

In [8]:

```
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"
```

In [15]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
import seaborn as sns
import csv

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from threading import Timer
from timeit import default_timer as timer
from IPython.display import clear_output
```

In [26]:

```
start = timer()
prep_dataset1 = pd.read_csv('../datasets/dataset_test_02_07.csv', delimiter="
df = prep_dataset1.iloc[:,1:4]
```

In [28]:

```
WINDOW = 35

for i in np.arange(df.shape[0]):
    init = i*WINDOW
    init2 = (i+1)*WINDOW
    if(init2<df.shape[0]):
        df.iloc[init:init+WINDOW,2] = df.iloc[init2,2]
```

In [29]:

```
df
```

Out[29]:

	temperature	label	delay
0	19.3024	1	126.251634
1	19.1652	1	126.251634
2	19175.0000	1	126.251634
3	19.1456	1	126.251634
4	19.1652	1	126.251634
...
4895	19.5768	0	420.416429
4896	19.5866	0	420.416429
4897	19567.0000	0	420.416429
4898	19.5572	0	420.416429
4899	19.5572	0	420.416429

4900 rows × 3 columns

In [30]:

```
train_size = int(len(df) * 0.8)
test_size = len(df) - train_size
train, test = df.iloc[0:train_size], df.iloc[train_size:len(df)]
```

In [31]:

test

Out[31]:

	temperature	label	delay
3920	17.5678	0	1121.053927
3921	17.5776	0	1121.053927
3922	17.5776	0	1121.053927
3923	17.5776	0	1121.053927
3924	17.5776	0	1121.053927
...
4895	19.5768	0	420.416429
4896	19.5866	0	420.416429
4897	19567.0000	0	420.416429
4898	19.5572	0	420.416429
4899	19.5572	0	420.416429

980 rows × 3 columns

In [32]:

```
def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        clear_output(wait=True)
        print('modeling to keras ', round((i/(len(X) - time_steps))*100,2), ('%
        s = round(timer() - start)
        if(s>60):
            s /=60
            print(' ', s, ' seconds')
        v = X.iloc[i: (i+time_steps), :].to_numpy()
        Xs.append(v)
        ys.append(y.iloc[i+time_steps])
    return np.array(Xs), np.array(ys)
```

In [33]:

```

def LSTMconf(X_train):
    print('Init config LSTM')
    model = keras.Sequential()
    model.add(
        keras.layers.Bidirectional(
            keras.layers.LSTM(
                activation="relu",
                units=512,
                input_shape=(X_train.shape[1],X_train.shape[2])
            )
        ))
    model.add(keras.layers.Dense(units=512, activation="relu"))
    model.add(keras.layers.Dense(units=512, activation="relu"))
    model.add(keras.layers.Dense(units=512, activation="relu"))
    model.add(keras.layers.Dense(units=512, activation="relu"))
    model.add(keras.layers.Dropout(rate=0.2))
    model.add(keras.layers.Dense(units=1))

    loss = "mse"
    optim = tf.keras.optimizers.Adam(
        learning_rate=0.0001)
    metrics=["accuracy"]

    model.compile(loss=loss, optimizer=optim,
#                 metrics=metrics
    )
    return model

```

In [34]:

```

def LSTMfit(model,X_train,Y_train):
    print('Init Train')
    start = timer()
    history = model.fit(
        X_train, Y_train,
        epochs=256,
        batch_size= 128,
        validation_split=0.1,
        shuffle=False,
#         callbacks=[tensorboard_callback]
    )
    return history

```

In [35]:

```
X_train,Y_train = create_dataset(train, train.delay)
model = LSTMconf(X_train)
history = LSTMfit(model,X_train, Y_train)
```

```
Epoch 226/256
28/28 [=====] - 1s 39ms/step - loss:
87727.0781 - val_loss: 88.1199
Epoch 227/256
28/28 [=====] - 1s 39ms/step - loss:
73071.6406 - val_loss: 36.7401
Epoch 228/256
```

In [36]:

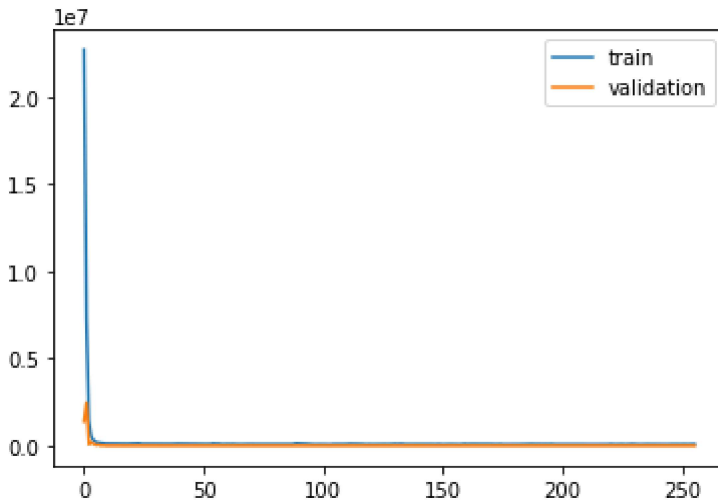
```
print('Saving Model')
model.save('models/lstm_mininet')
```

