In [1]:

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

In [2]:

```
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 [3]:

```
start = timer()
prep_dataset1 = pd.read_csv('../datasets/dataset_test_02_07.csv', delimiter="
# prep_dataset2 = pd.read_csv('datasets/com_concept_drift/sdn_train_unormalize
# prep_dataset3 = pd.read_csv('datasets/com_concept_drift/sdn_train_unormalize
# prep_test = pd.read_csv('datasets/com_concept_drift/sdn_test_unormalized.csv

# prep_dataset1 = prep_dataset1[prep_dataset1.delay>0]
# prep_dataset2 = prep_dataset2[prep_dataset2.delay>0]
# prep_dataset3 = prep_dataset3[prep_dataset3.delay>0]
# prep_test = prep_test[prep_test.delay>0]
df = prep_dataset1.iloc[:,1:4]
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 [4]:

```
df
```

Out[4]:

	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 [5]:

```
train
```

Out[5]:

	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
...	...	...	...
3915	17.5678	0	125.066162
3916	17.5776	0	125.066162
3917	17.5776	0	125.066162
3918	17.5776	0	125.066162
3919	17.5776	0	125.066162

3920 rows × 3 columns

In [6]:

```
test
```

Out[6]:

	temperature	label	delay
3920	17.5678	0	125.066162
3921	17.5776	0	125.066162
3922	17.5776	0	125.066162
3923	17.5776	0	125.066162
3924	17.5776	0	125.066162
...	...	...	...
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

Normalizing

In [7]:

```
def normalizing(df):
    f_columns = ['temperature']
    scaler1 = StandardScaler().fit(df)
    scaler2 = StandardScaler().fit(df)

    scaler1= scaler1.fit(df[f_columns].to_numpy())
    scaler2 = scaler2.fit(df[['delay']])

    df.loc[:,f_columns] = scaler1.transform(df[f_columns].to_numpy())
    df['delay'] = scaler2.transform(df[['delay']])
    return df


def unormalizing(df,Y_test,y_pred ):

    scaler = StandardScaler().fit(df)
    scaler = scaler.fit(df[['delay']])
    y_test_inv = scaler.inverse_transform(Y_test.reshape(1,-1))
    y_pred_inv = scaler.inverse_transform(y_pred)

    return y_test_inv, y_pred_inv
```

In [8]:

```
def saveFile(dataset, name='dataset'):
    print('saving: ',name, '......')
    f = open(name,'w')
    try:
        writer = csv.writer(f)
        writer.writerow(dataset.columns)
        for i in np.arange(int(dataset.shape[0])):
            writer.writerow(dataset.iloc[i,])
    finally:
        f.close()


def preprocessing(dataset, order):
    print(dataset)
    saveFile(dataset, 'datasets/mininet/sdn_train_mininet_unormalized_'+str(o
    norm = normalizing(dataset)
    saveFile(norm, 'datasets/mininet/sdn_train_mininet_normalized_'+str(order

    return norm
```

In [9]:

```
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), 1:3].to_numpy()
        Xs.append(v)
        ys.append(y.iloc[i+time_steps])
    return np.array(Xs), np.array(ys)
```

In [10]:

```
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 [11]:

```
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 [12]:

```

# train = pd.read_csv('datasets/mininet/sdn_train_mininet_normalized_train.cs
# test = pd.read_csv('datasets/mininet/sdn_train_mininet_normalized_test.csv'
X_train,Y_train = create_dataset(train, train.delay)
model = LSTMconf(X_train)
history = LSTMfit(model,X_train, Y_train)

# r = Timer(1.0, preprocessing, (prep_dataset.iloc[cont:cont+window,:]))
# r.start()
# print(X_train)
```

```
modeling to keras  99.97 %Init config LSTM
Init Train
Epoch 1/256
28/28 [==============================] - 8s 79ms/step - los
s: 22323790.0000 - val_loss: 8404.9883
Epoch 2/256
28/28 [==============================] - 1s 51ms/step - los
s: 3529790.5000 - val_loss: 21.4900
Epoch 3/256
28/28 [==============================] - 2s 56ms/step - los
s: 2105504.0000 - val_loss: 1773.1234
Epoch 4/256
28/28 [==============================] - 2s 58ms/step - los
s: 2082267.0000 - val_loss: 24.1836
Epoch 5/256
28/28 [==============================] - 2s 55ms/step - los
s: 1326932.1250 - val_loss: 932.9672
Epoch 6/256
28/28 [==============================] - 1s 51ms/step - los
s: 1356240.2500 - val_loss: 285.0307
```

In [13]:

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

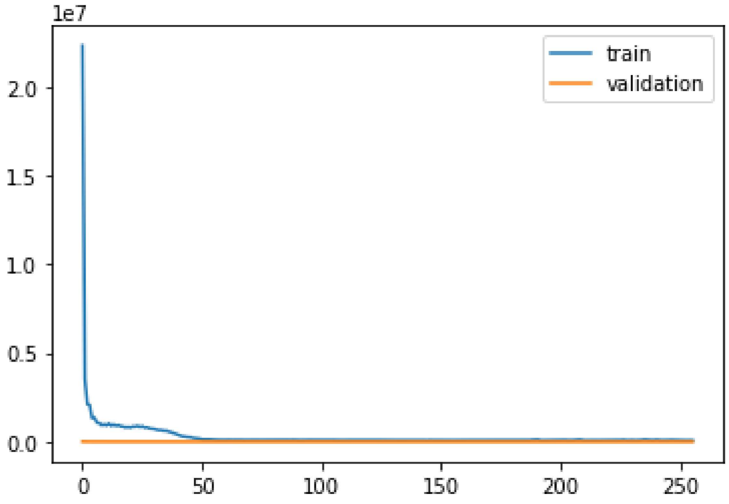
```
Saving Model
INFO:tensorflow:Assets written to: models/lstm_mininet\assets
```

loss training

In [14]:

```

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 [15]:

```
# test_un = pd.read_csv('datasets/mininet/sdn_train_mininet_unormalized_test.
test = pd.read_csv('datasets/mininet/sdn_train_mininet_unormalized_test.csv',
X_test,Y_test = create_dataset(test, test.delay)
```

```
modeling to keras  99.93 %  7.4  seconds
```

predicting

In [16]:

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

unormalizing

In [17]:

```
y_test_inv, y_pred_inv = Y_test, y_pred
```

In [18]:

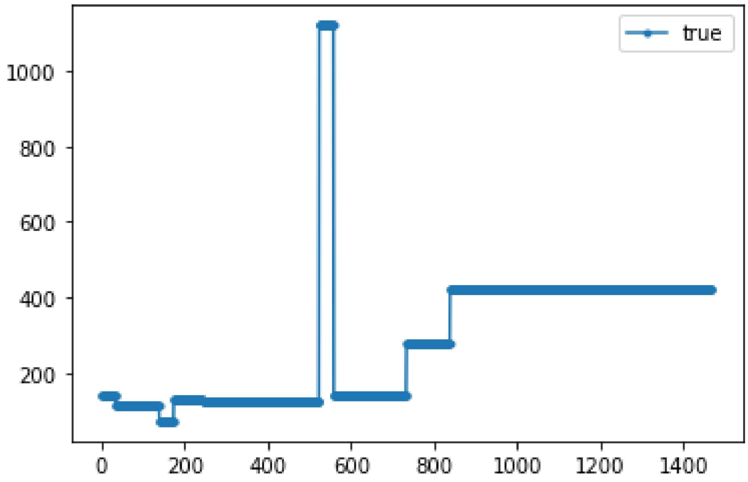
```
y_test_inv
```

Out[18]:

```
array([140.1252563 , 140.1252563 , 140.1252563 , ..., 420.41642
904,
       420.41642904, 420.41642904])
```

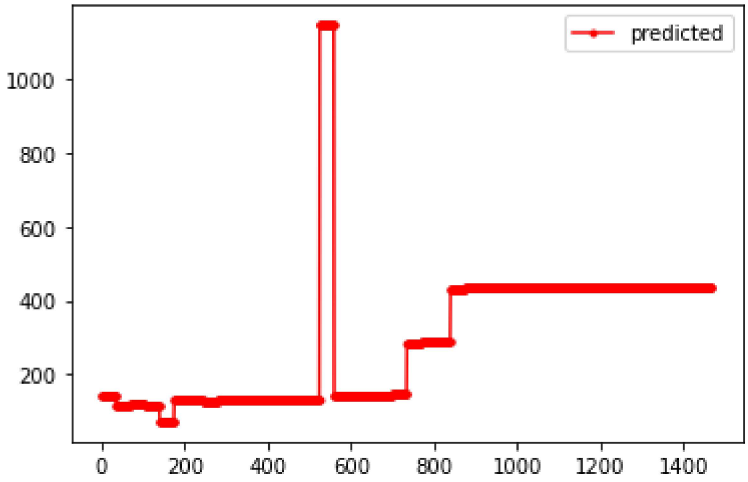
In [19]:

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



In [20]:

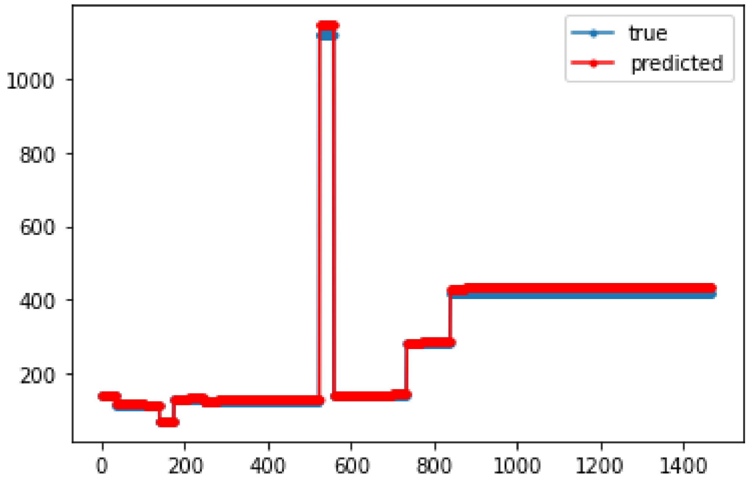
```
fig3 = plt.figure()
a3 = fig3.add_subplot(1,1,1)
a3.plot(y_pred_inv.flatten(),'r',marker='.', label='predicted')
a3.legend();
```



In [21]:

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

a4.plot(y_test_inv.flatten(), marker='.', label='true')
a4.plot(y_pred_inv.flatten(),'r',marker='.', label='predicted')
a4.legend();
```



