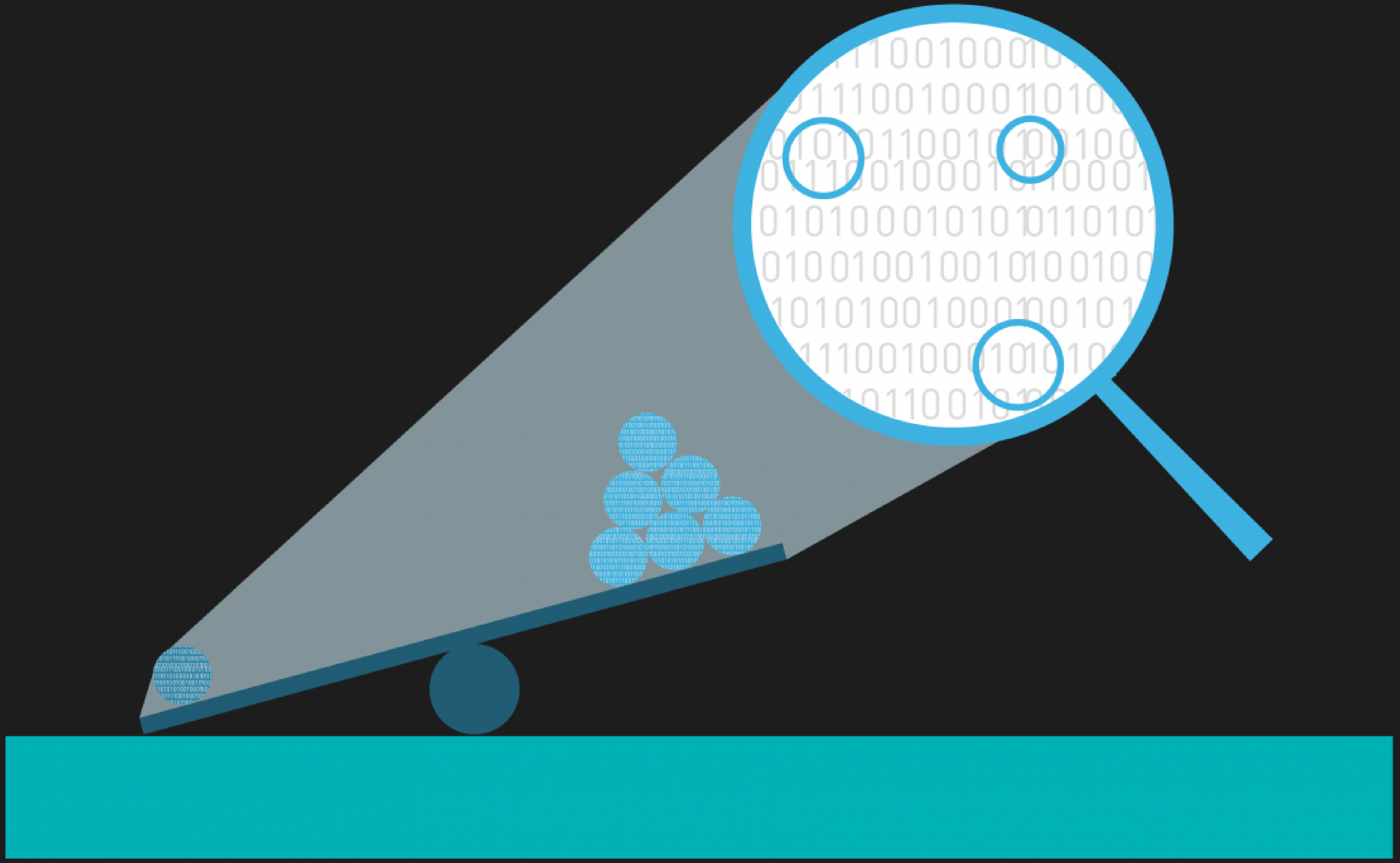
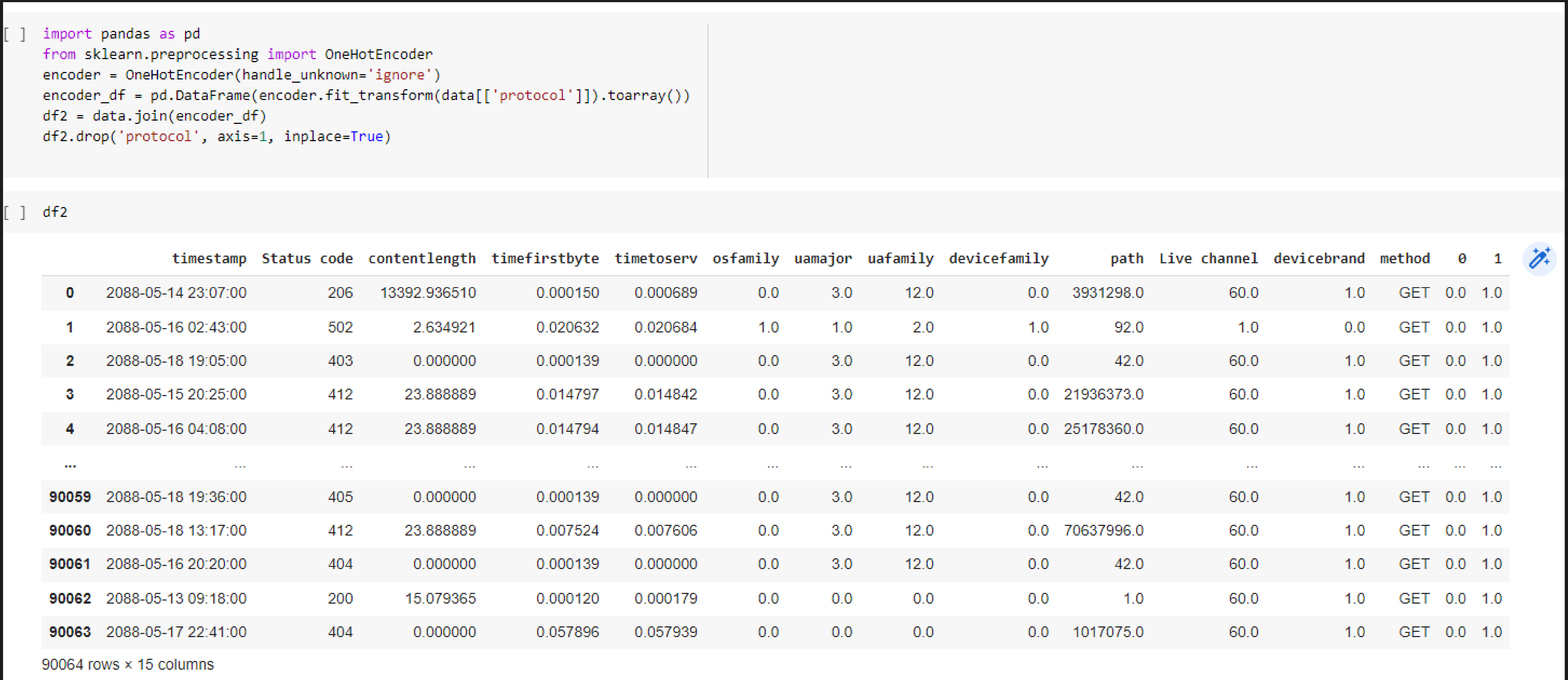
Our Mission
and Goals

Realtime Anomaly Detection with CDN



What we aim to accomplish *by the end of the term*
week 1-4 December

One hot encoder



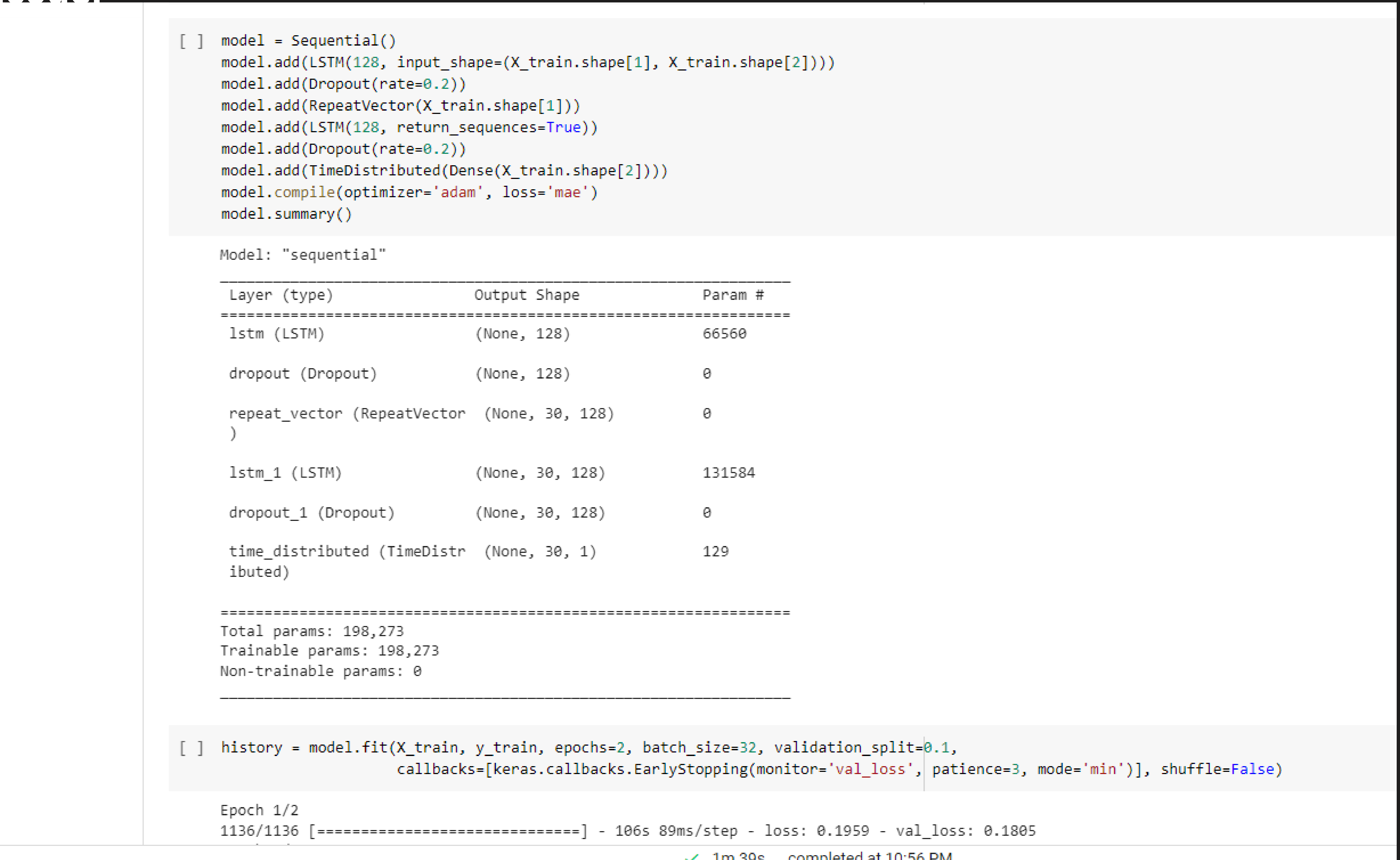
```
[ ] import pandas as pd
    from sklearn.preprocessing import OneHotEncoder
    encoder = OneHotEncoder(handle_unknown='ignore')
    encoder_df = pd.DataFrame(encoder.fit_transform(data[['protocol']]).toarray())
    df2 = data.join(encoder_df)
    df2.drop('protocol', axis=1, inplace=True)
```

```
[ ] df2
```

	timestamp	Status code	contentlength	timefirstbyte	timetoserv	osfamily	uamajor	uafamily	devicefamily	path	Live channel	devicebrand	method	0	1
0	2088-05-14 23:07:00	206	13392.936510	0.000150	0.000689	0.0	3.0	12.0	0.0	3931298.0	60.0	1.0	GET	0.0	1.0
1	2088-05-16 02:43:00	502	2.634921	0.020632	0.020684	1.0	1.0	2.0	1.0	92.0	1.0	0.0	GET	0.0	1.0
2	2088-05-18 19:05:00	403	0.000000	0.000139	0.000000	0.0	3.0	12.0	0.0	42.0	60.0	1.0	GET	0.0	1.0
3	2088-05-15 20:25:00	412	23.888889	0.014797	0.014842	0.0	3.0	12.0	0.0	21936373.0	60.0	1.0	GET	0.0	1.0
4	2088-05-16 04:08:00	412	23.888889	0.014794	0.014847	0.0	3.0	12.0	0.0	25178360.0	60.0	1.0	GET	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
90059	2088-05-18 19:36:00	405	0.000000	0.000139	0.000000	0.0	3.0	12.0	0.0	42.0	60.0	1.0	GET	0.0	1.0
90060	2088-05-18 13:17:00	412	23.888889	0.007524	0.007606	0.0	3.0	12.0	0.0	70637996.0	60.0	1.0	GET	0.0	1.0
90061	2088-05-16 20:20:00	404	0.000000	0.000139	0.000000	0.0	3.0	12.0	0.0	42.0	60.0	1.0	GET	0.0	1.0
90062	2088-05-13 09:18:00	200	15.079365	0.000120	0.000179	0.0	0.0	0.0	0.0	1.0	60.0	1.0	GET	0.0	1.0
90063	2088-05-17 22:41:00	404	0.000000	0.057896	0.057939	0.0	0.0	0.0	0.0	1017075.0	60.0	1.0	GET	0.0	1.0

90064 rows × 15 columns

Implementation of LSTM Autoencoder

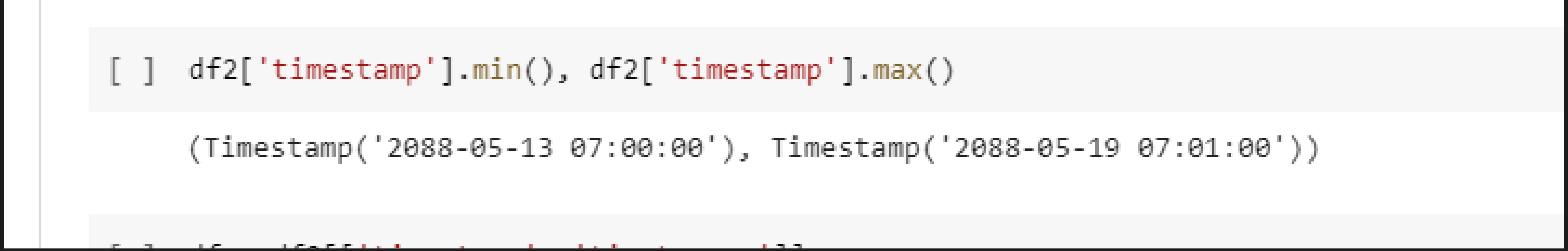


```
[ ] model = Sequential()
    model.add(LSTM(128, input_shape=(X_train.shape[1], X_train.shape[2])))
    model.add(Dropout(rate=0.2))
    model.add(RepeatVector(X_train.shape[1]))
    model.add(LSTM(128, return_sequences=True))
    model.add(Dropout(rate=0.2))
    model.add(TimeDistributed(Dense(X_train.shape[2])))
    model.compile(optimizer='adam', loss='mae')
    model.summary()
```

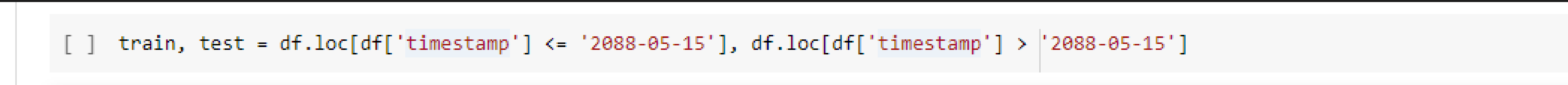
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 128)               66560

 dropout (Dropout)           (None, 128)               0

 repeat_vector (RepeatVector  (None, 30, 128)          0
 )

 lstm_1 (LSTM)               (None, 30, 128)           131584

 dropout_1 (Dropout)         (None, 30, 128)           0

 time_distributed (TimeDistr  (None, 30, 1)            129
 ibuted)

=================================================================
Total params: 198,273
Trainable params: 198,273
Non-trainable params: 0
_________________________________________________________________
```

```
[ ] history = model.fit(X_train, y_train, epochs=2, batch_size=32, validation_split=0.1,
                        callbacks=[keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, mode='min')], shuffle=False)
```

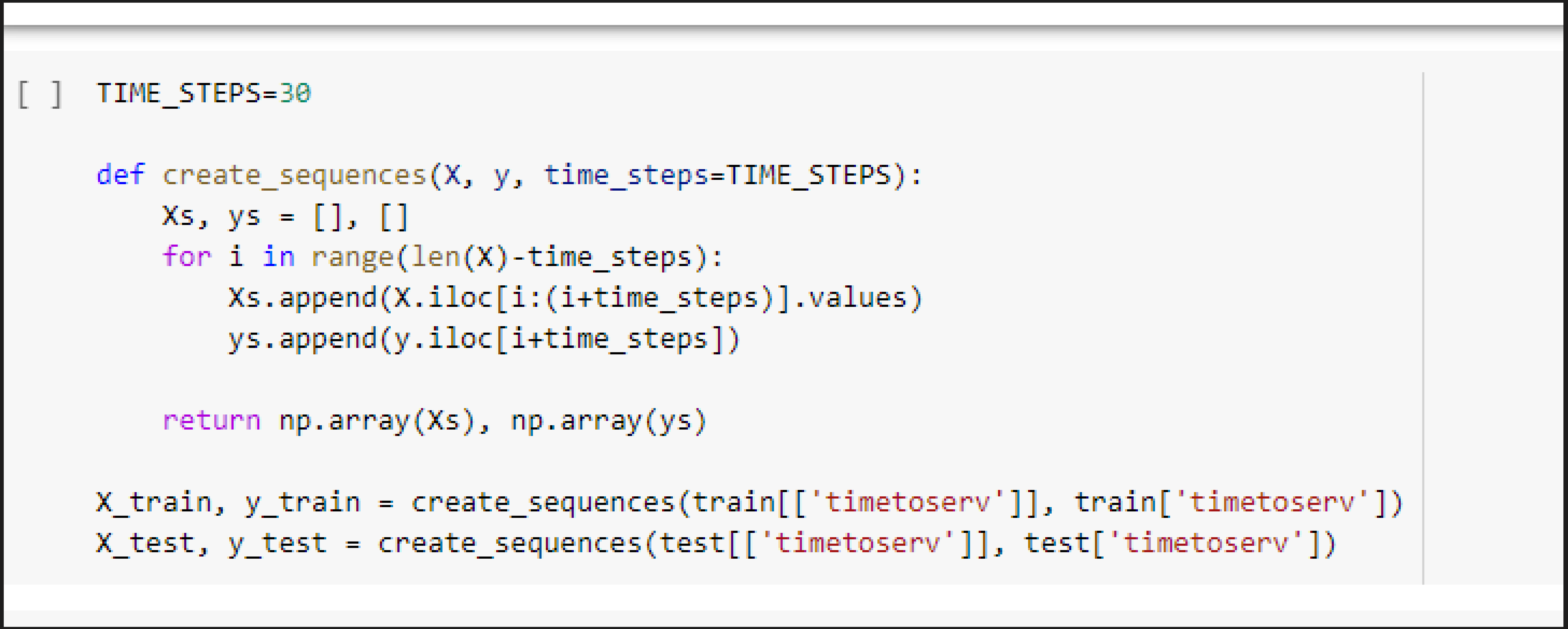
```
Epoch 1/2
1136/1136 [==============================] - 106s 89ms/step - loss: 0.1959 - val_loss: 0.1805
```



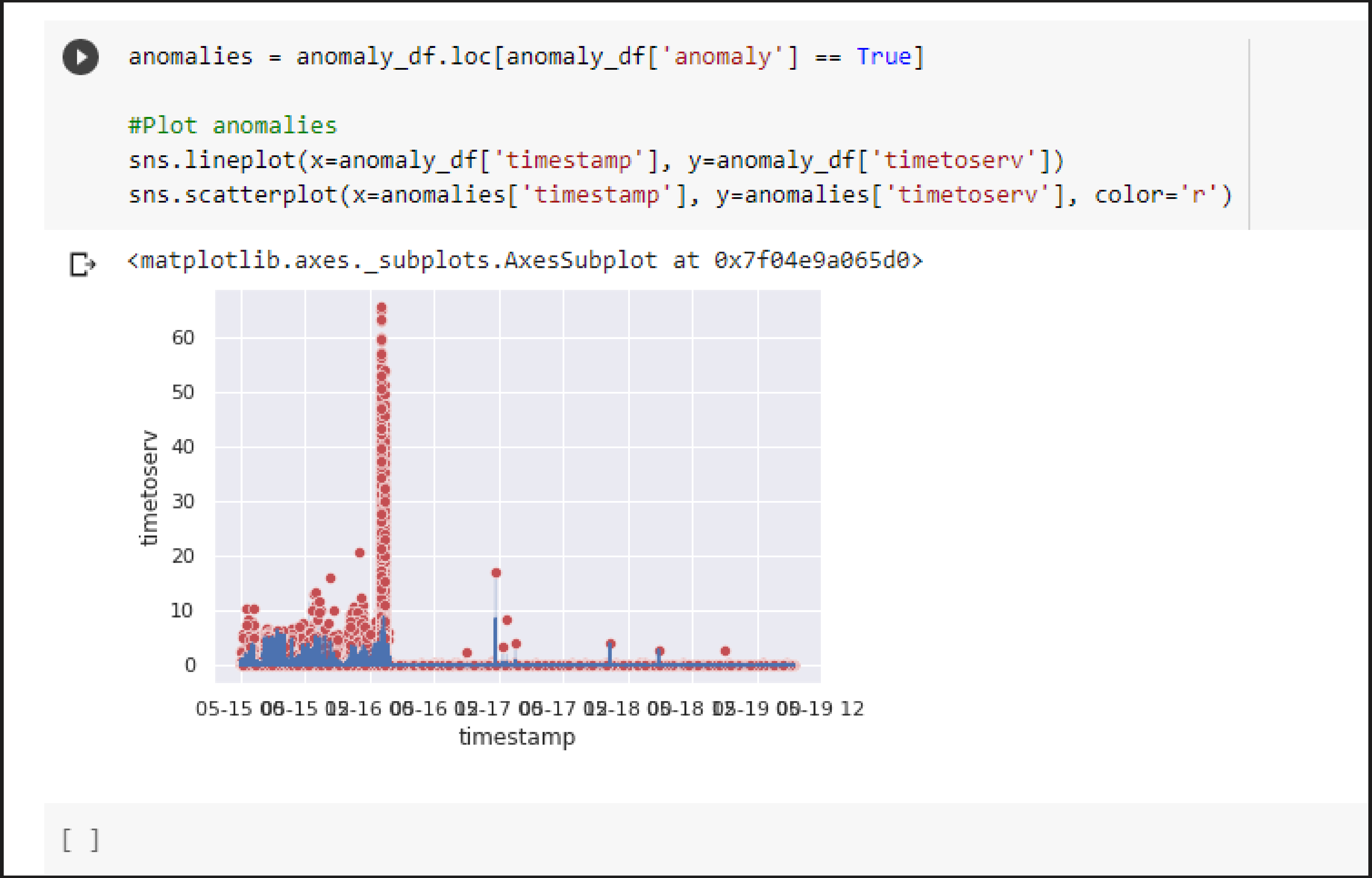
```
[ ]  df2['timestamp'].min(), df2['timestamp'].max()

     (Timestamp('2088-05-13 07:00:00'), Timestamp('2088-05-19 07:01:00'))
```



```
[ ]  train, test = df.loc[df['timestamp'] <= '2088-05-15'], df.loc[df['timestamp'] > '2088-05-15']
```

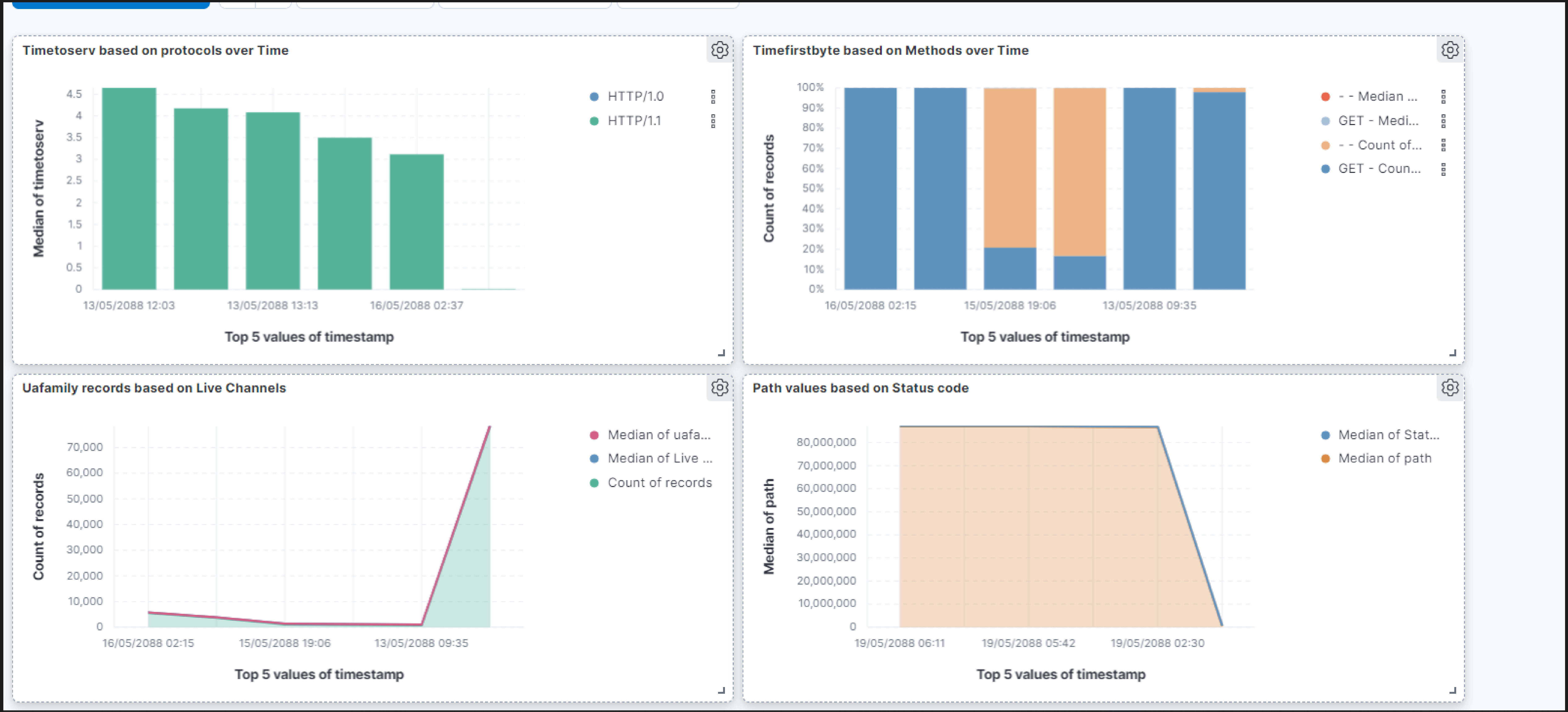


```
[ ]  TIME_STEPS=30

     def create_sequences(X, y, time_steps=TIME_STEPS):
         Xs, ys = [], []
         for i in range(len(X)-time_steps):
             Xs.append(X.iloc[i:(i+time_steps)].values)
             ys.append(y.iloc[i+time_steps])

         return np.array(Xs), np.array(ys)

     X_train, y_train = create_sequences(train[['timetoserv']], train['timetoserv'])
     X_test, y_test = create_sequences(test[['timetoserv']], test['timetoserv'])
```



```
anomalies = anomaly_df.loc[anomaly_df['anomaly'] == True]

#Plot anomalies
sns.lineplot(x=anomaly_df['timestamp'], y=anomaly_df['timetoserv'])
sns.scatterplot(x=anomalies['timestamp'], y=anomalies['timetoserv'], color='r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f04e9a065d0>
```



Timetoserv based on protocols over Time

Timefirstbyte based on Methods over Time

Uafamily records based on Live Channels

Path values based on Status code