```
Saving Model
INFO:tensorflow:Assets written to: models/lstm_mininet/assets
```

loss training

In [37]:

```
fig1 = plt.figure()
ax1 = fig1.add_subplot(1,1,1)
ax1.plot(history.history['loss'], label='train')
ax1.plot(history.history['val_loss'], label='validation')
ax1.legend();
```



In [38]:

```
X_test,Y_test = create_dataset(test, test.delay)
```

modeling to keras 99.9 % 5.316666666666666 seconds

In [39]:

```
Y_test
140.13403916, 140.13403916, 140.13403916, 140.1340
3916,
140.13913512, 140.13913512, 140.13913512, 140.1391
3512,
140.13913512, 140.13913512, 140.13913512, 140.1391
3512,
140.13913512, 140.13913512, 140.13913512, 140.1391
3512,
140.13913512, 140.13913512, 140.13913512, 140.1391
3512,
140.13913512, 140.13913512, 140.13913512, 140.1391
3512,
140.13913512, 140.13913512, 140.13913512, 140.1391
3512,
140.13913512, 140.13913512, 140.13913512, 140.1391
3512,
140.13913512, 140.13913512, 140.13913512, 140.1357
0857,
```

predicting

In [40]:

```
y_pred = model.predict(X_test)
```

unnormalizing

In [41]:

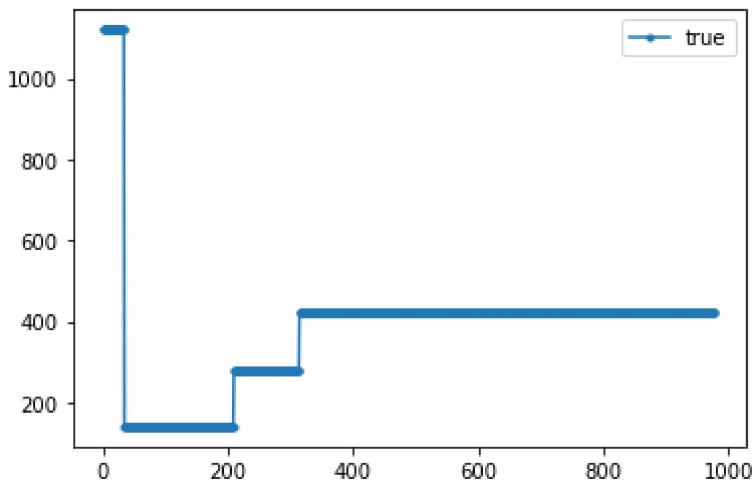
y_pred

```
[ 425.84042 ],
[ 160.54103 ],
[ 425.76636 ],
[ 425.796   ],
[ 425.796   ],
[ 160.54103 ],
[ 425.76636 ],
[ 425.76636 ],
[ 160.52351 ],
[ 425.6775   ],
[ 425.66266 ],
[ 425.66266 ],
[ 425.66266 ],
[ 425.64786 ],
[ 425.6182   ],

[ 425.60342 ],
[ 425.58856 ],
[ 160.53746 ],
[ 425.52927 ],
[ 425.51427 ]
```

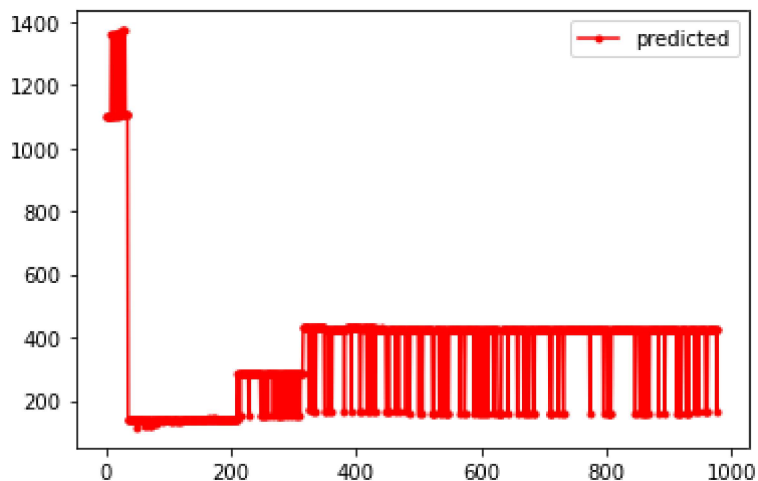
In [42]:

```
fig2 = plt.figure()
a2 = fig2.add_subplot(1,1,1)
a2.plot(Y_test, marker='.', label='true')
a2.legend();
```



In [43]:

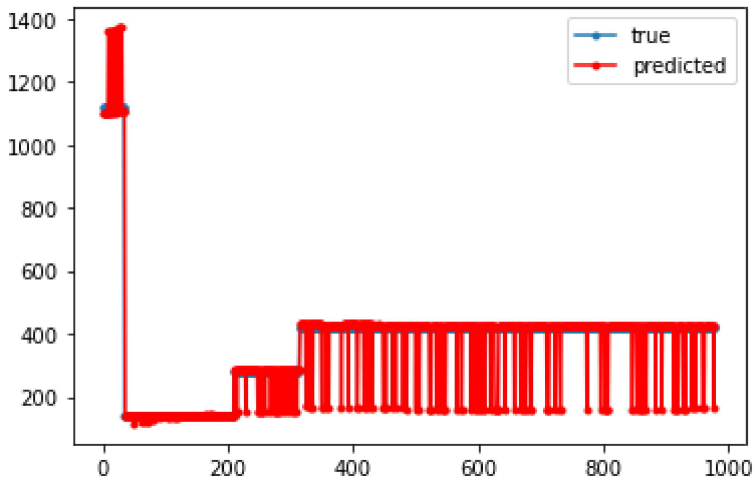
```
fig3 = plt.figure()
a3 = fig3.add_subplot(1,1,1)
a3.plot(y_pred,'r',marker='.', label='predicted')
a3.legend();
```



In [44]:

```
fig4 = plt.figure()
a4 = fig4.add_subplot(1,1,1)

a4.plot(Y_test, marker='.', label='true')
a4.plot(y_pred,'r',marker='.', label='predicted')
a4.legend();
```



In [45]:

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import median_absolute_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import explained_variance_score
```

In [46]:

```
size = np.min([y_pred.shape[0],Y_test.shape[0] ])
rmse = mean_squared_error(Y_test[0:size], y_pred[0:size], squared=False)
mae = mean_absolute_error(Y_test[0:size], y_pred[0:size])
median_mae = median_absolute_error(Y_test[0:size], y_pred[0:size])
evs = explained_variance_score(Y_test[0:size], y_pred[0:size])

print(rmse)
print(mae)
print(median_mae)
print('Explained Variance Score: ',evs)
```

93.19871231802112

37.191758939754735

5.097944736480713

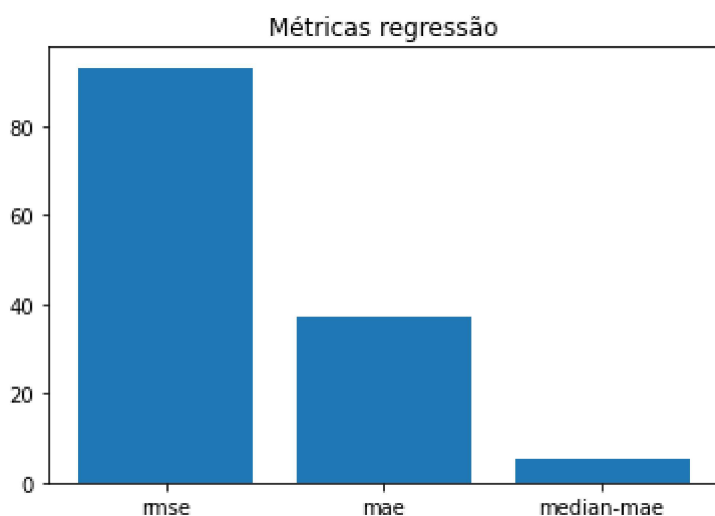
Explained Variance Score: 0.7401636930077289

In [47]:

```
objects = ('rmse', 'mae', 'median-mae')
y_pos = np.arange(3)
performance = [rmse,mae,median_mae]

plt.bar(y_pos, performance, align='center')
plt.xticks(y_pos, objects)
#plt.ylabel('Usage')
plt.title('Métricas regressão')

plt.show()
```



In [48]:

```
Y_test[0:size]
```

```
420.41642904, 420.41642904, 420.41642904, 420.4164
2904,
420.41642904, 420.41642904, 420.41642904, 420.4164
2904,
420.41642904, 420.41642904, 420.41642904, 420.4164
2904,
420.41642904, 420.41642904, 420.41642904, 420.4164
2904,
420.41642904, 420.41642904, 420.41642904, 420.4164
2904,
420.41642904, 420.41642904, 420.41642904, 420.4164
2904,
420.41642904, 420.41642904, 420.41642904, 420.4164
2904,
420.41642904, 420.41642904, 420.41642904, 420.4164
2904,
420.41642904, 420.41642904, 420.41642904, 420.4164
2904.
```

In []:

In []:

In []: