data into your bucket.

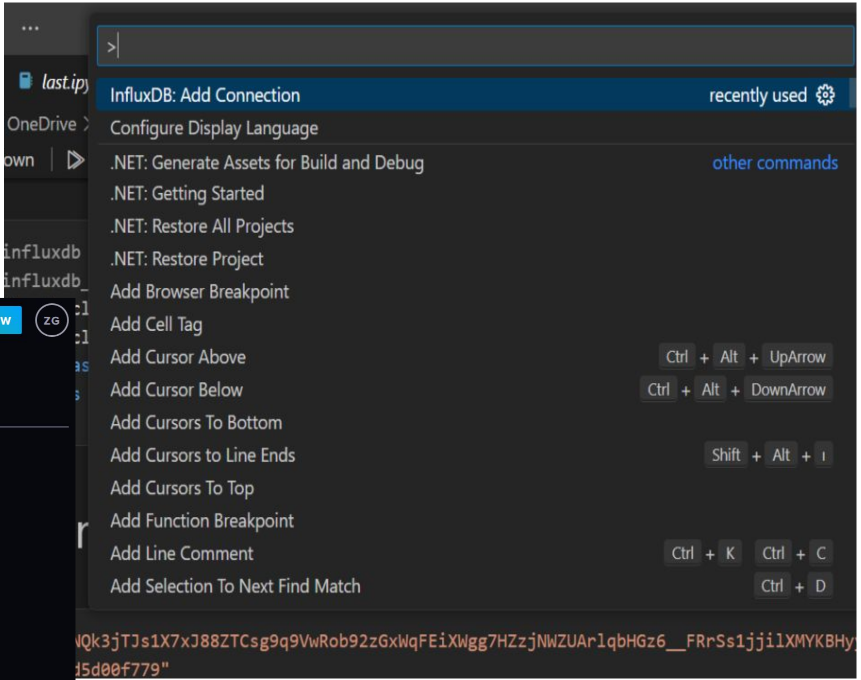
Line Protocol — Quickly load an existing line protocol file.

CSV Upload — Quickly load an existing csv file.

Client Library — Write data easily from your own application.

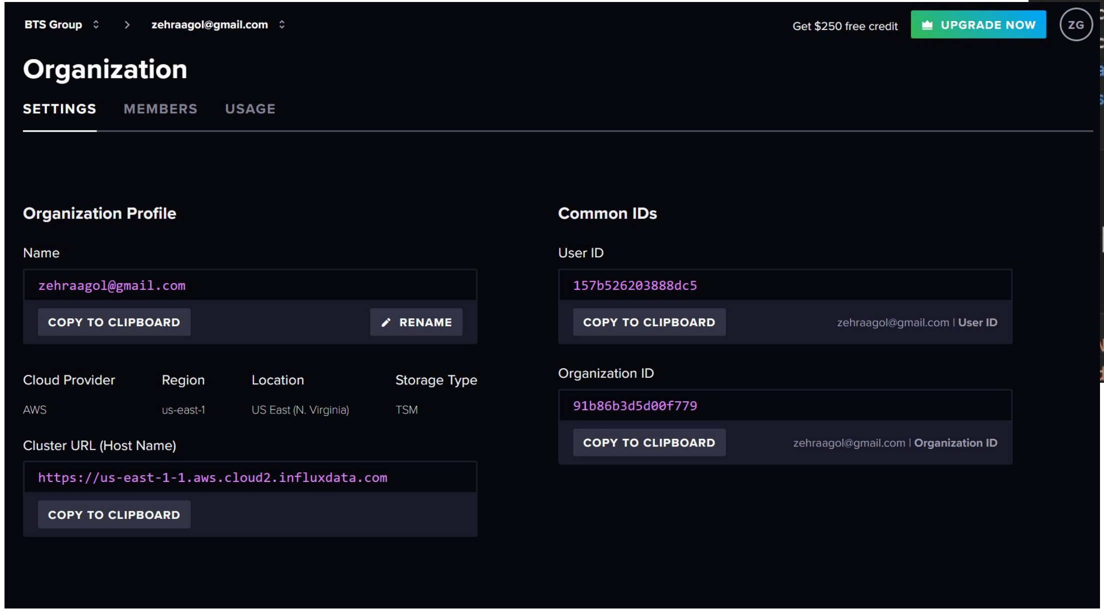
# influxdb-vscode extension

ctrl+shift+p



```
>|
InfluxDB: Add Connection                                   recently used
Configure Display Language
.NET: Generate Assets for Build and Debug                  other commands
.NET: Getting Started
.NET: Restore All Projects
.NET: Restore Project
Add Browser Breakpoint
Add Cell Tag
Add Cursor Above                                  Ctrl + Alt + UpArrow
Add Cursor Below                                  Ctrl + Alt + DownArrow
Add Cursors To Bottom
Add Cursors to Line Ends                          Shift + Alt + I
Add Cursors To Top
Add Function Breakpoint
Add Line Comment                                  Ctrl + K  Ctrl + C
Add Selection To Next Find Match                  Ctrl + D
```

```
last.ipy
OneDrive >
own
influxdb
influxdb_
NQk3jTJs1X7xJ88ZTCsg9q9VwRob92zGxWqFEiXWgg7HZzjNWZUArlqbHGz6__FRrSs1jjilXMYKBHy
d5d00f779"
```



BTS Group > zehraagol@gmail.com — Get $250 free credit — UPGRADE NOW — ZG

## Organization

SETTINGS MEMBERS USAGE

### Organization Profile

Name: zehraagol@gmail.com

COPY TO CLIPBOARD — RENAME

| Cloud Provider | Region | Location | Storage Type |
|---|---|---|---|
| AWS | us-east-1 | US East (N. Virginia) | TSM |

Cluster URL (Host Name): https://us-east-1-1.aws.cloud2.influxdata.com

COPY TO CLIPBOARD

### Common IDs

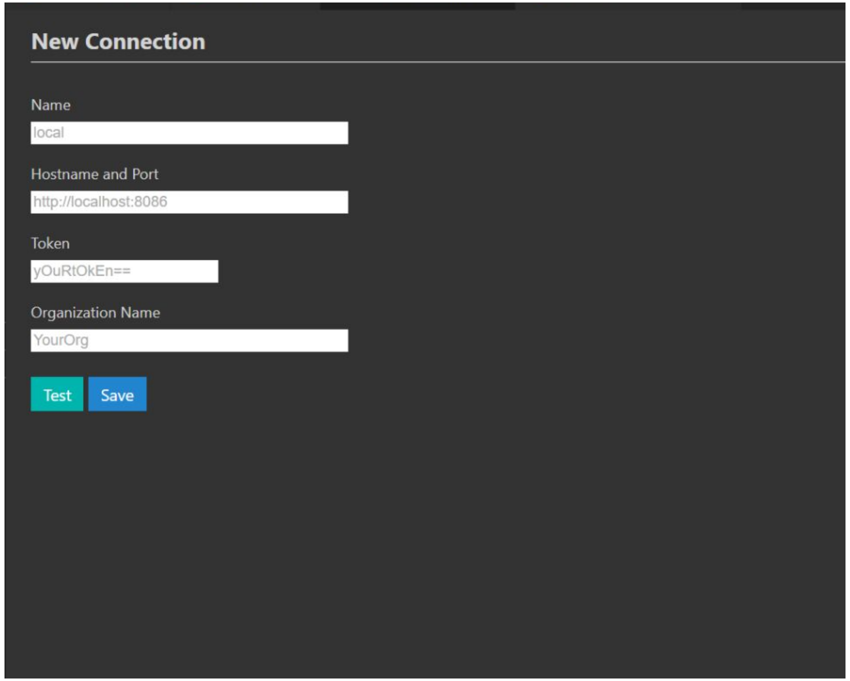
User ID: 157b526203888dc5

COPY TO CLIPBOARD — zehraagol@gmail.com | User ID

Organization ID: 91b86b3d5d00f779

COPY TO CLIPBOARD — zehraagol@gmail.com | Organization ID

all needed information is in influxdbcloud, here is mine.



New Connection

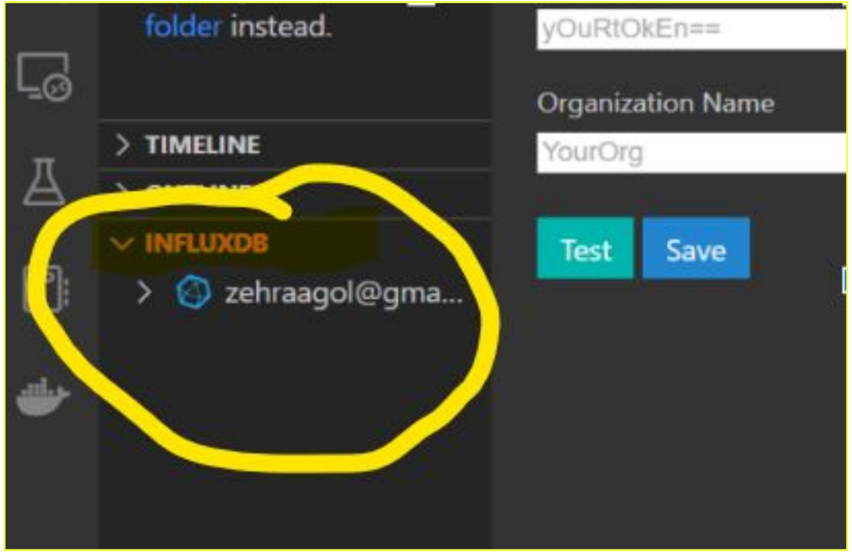
Name
local

Hostname and Port
http://localhost:8086

Token
yOuRtOkEn==

Organization Name
YourOrg

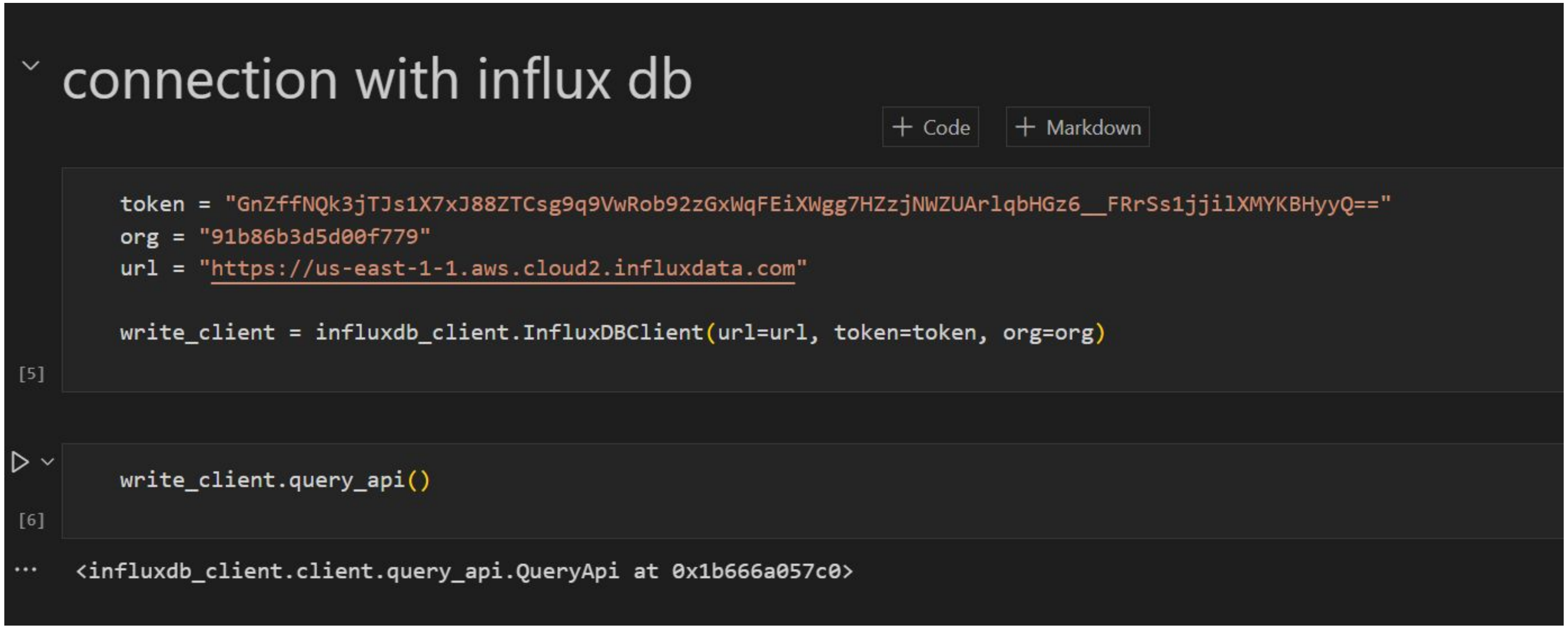
after entering the information, extension was done



folder instead.

yOuRtOkEn==

Organization Name
YourOrg



## connection with influx db

```
token = "GnZffNQk3jTJs1X7xJ88ZTCsg9q9VwRob92zGxWqFEiXWgg7HZzjNWZUArlqbHGz6__FRrSs1jjilXMYKBHyyQ=="
org = "91b86b3d5d00f779"
url = "https://us-east-1-1.aws.cloud2.influxdata.com"

write_client = influxdb_client.InfluxDBClient(url=url, token=token, org=org)
```

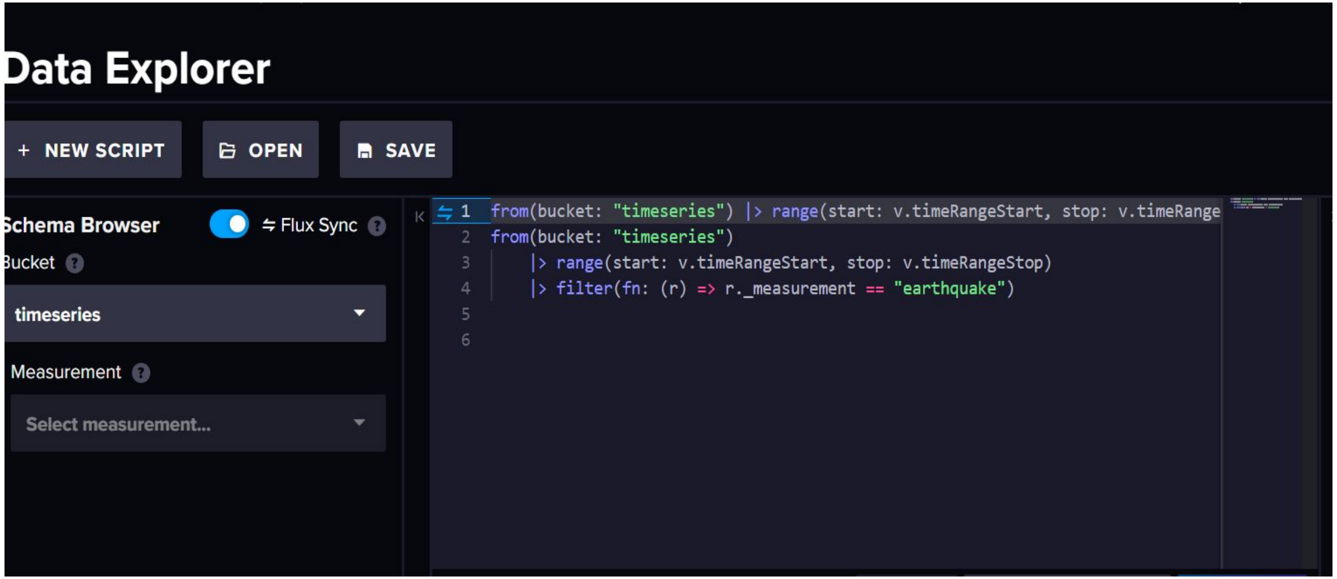
[5]

```
write_client.query_api()
```

[6]

```
<influxdb_client.client.query_api.QueryApi at 0x1b666a057c0>
```

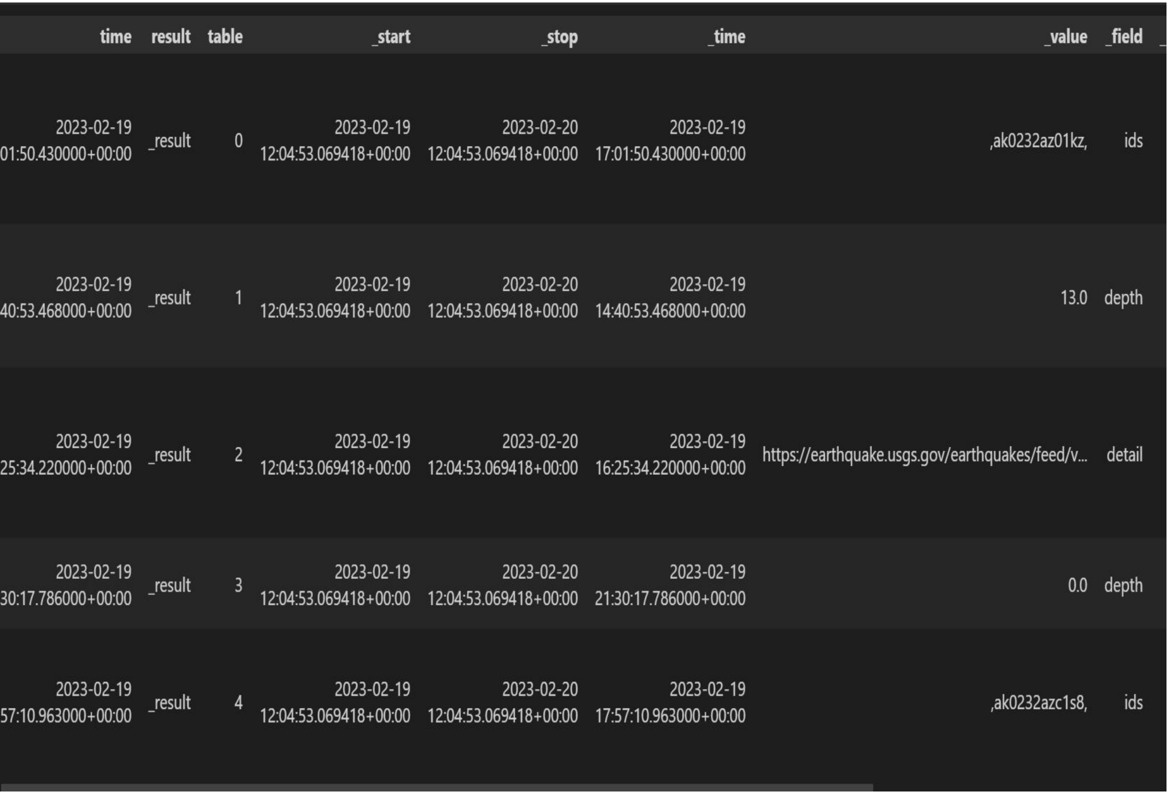
# reading query with flux language



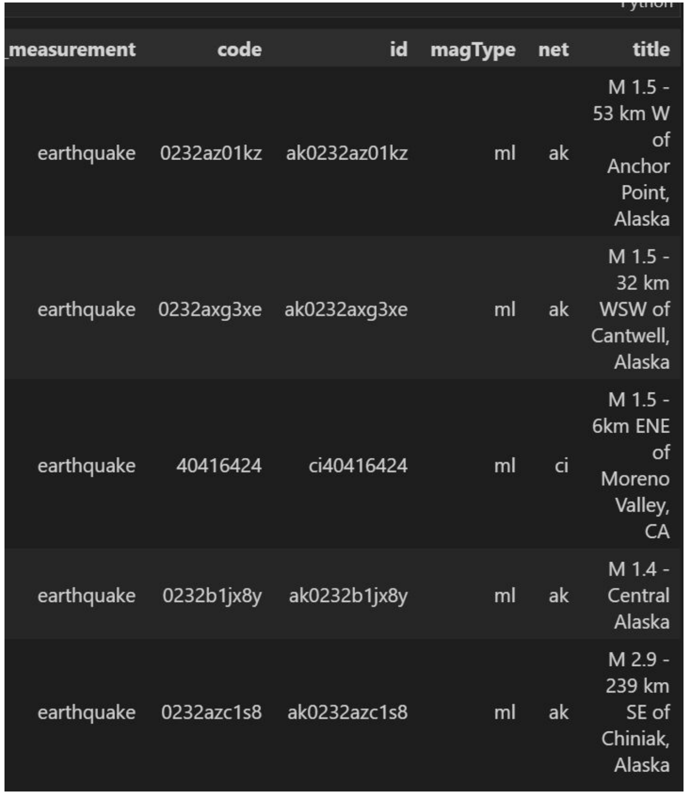
```
from(bucket: "timeseries") |> range(start: v.timeRangeStart, stop: v.timeRange
from(bucket: "timeseries")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r._measurement == "earthquake")
```

```python
query_api = write_client.query_api()

query = 'from(bucket: "timeseries")|> range(start: -1d) |> filter(fn: (r) => r._measurement == "earthquake")'

result = query_api.query(query=query)
```

# Data

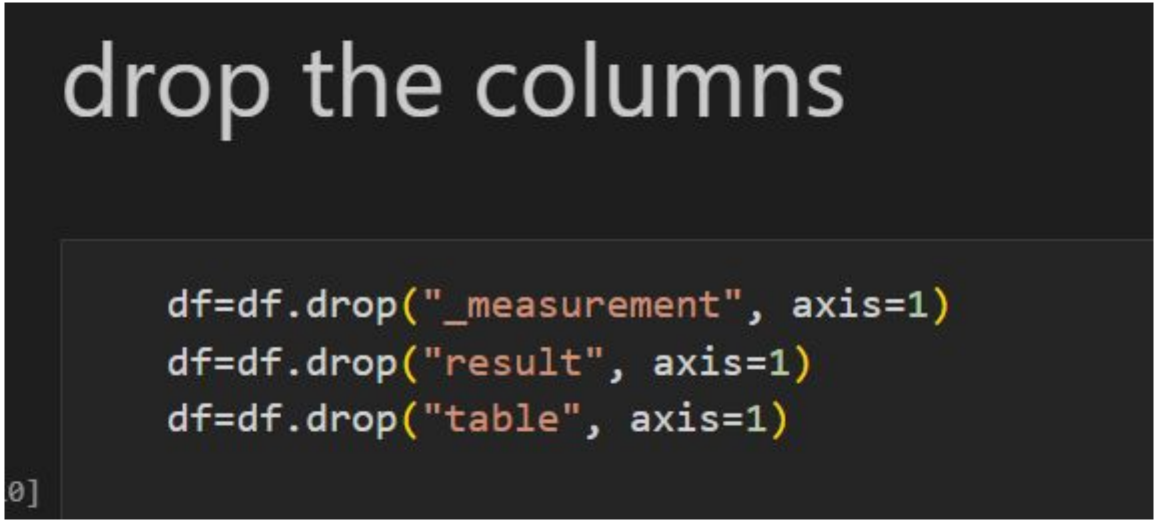


| time | result | table | _start | _stop | _time | _value | _field |
|---|---|---|---|---|---|---|---|
| 2023-02-19 01:50.430000+00:00 | _result | 0 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 17:01:50.430000+00:00 | ,ak0232az01kz, | ids |
| 2023-02-19 40:53.468000+00:00 | _result | 1 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 14:40:53.468000+00:00 | 13.0 | depth |
| 2023-02-19 25:34.220000+00:00 | _result | 2 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 16:25:34.220000+00:00 | https://earthquake.usgs.gov/earthquakes/feed/v... | detail |
| 2023-02-19 30:17.786000+00:00 | _result | 3 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 21:30:17.786000+00:00 | 0.0 | depth |
| 2023-02-19 57:10.963000+00:00 | _result | 4 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 17:57:10.963000+00:00 | ,ak0232azc1s8, | ids |



| measurement | code | id | magType | net | title |
|---|---|---|---|---|---|
| earthquake | 0232az01kz | ak0232az01kz | ml | ak | M 1.5 - 53 km W of Anchor Point, Alaska |
| earthquake | 0232axg3xe | ak0232axg3xe | ml | ak | M 1.5 - 32 km WSW of Cantwell, Alaska |
| earthquake | 40416424 | ci40416424 | ml | ci | M 1.5 - 6km ENE of Moreno Valley, CA |
| earthquake | 0232b1jx8y | ak0232b1jx8y | ml | ak | M 1.4 - Central Alaska |
| earthquake | 0232azc1s8 | ak0232azc1s8 | ml | ak | M 2.9 - 239 km SE of Chiniak, Alaska |

## Drop the columns that contains same value in it.

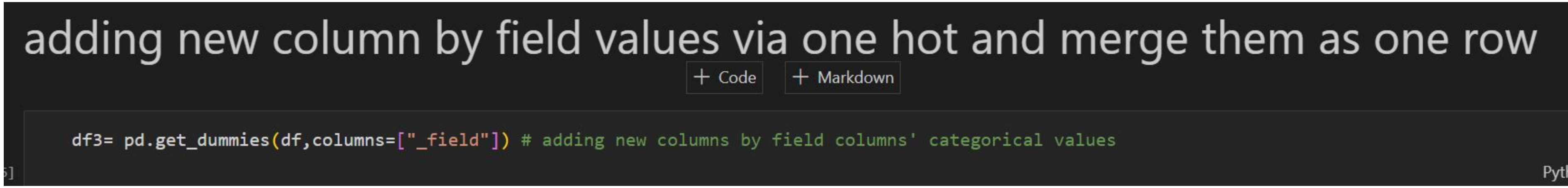


drop the columns

```
df=df.drop("_measurement", axis=1)
df=df.drop("result", axis=1)
df=df.drop("table", axis=1)
```

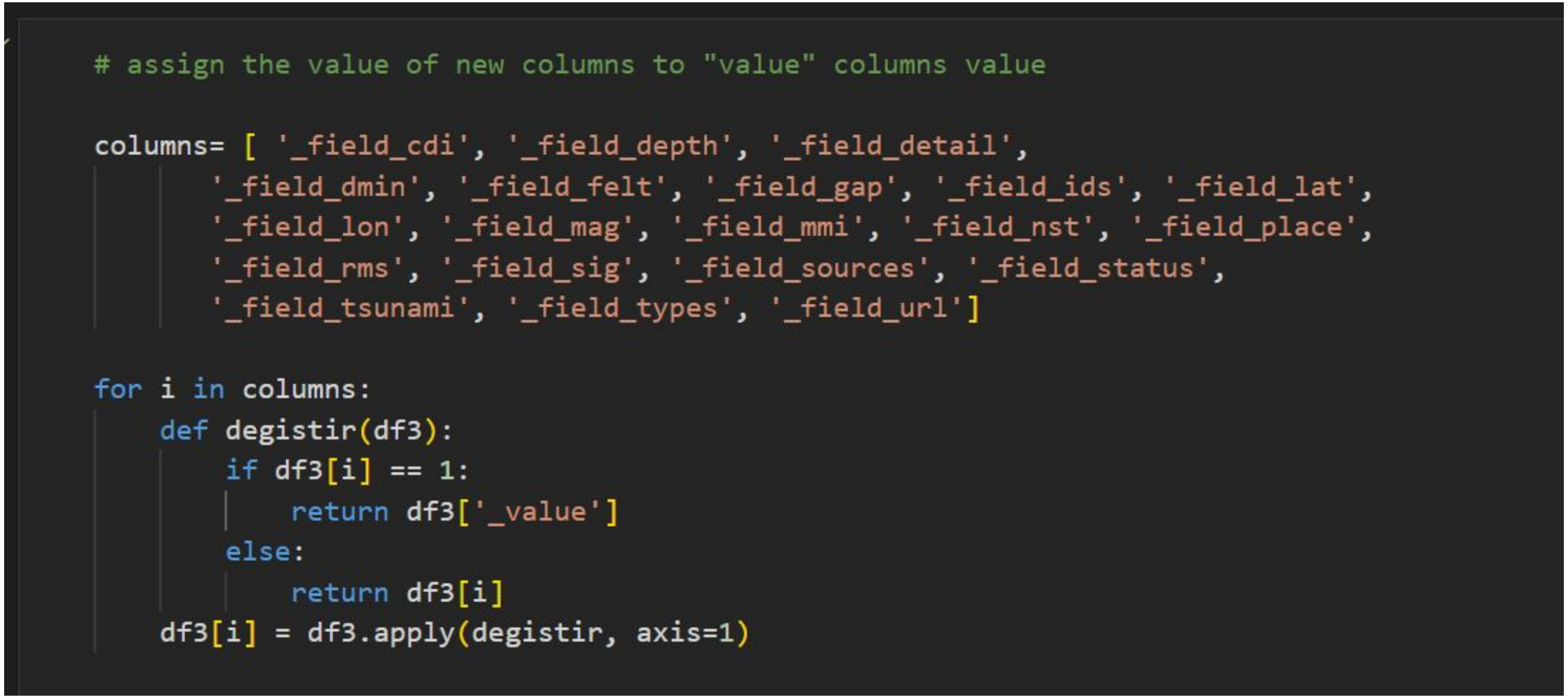
0]

one hot process to add +1 column for any kind of categorical data in “_field” column


adding new column by field values via one hot and merge them as one row

```
df3= pd.get_dummies(df,columns=["_field"]) # adding new columns by field columns' categorical values
```

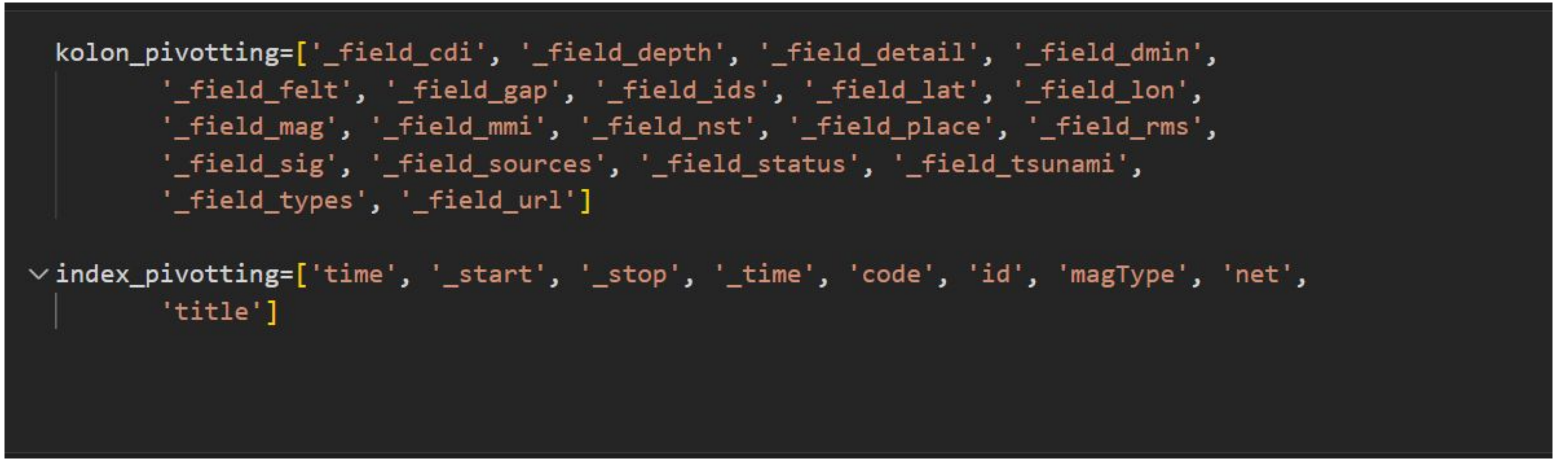
## The line "1" in the relevant line of the columns obtained with one hot has been replaced with "value" in the same line



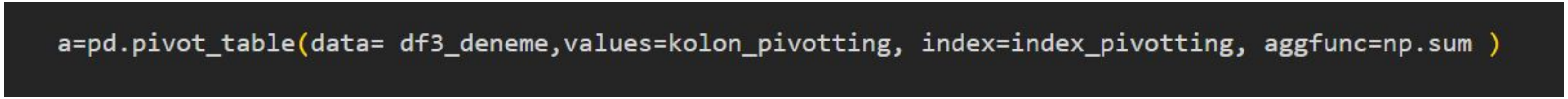
```
# assign the value of new columns to "value" columns value

columns= [ '_field_cdi', '_field_depth', '_field_detail',
        '_field_dmin', '_field_felt', '_field_gap', '_field_ids', '_field_lat',
        '_field_lon', '_field_mag', '_field_mmi', '_field_nst', '_field_place',
        '_field_rms', '_field_sig', '_field_sources', '_field_status',
        '_field_tsunami', '_field_types', '_field_url']

for i in columns:
    def degistir(df3):
        if df3[i] == 1:
            return df3['_value']
        else:
            return df3[i]
    df3[i] = df3.apply(degistir, axis=1)
```

# pivot table

Pivot table used to avoid data duplication

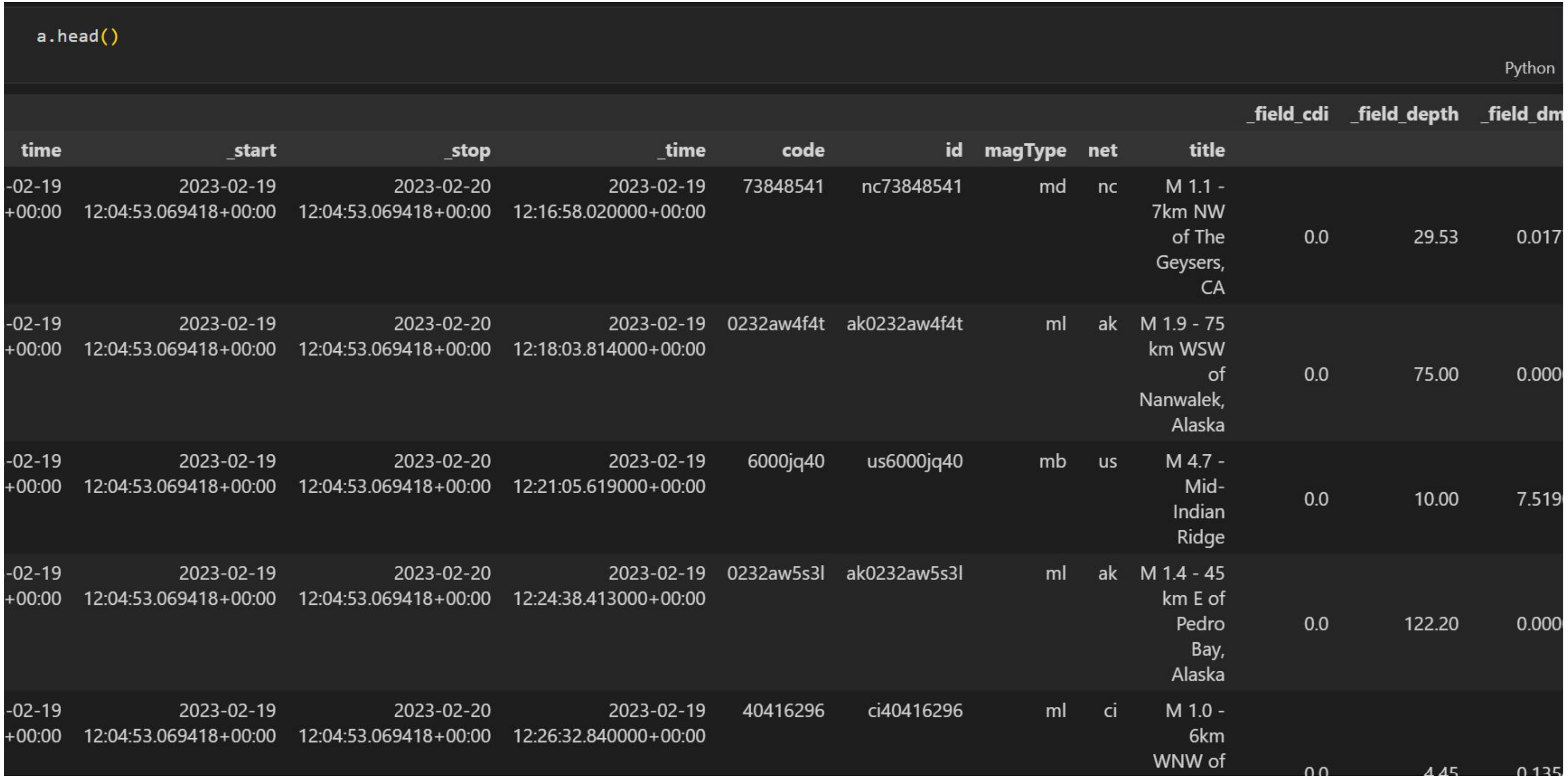


```
kolon_pivotting=['_field_cdi', '_field_depth', '_field_detail', '_field_dmin',
       '_field_felt', '_field_gap', '_field_ids', '_field_lat', '_field_lon',
       '_field_mag', '_field_mmi', '_field_nst', '_field_place', '_field_rms',
       '_field_sig', '_field_sources', '_field_status', '_field_tsunami',
       '_field_types', '_field_url']

index_pivotting=['time', '_start', '_stop', '_time', 'code', 'id', 'magType', 'net',
       'title']
```



```
a=pd.pivot_table(data= df3_deneme,values=kolon_pivotting, index=index_pivotting, aggfunc=np.sum )
```

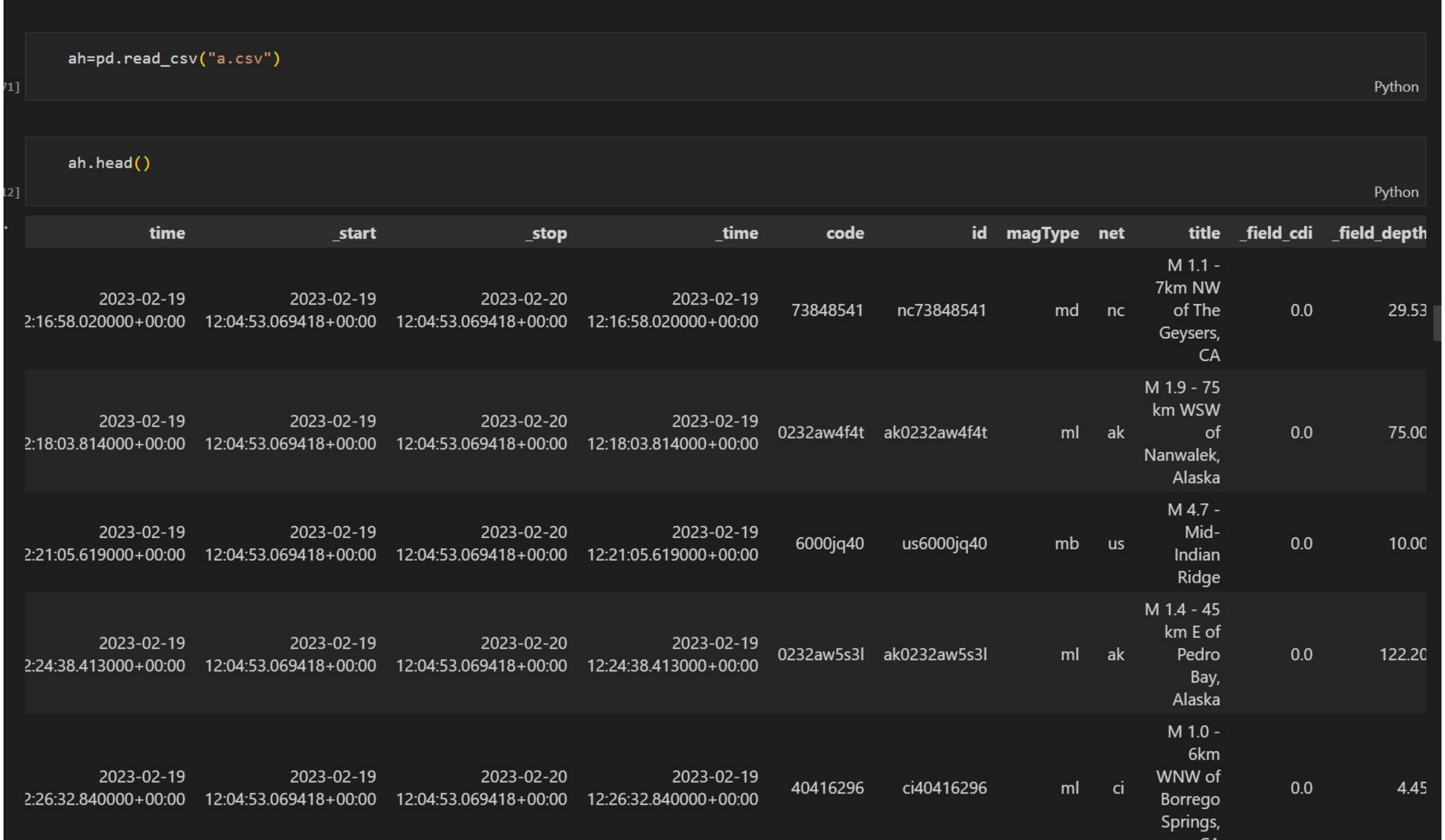
after the pivot table process, it seems there is difference. some of the columns seems indexes, others are columns. to avoid this, we’re going to save as csv file then read it again so that our indexes will be column here



```
a.head()
```

Python

| | | | | | | | | | _field_cdi | _field_depth | _field_dm |
|---|---|---|---|---|---|---|---|---|---|---|---|
| time | _start | _stop | _time | code | id | magType | net | title | | | |
| -02-19 +00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:16:58.020000+00:00 | 73848541 | nc73848541 | md | nc | M 1.1 - 7km NW of The Geysers, CA | 0.0 | 29.53 | 0.017 |
| -02-19 +00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:18:03.814000+00:00 | 0232aw4f4t | ak0232aw4f4t | ml | ak | M 1.9 - 75 km WSW of Nanwalek, Alaska | 0.0 | 75.00 | 0.000 |
| -02-19 +00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:21:05.619000+00:00 | 6000jq40 | us6000jq40 | mb | us | M 4.7 - Mid-Indian Ridge | 0.0 | 10.00 | 7.519 |
| -02-19 +00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:24:38.413000+00:00 | 0232aw5s3l | ak0232aw5s3l | ml | ak | M 1.4 - 45 km E of Pedro Bay, Alaska | 0.0 | 122.20 | 0.000 |
| -02-19 +00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:26:32.840000+00:00 | 40416296 | ci40416296 | ml | ci | M 1.0 - 6km WNW of | 0.0 | 4.45 | 0.135 |



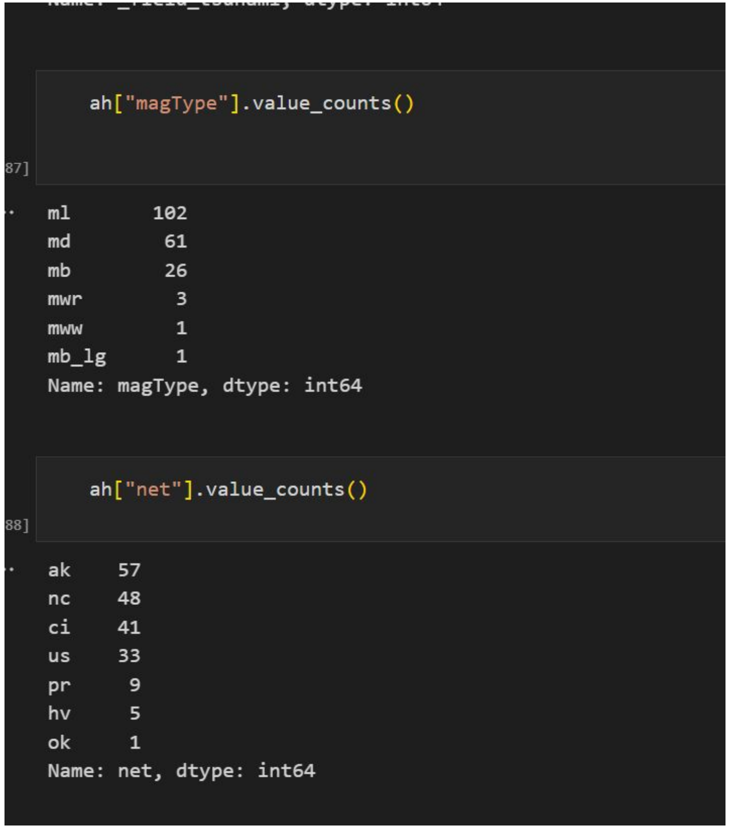
```
ah=pd.read_csv("a.csv")
```

71] Python

```
ah.head()
```

12] Python

| time | _start | _stop | _time | code | id | magType | net | title | _field_cdi | _field_depth |
|---|---|---|---|---|---|---|---|---|---|---|
| 2023-02-19 2:16:58.020000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:16:58.020000+00:00 | 73848541 | nc73848541 | md | nc | M 1.1 - 7km NW of The Geysers, CA | 0.0 | 29.53 |
| 2023-02-19 2:18:03.814000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:18:03.814000+00:00 | 0232aw4f4t | ak0232aw4f4t | ml | ak | M 1.9 - 75 km WSW of Nanwalek, Alaska | 0.0 | 75.00 |
| 2023-02-19 2:21:05.619000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:21:05.619000+00:00 | 6000jq40 | us6000jq40 | mb | us | M 4.7 - Mid-Indian Ridge | 0.0 | 10.00 |
| 2023-02-19 2:24:38.413000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:24:38.413000+00:00 | 0232aw5s3l | ak0232aw5s3l | ml | ak | M 1.4 - 45 km E of Pedro Bay, Alaska | 0.0 | 122.20 |
| 2023-02-19 2:26:32.840000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:26:32.840000+00:00 | 40416296 | ci40416296 | ml | ci | M 1.0 - 6km WNW of Borrego Springs, | 0.0 | 4.45 |



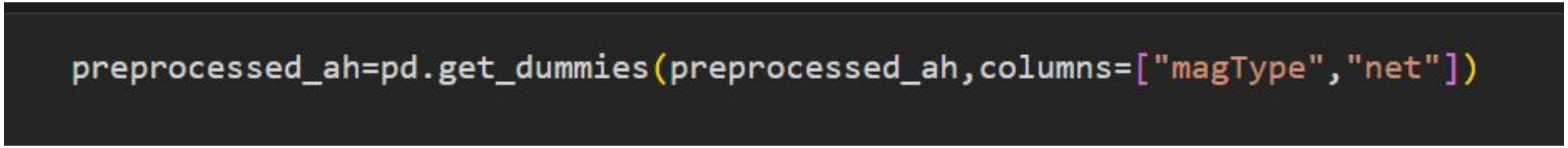
```
ah["magType"].value_counts()
```

```
ml        102
md         61
mb         26
mwr         3
mww         1
mb_lg       1
Name: magType, dtype: int64
```

```
ah["net"].value_counts()
```

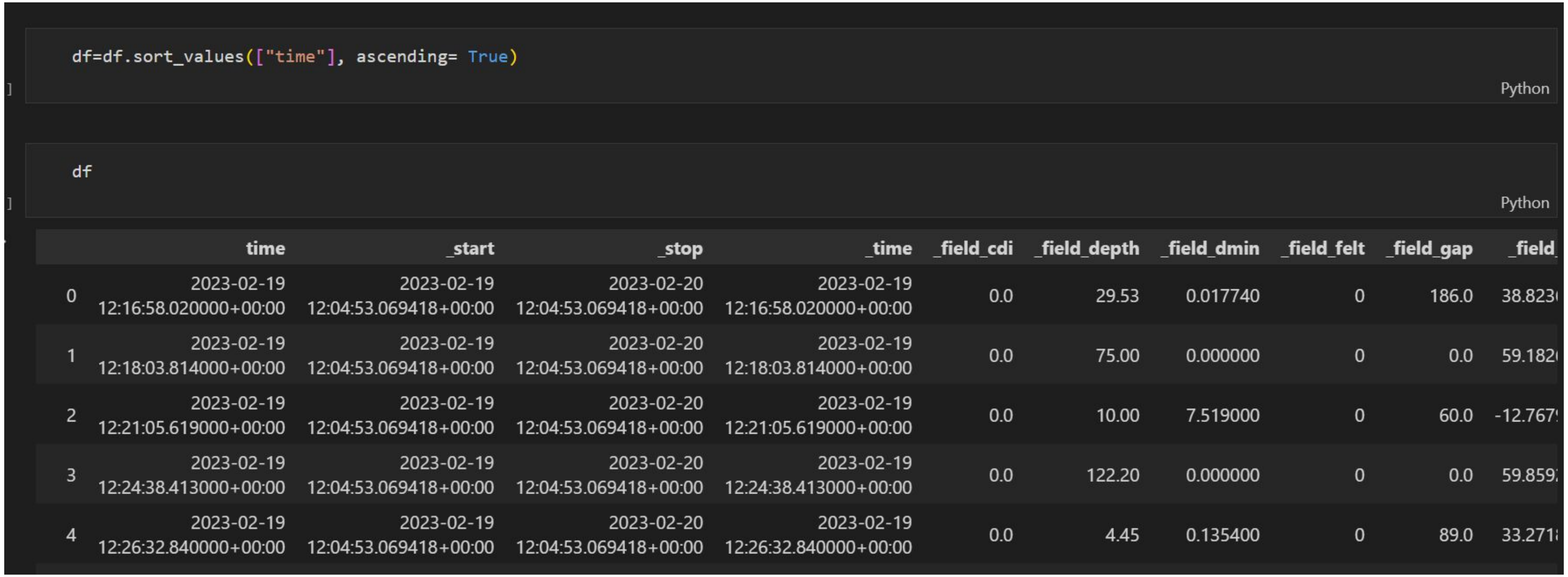
```
ak    57
nc    48
ci    41
us    33
pr     9
hv     5
ok     1
Name: net, dtype: int64
```

## one hot process to categorical columns



```
preprocessed_ah=pd.get_dummies(preprocessed_ah,columns=["magType","net"])
```

## sorting the data ascending order
## (this point is important because this is a time series data)



```
df=df.sort_values(["time"], ascending= True)
```

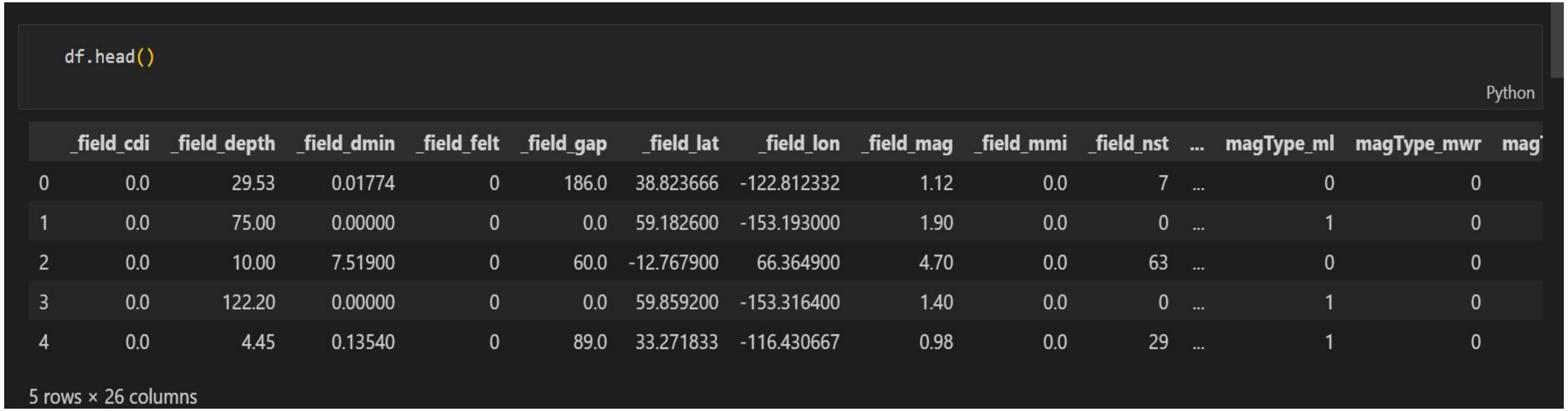
Python

```
df
```

Python

| | time | _start | _stop | _time | _field_cdi | _field_depth | _field_dmin | _field_felt | _field_gap | _field_ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2023-02-19 12:16:58.020000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:16:58.020000+00:00 | 0.0 | 29.53 | 0.017740 | 0 | 186.0 | 38.823 |
| 1 | 2023-02-19 12:18:03.814000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:18:03.814000+00:00 | 0.0 | 75.00 | 0.000000 | 0 | 0.0 | 59.182 |
| 2 | 2023-02-19 12:21:05.619000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:21:05.619000+00:00 | 0.0 | 10.00 | 7.519000 | 0 | 60.0 | -12.767 |
| 3 | 2023-02-19 12:24:38.413000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:24:38.413000+00:00 | 0.0 | 122.20 | 0.000000 | 0 | 0.0 | 59.859 |
| 4 | 2023-02-19 12:26:32.840000+00:00 | 2023-02-19 12:04:53.069418+00:00 | 2023-02-20 12:04:53.069418+00:00 | 2023-02-19 12:26:32.840000+00:00 | 0.0 | 4.45 | 0.135400 | 0 | 89.0 | 33.271 |

after ordering process, we drop the columns which is meaningless for our analysis and model

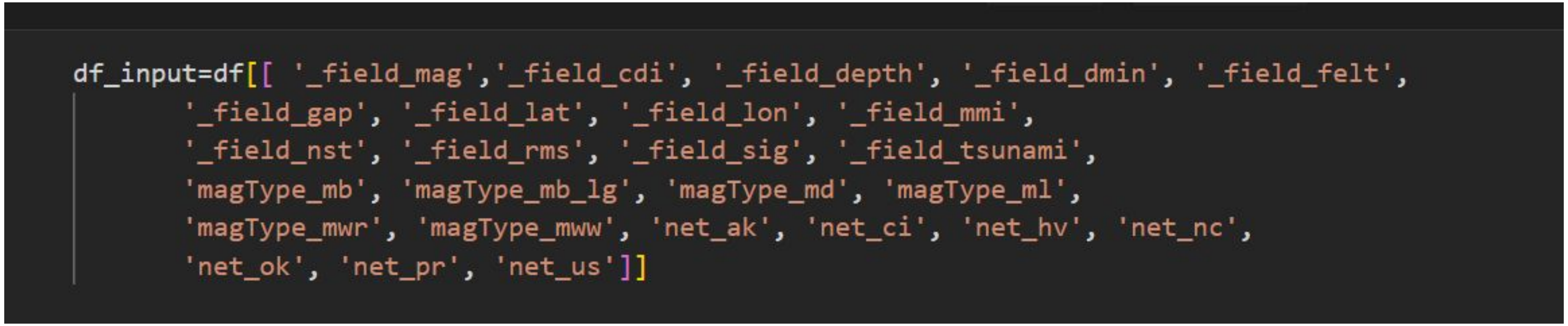


```
df.head()
```

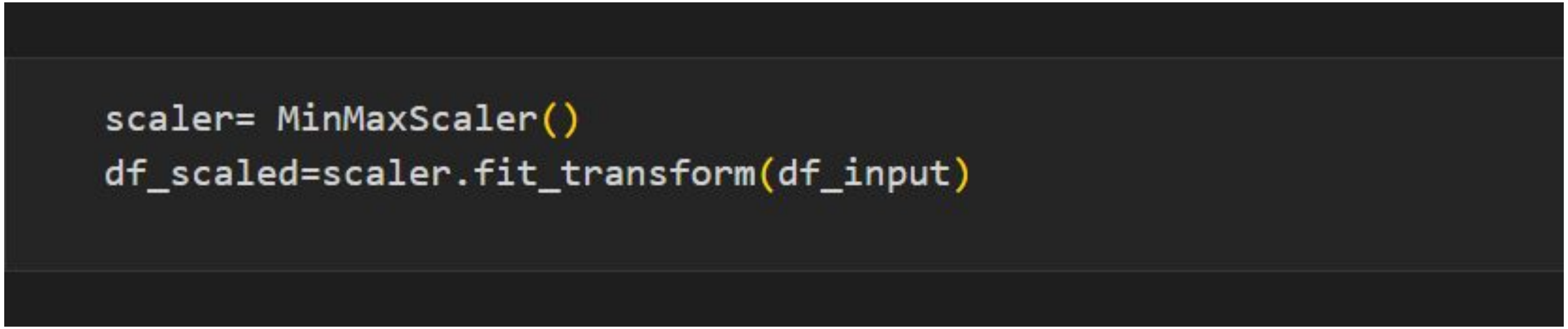
Python

```
   _field_cdi  _field_depth  _field_dmin  _field_felt  _field_gap  _field_lat   _field_lon  _field_mag  _field_mmi  _field_nst  ...  magType_ml  magType_mwr  mag
0         0.0         29.53      0.01774            0       186.0   38.823666  -122.812332        1.12         0.0           7  ...           0            0
1         0.0         75.00      0.00000            0         0.0   59.182600  -153.193000        1.90         0.0           0  ...           1            0
2         0.0         10.00      7.51900            0        60.0  -12.767900    66.364900        4.70         0.0          63  ...           0            0
3         0.0        122.20      0.00000            0         0.0   59.859200  -153.316400        1.40         0.0           0  ...           1            0
4         0.0          4.45      0.13540            0        89.0   33.271833  -116.430667        0.98         0.0          29  ...           1            0

5 rows × 26 columns
```

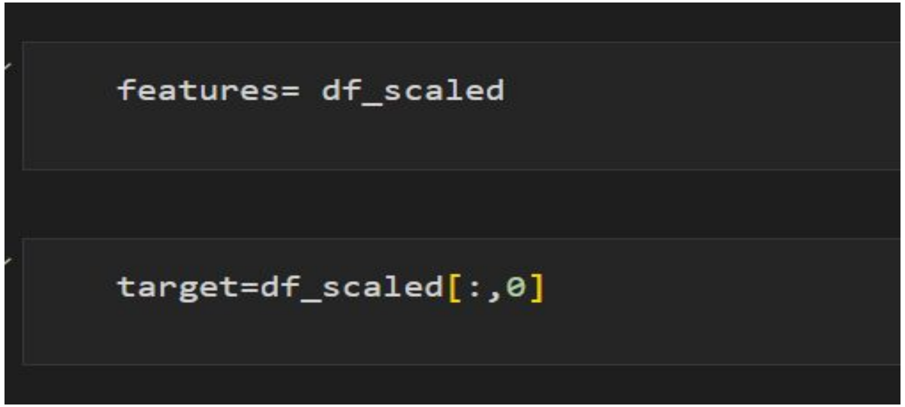
## min-max scaler



```
df_input=df[[ '_field_mag','_field_cdi', '_field_depth', '_field_dmin', '_field_felt',
        '_field_gap', '_field_lat', '_field_lon', '_field_mmi',
        '_field_nst', '_field_rms', '_field_sig', '_field_tsunami',
        'magType_mb', 'magType_mb_lg', 'magType_md', 'magType_ml',
        'magType_mwr', 'magType_mww', 'net_ak', 'net_ci', 'net_hv', 'net_nc',
        'net_ok', 'net_pr', 'net_us']]
```

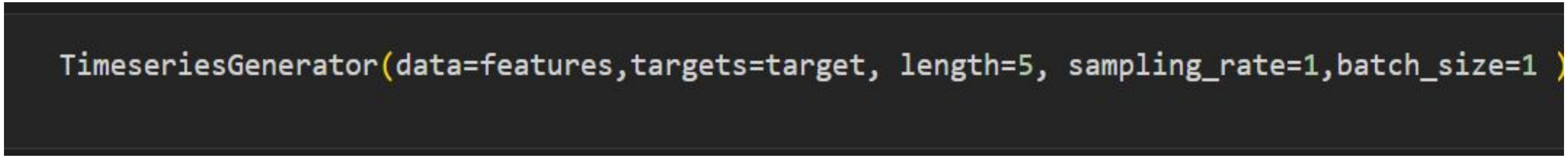


```
scaler= MinMaxScaler()
df_scaled=scaler.fit_transform(df_input)
```



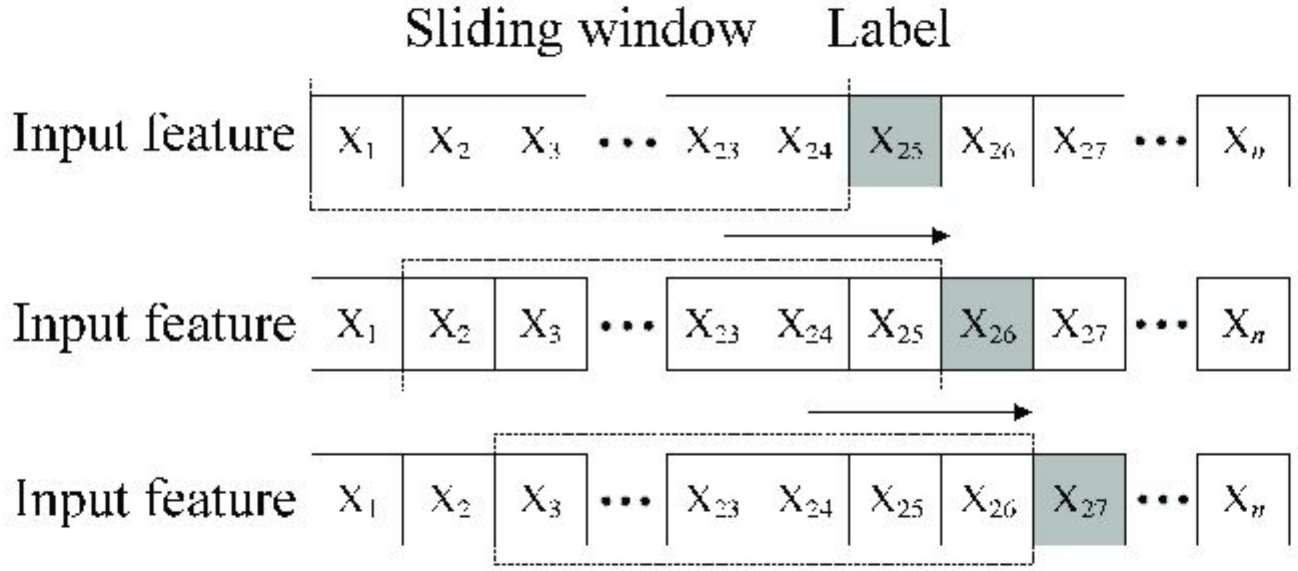
```
features= df_scaled
```

```
target=df_scaled[:,0]
```

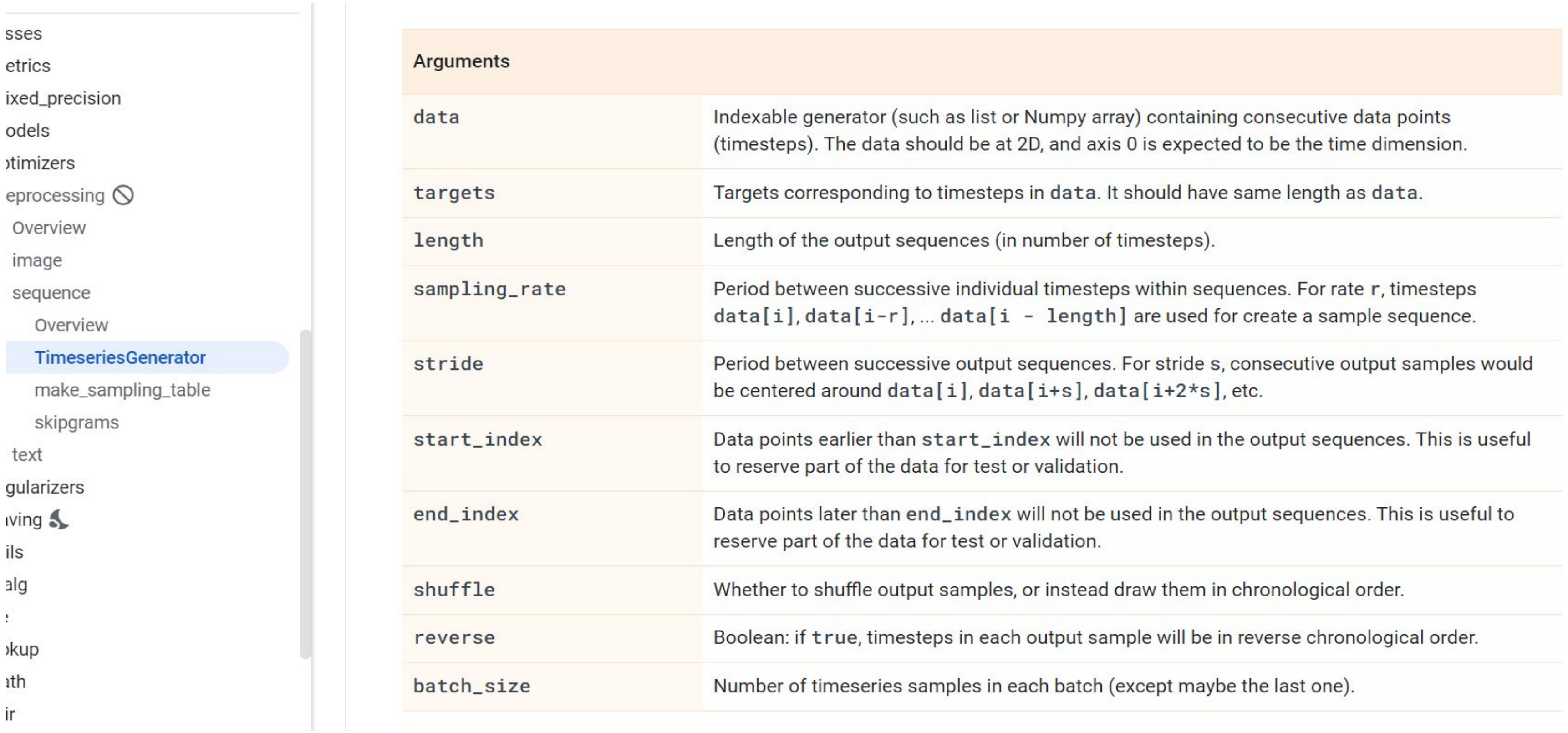


```
TimeseriesGenerator(data=features,targets=target, length=5, sampling_rate=1,batch_size=1 )
```

windowing techniques

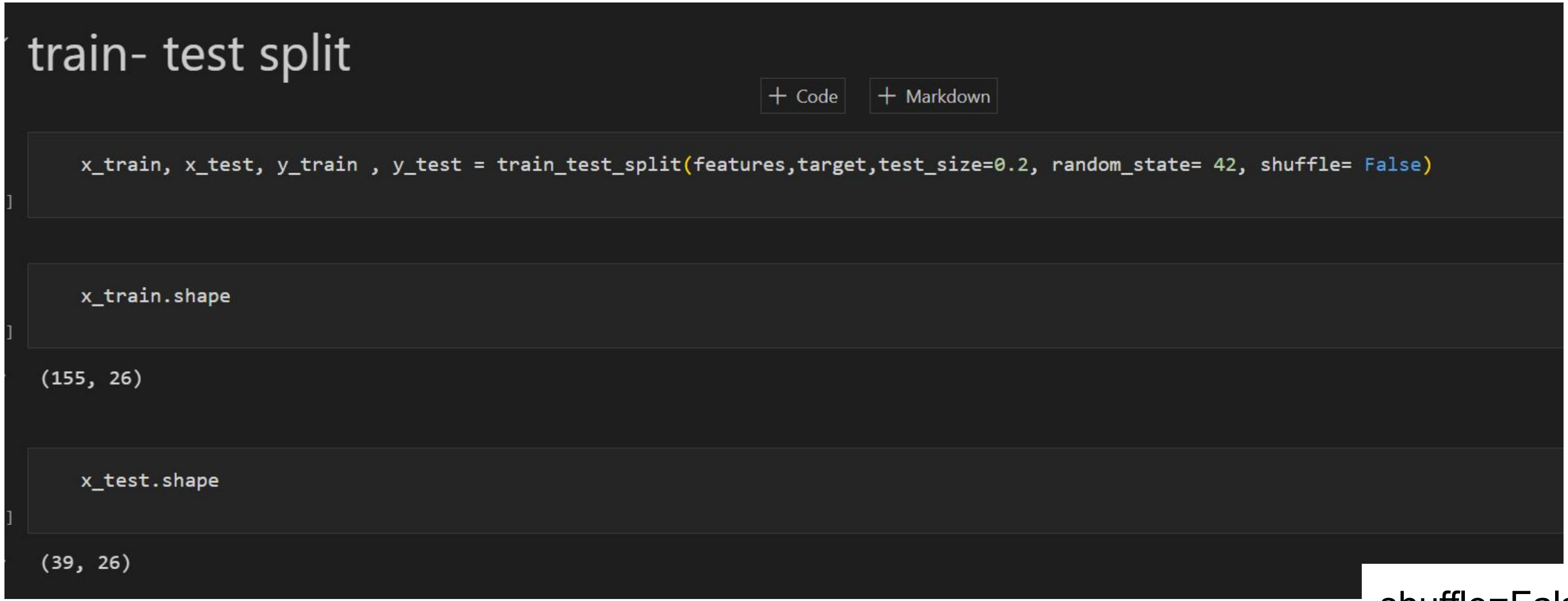


# Time Series Generator parameters



| Arguments | |
|---|---|
| `data` | Indexable generator (such as list or Numpy array) containing consecutive data points (timesteps). The data should be at 2D, and axis 0 is expected to be the time dimension. |
| `targets` | Targets corresponding to timesteps in `data`. It should have same length as `data`. |
| `length` | Length of the output sequences (in number of timesteps). |
| `sampling_rate` | Period between successive individual timesteps within sequences. For rate `r`, timesteps `data[i]`, `data[i-r]`, ... `data[i - length]` are used for create a sample sequence. |
| `stride` | Period between successive output sequences. For stride `s`, consecutive output samples would be centered around `data[i]`, `data[i+s]`, `data[i+2*s]`, etc. |
| `start_index` | Data points earlier than `start_index` will not be used in the output sequences. This is useful to reserve part of the data for test or validation. |
| `end_index` | Data points later than `end_index` will not be used in the output sequences. This is useful to reserve part of the data for test or validation. |
| `shuffle` | Whether to shuffle output samples, or instead draw them in chronological order. |
| `reverse` | Boolean: if `true`, timesteps in each output sample will be in reverse chronological order. |
| `batch_size` | Number of timeseries samples in each batch (except maybe the last one). |

# train-test split



```
x_train, x_test, y_train , y_test = train_test_split(features,target,test_size=0.2, random_state= 42, shuffle= False)
```

```
x_train.shape
```

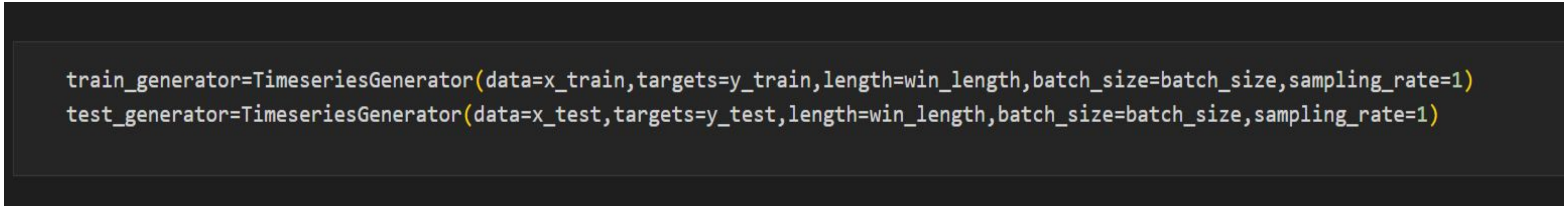
```
(155, 26)
```

```
x_test.shape
```

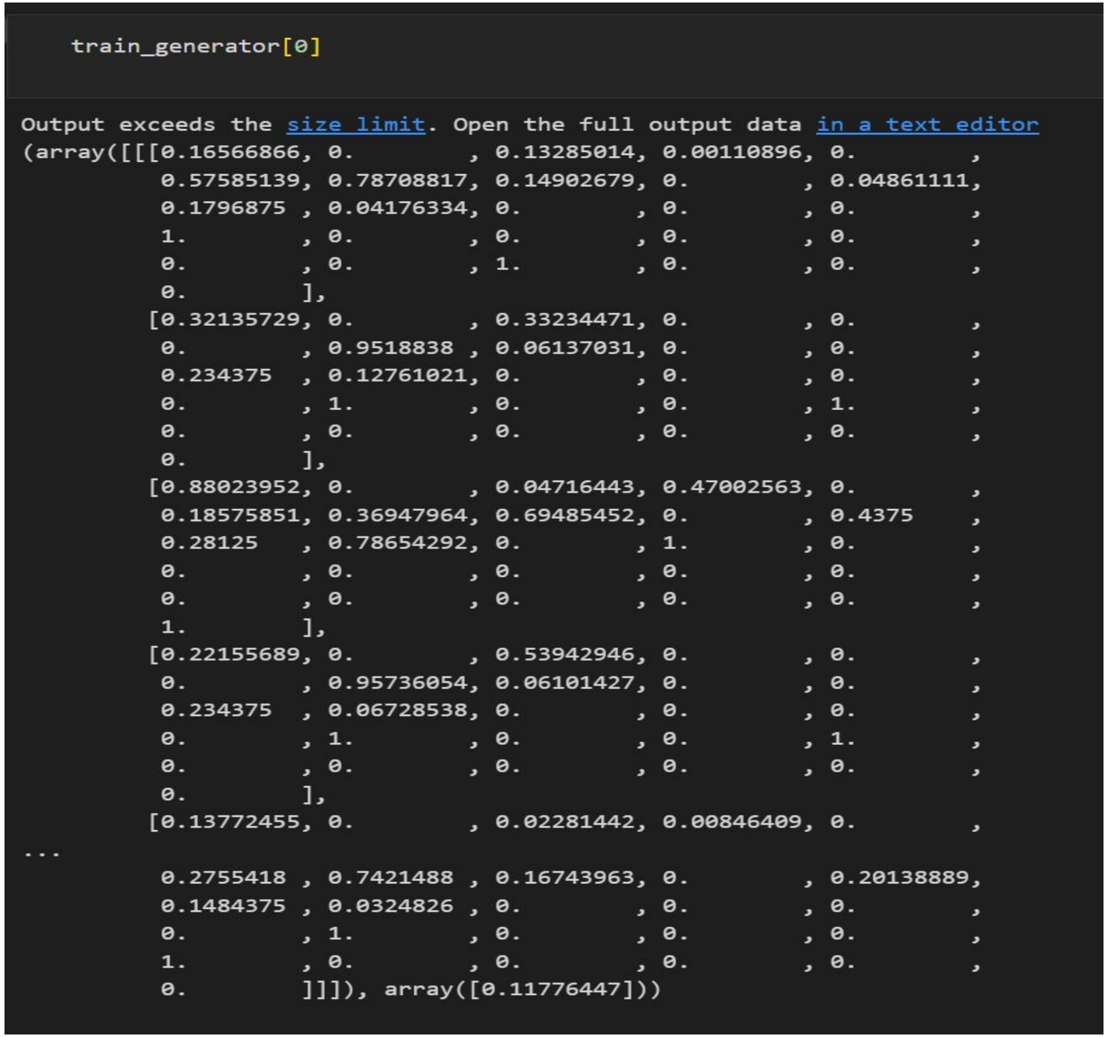
```
(39, 26)
```

shuffle=False!!

## train_generator - test_generator

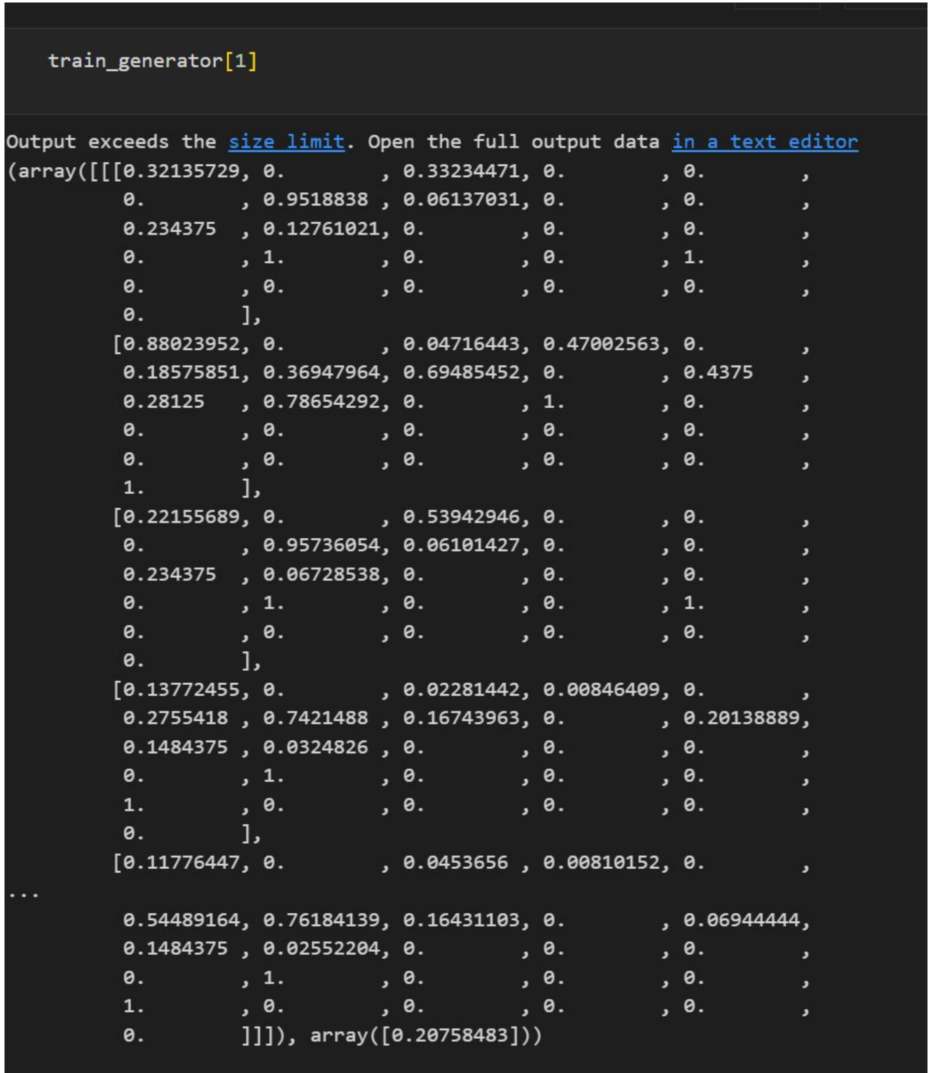


```
train_generator=TimeseriesGenerator(data=x_train,targets=y_train,length=win_length,batch_size=batch_size,sampling_rate=1)
test_generator=TimeseriesGenerator(data=x_test,targets=y_test,length=win_length,batch_size=batch_size,sampling_rate=1)
```



```
    train_generator[0]
```

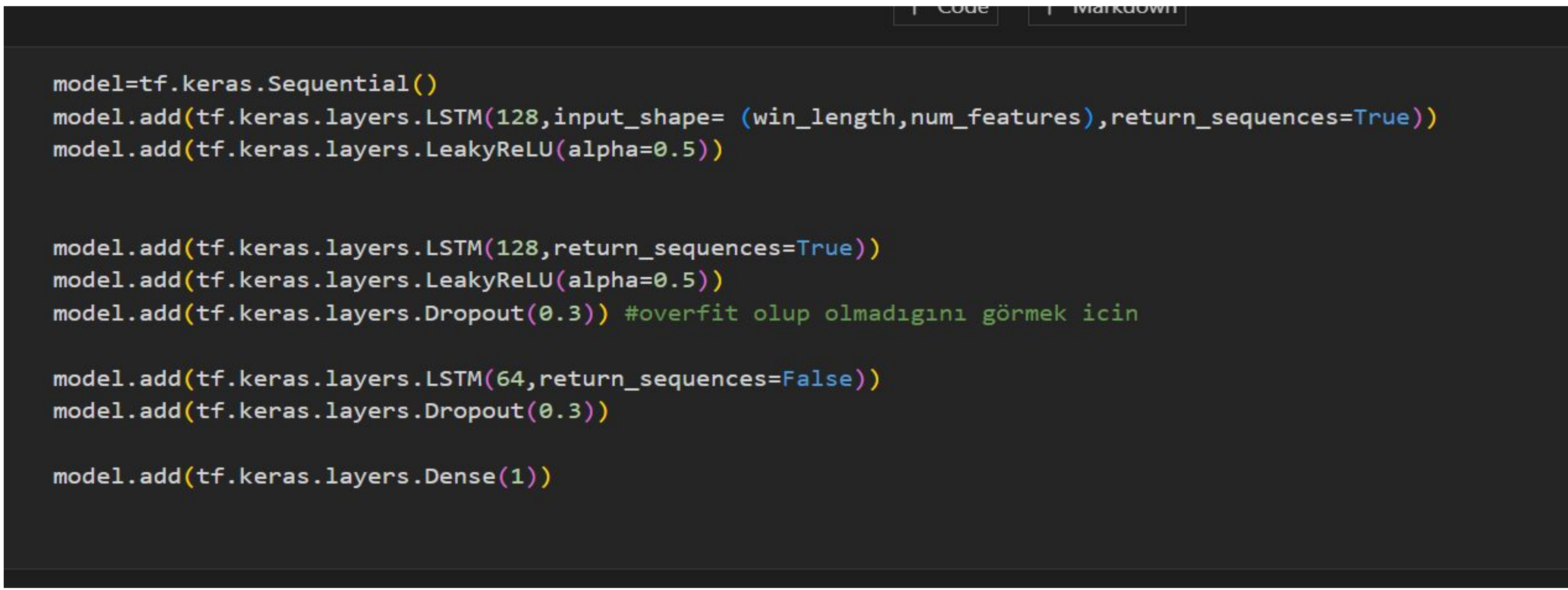
```
Output exceeds the size limit. Open the full output data in a text editor
(array([[[0.16566866, 0.        , 0.13285014, 0.00110896, 0.        ,
          0.57585139, 0.78708817, 0.14902679, 0.        , 0.04861111,
          0.1796875 , 0.04176334, 0.        , 0.        , 0.        ,
          1.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 1.        , 0.        , 0.        ,
          0.        ],
         [0.32135729, 0.        , 0.33234471, 0.        , 0.        ,
          0.        , 0.9518838 , 0.06137031, 0.        , 0.        ,
          0.234375  , 0.12761021, 0.        , 0.        , 0.        ,
          0.        , 1.        , 0.        , 0.        , 1.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        ],
         [0.88023952, 0.        , 0.04716443, 0.47002563, 0.        ,
          0.18575851, 0.36947964, 0.69485452, 0.        , 0.4375    ,
          0.28125   , 0.78654292, 0.        , 1.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          1.        ],
         [0.22155689, 0.        , 0.53942946, 0.        , 0.        ,
          0.        , 0.95736054, 0.06101427, 0.        , 0.        ,
          0.234375  , 0.06728538, 0.        , 0.        , 0.        ,
          0.        , 1.        , 0.        , 0.        , 1.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        ],
         [0.13772455, 0.        , 0.02281442, 0.00846409, 0.        ,
...
          0.2755418 , 0.7421488 , 0.16743963, 0.        , 0.20138889,
          0.1484375 , 0.0324826 , 0.        , 0.        , 0.        ,
          0.        , 1.        , 0.        , 0.        , 0.        ,
          1.        , 0.        , 0.        , 0.        , 0.        ,
          0.        ]]]), array([0.11776447]))
```


```
train_generator[1]
```

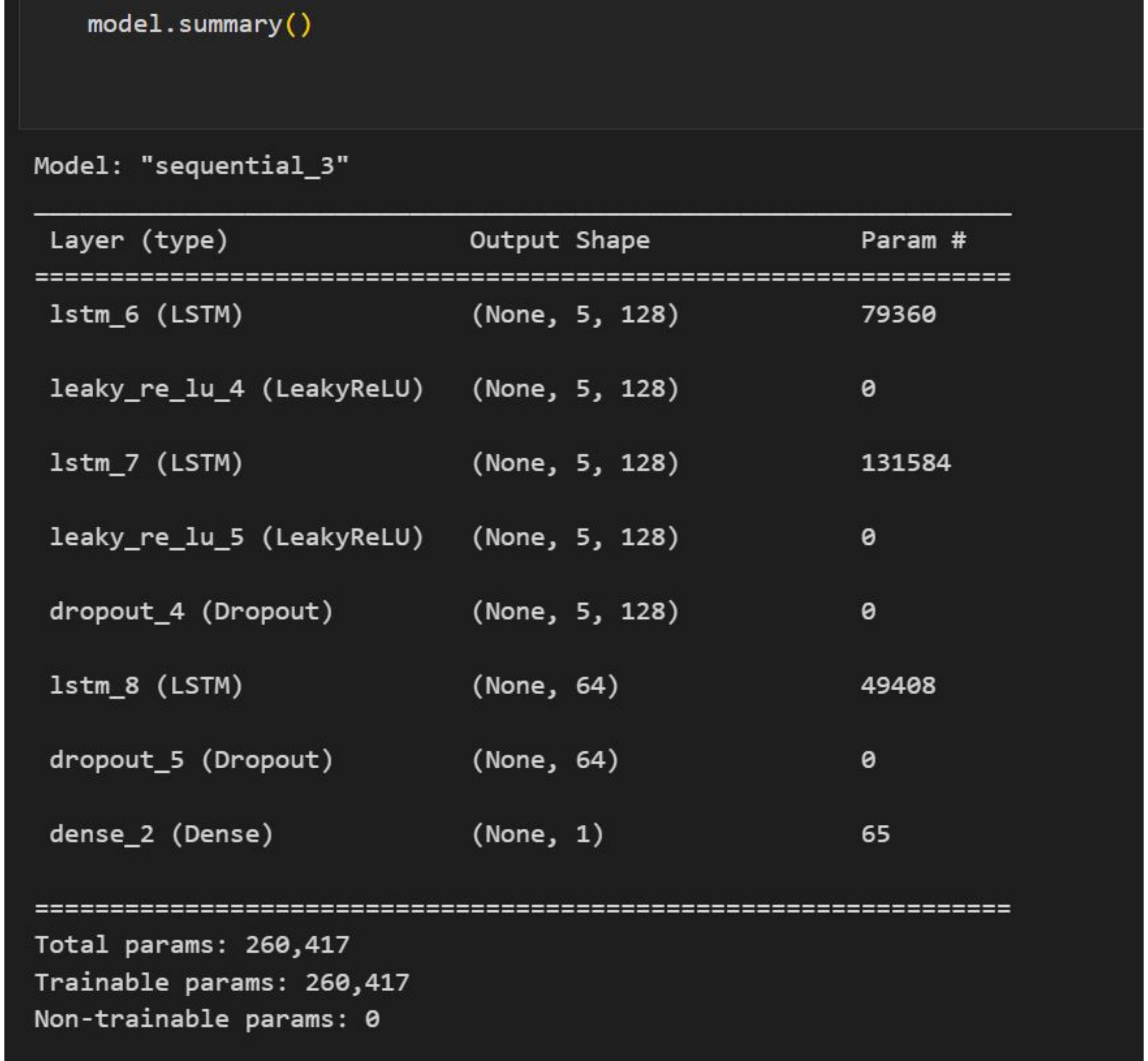
```
Output exceeds the size limit. Open the full output data in a text editor
(array([[[0.32135729, 0.        , 0.33234471, 0.        , 0.        ,
          0.        , 0.9518838 , 0.06137031, 0.        , 0.        ,
          0.234375  , 0.12761021, 0.        , 0.        , 0.        ,
          0.        , 1.        , 0.        , 0.        , 1.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        ],
         [0.88023952, 0.        , 0.04716443, 0.47002563, 0.        ,
          0.18575851, 0.36947964, 0.69485452, 0.        , 0.4375    ,
          0.28125   , 0.78654292, 0.        , 1.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          1.        ],
         [0.22155689, 0.        , 0.53942946, 0.        , 0.        ,
          0.        , 0.95736054, 0.06101427, 0.        , 0.        ,
          0.234375  , 0.06728538, 0.        , 0.        , 0.        ,
          0.        , 1.        , 0.        , 0.        , 1.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        ],
         [0.13772455, 0.        , 0.02281442, 0.00846409, 0.        ,
          0.2755418 , 0.7421488 , 0.16743963, 0.        , 0.20138889,
          0.1484375 , 0.0324826 , 0.        , 0.        , 0.        ,
          0.        , 1.        , 0.        , 0.        , 0.        ,
          1.        , 0.        , 0.        , 0.        , 0.        ,
          0.        ],
         [0.11776447, 0.        , 0.0453656 , 0.00810152, 0.        ,
...
          0.54489164, 0.76184139, 0.16431103, 0.        , 0.06944444,
          0.1484375 , 0.02552204, 0.        , 0.        , 0.        ,
          0.        , 1.        , 0.        , 0.        , 0.        ,
          1.        , 0.        , 0.        , 0.        , 0.        ,
          0.        ]]]), array([0.20758483]))
```

it seems, windowed process has done

## model architecture

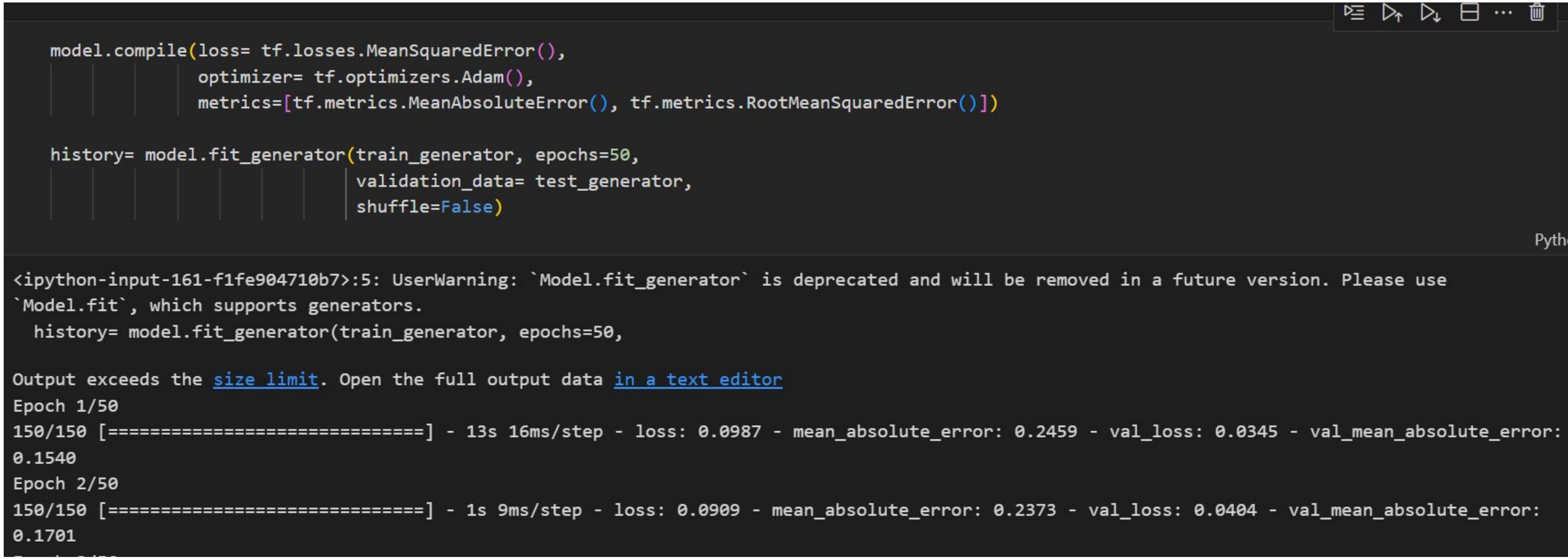


```
model=tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128,input_shape= (win_length,num_features),return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))


model.add(tf.keras.layers.LSTM(128,return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3)) #overfit olup olmadıgını görmek icin

model.add(tf.keras.layers.LSTM(64,return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Dense(1))
```



```
model.summary()
```

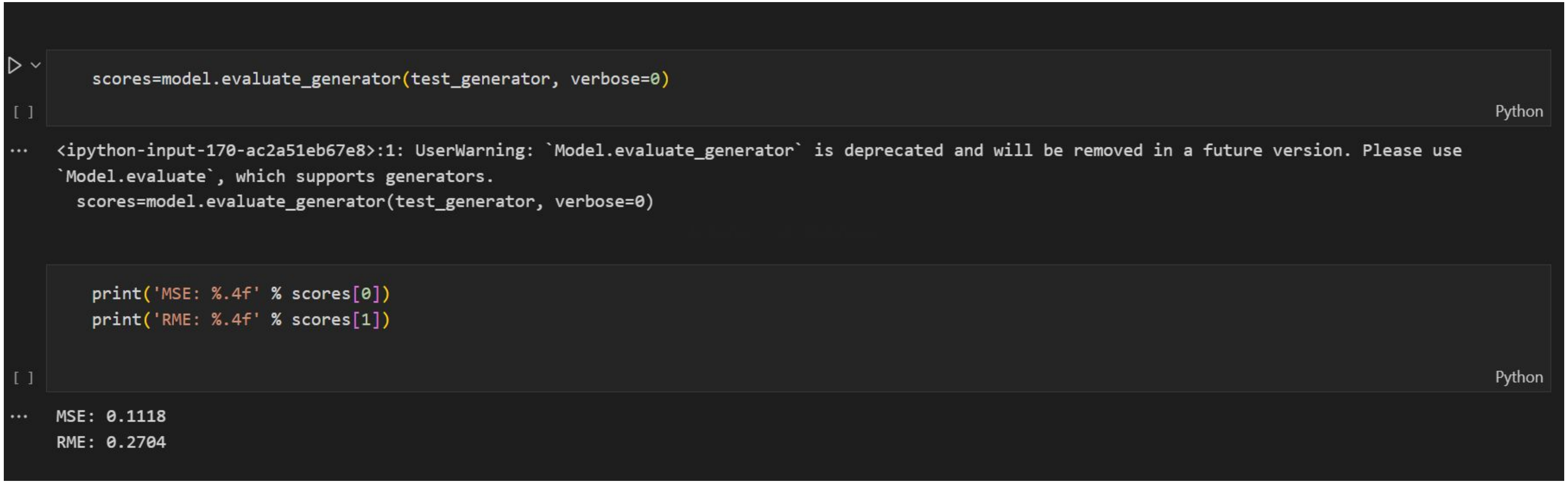
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm_6 (LSTM)               (None, 5, 128)            79360     
                                                                 
 leaky_re_lu_4 (LeakyReLU)   (None, 5, 128)            0         
                                                                 
 lstm_7 (LSTM)               (None, 5, 128)            131584    
                                                                 
 leaky_re_lu_5 (LeakyReLU)   (None, 5, 128)            0         
                                                                 
 dropout_4 (Dropout)         (None, 5, 128)            0         
                                                                 
 lstm_8 (LSTM)               (None, 64)                49408     
                                                                 
 dropout_5 (Dropout)         (None, 64)                0         
                                                                 
 dense_2 (Dense)             (None, 1)                 65        
                                                                 
=================================================================
Total params: 260,417
Trainable params: 260,417
Non-trainable params: 0
```



```
model.compile(loss= tf.losses.MeanSquaredError(),
              optimizer= tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError(), tf.metrics.RootMeanSquaredError()])

history= model.fit_generator(train_generator, epochs=50,
                             validation_data= test_generator,
                             shuffle=False)
```

Pyth

```
<ipython-input-161-f1fe904710b7>:5: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
`Model.fit`, which supports generators.
  history= model.fit_generator(train_generator, epochs=50,

Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/50
150/150 [==============================] - 13s 16ms/step - loss: 0.0987 - mean_absolute_error: 0.2459 - val_loss: 0.0345 - val_mean_absolute_error:
0.1540
Epoch 2/50
150/150 [==============================] - 1s 9ms/step - loss: 0.0909 - mean_absolute_error: 0.2373 - val_loss: 0.0404 - val_mean_absolute_error:
0.1701
```



```
scores=model.evaluate_generator(test_generator, verbose=0)
```

```
<ipython-input-170-ac2a51eb67e8>:1: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use
`Model.evaluate`, which supports generators.
  scores=model.evaluate_generator(test_generator, verbose=0)
```

```
print('MSE: %.4f' % scores[0])
print('RME: %.4f' % scores[1])
```

```
MSE: 0.1118
RME: 0.2704
```