In [22]:

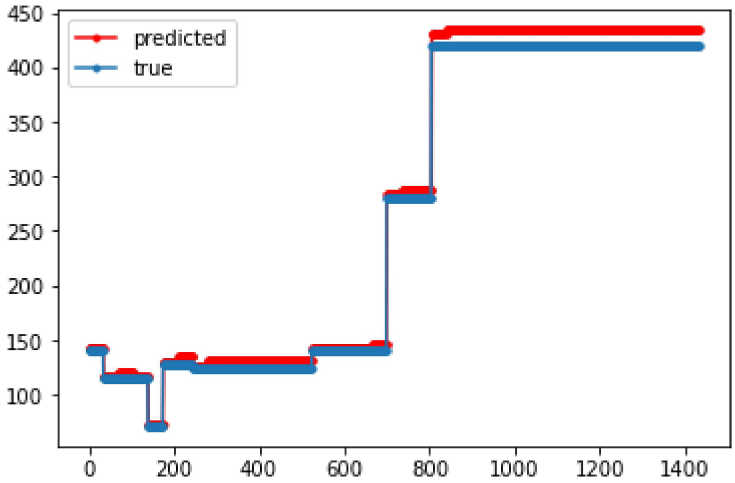
```
l1 = []
l2 = []
for i in np.arange(y_pred_inv.shape[0]):
    clear_output(wait=True)
    print('progress ',round((i/y_pred_inv.shape[0])*100,2), ('%'))
    if(y_pred_inv[i,0]<=1000):
        l1.append(y_pred_inv[i,0])
    if(y_test_inv[i]<=1000):
        l2.append(y_test_inv[i])

y_pred_inv2 = np.array(l1)
y_test_inv2 = np.array(l2)
```

```
progress  99.93 %
```

In [23]:

```
plt.plot(y_pred_inv2.flatten(),'r',marker='.', label='predicted')
plt.plot(y_test_inv2.flatten(), marker='.', label='true')
plt.legend();
```



In [24]:

```
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
```

In [25]:

```
size = np.min([y_pred_inv2.shape[0],y_test_inv2.shape[0] ])
rmse =  mean_squared_error(y_test_inv2[0:size], y_pred_inv2[0:size], squared=
mae =  mean_absolute_error(y_test_inv2[0:size], y_pred_inv2[0:size])
median_mae = median_absolute_error(y_test_inv2[0:size], y_pred_inv2[0:size])


print(rmse)
print(mae)
print(median_mae)
```

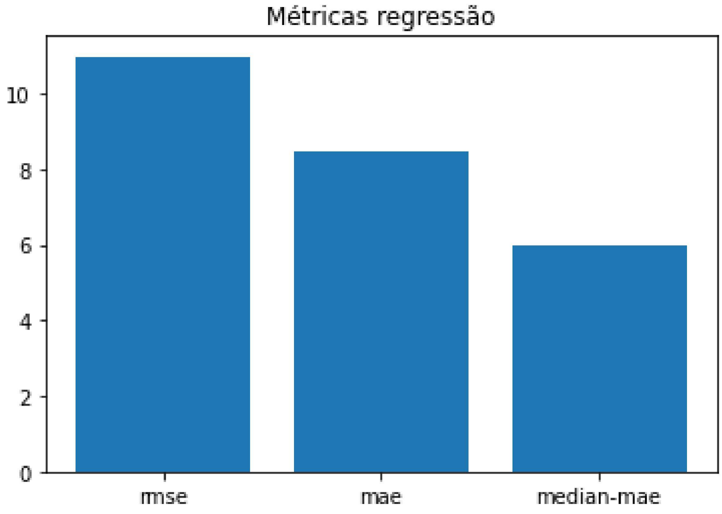
```
10.981107967162046
8.492871698974072
5.981628179550171
```

In [26]:

```
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 [27]:

```
from sklearn.metrics import explained_variance_score
```

In [28]:

```
explained_variance_score(y_test_inv2[0:size], y_pred_inv2[0:size])
```

Out[28]:

```
0.9972220565104594
```

In [29]:

```
y_test_inv2[0:size]
```

Out[29]:

```
array([140.1252563 , 140.1252563 , 140.1252563 , ..., 420.41642
904,
       420.41642904, 420.41642904])
```

In [ ]:

In [ ]:

In [ ]: