In [1]:

```python
import os
import tensorflow as tf
from tensorflow import keras

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from timeit import default_timer as timer
from IPython.display import clear_output
```

In [2]:

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

In [3]:

```python
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), 2:4].to_numpy()
        Xs.append(v)
        ys.append(y.iloc[i+time_steps])
    return np.array(Xs), np.array(ys)
```

In [4]:

```python
#carregando datasets
print('loading dataset...')
train = pd.read_csv('../datasets/com_concept_drift/sdn_train_unormalized.csv'
test = pd.read_csv('../datasets/com_concept_drift/sdn_test_unormalized.csv')

train = train[train.delay>=0]
test = test[test.delay>=0]

train = train[train.delay<=10000]
test = test[test.delay<=10000]
```

loading dataset...

In [ ]:

In [5]:

```python
start = timer()
print('creating window')
TIME_STEPS = 1
X_train,Y_train = create_dataset(train, train.delay, time_steps=TIME_STEPS)
X_test,Y_test = create_dataset(test, test.delay, time_steps=TIME_STEPS)

print('2D to 3D duration: ', round(timer() - start))
```

modeling to keras 100.0 % 11.0 seconds
2D to 3D duration: 660

setting MLP

In [6]:

```python
#configurando rede para treinamento
print('Init Train')
model = keras.Sequential()
model.add(tf.keras.layers.Dense(activation="relu", input_dim=2, units=10, ker
model.add(tf.keras.layers.Dense(activation="relu", units=128, kernel_initiali
model.add(tf.keras.layers.Dense(activation="relu", units=128, kernel_initiali
model.add(tf.keras.layers.Dense(activation="relu", units=128, kernel_initiali
model.add(keras.layers.Dropout(rate=0.2))
model.add(keras.layers.Dense(units=1))
```

Init Train

compiling

In [7]:

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

model.compile(loss=loss, optimizer=optim,
            metrics=metrics
            )
```

training

In [8]:

```python
print('Init Train')
start = timer()
history = model.fit(
    X_train, Y_train,
    epochs=100,
    batch_size= 10,
    validation_split=0.1,
    shuffle=False,
#     callbacks=[tensorboard_callback]
)
print('trraining duration: ',round(timer() - start))
```

```
Epoch 30/100
4995/4995 [==============================] - 15s 3ms/step -
loss: 2127.2251 - accuracy: 0.5622 - val_loss: 1622.8667 - v
al_accuracy: 0.0000e+00
Epoch 31/100
4995/4995 [==============================] - 15s 3ms/step -
loss: 2042.5035 - accuracy: 0.5716 - val_loss: 1546.9814 - v
al_accuracy: 0.0000e+00
Epoch 32/100
4995/4995 [==============================] - 15s 3ms/step -
loss: 2019.0492 - accuracy: 0.5839 - val_loss: 1517.6700 - v
al_accuracy: 0.0000e+00

Epoch 33/100
4995/4995 [==============================] - 15s 3ms/step -
loss: 2042.0206 - accuracy: 0.5749 - val_loss: 1517.4965 - v
al_accuracy: 0.0000e+00
Epoch 34/100
4995/4995 [==============================] - 14s 3ms/step -
loss: 2065.6055 - accuracy: 0.5702 - val_loss: 1626.9097 - v
al accuracy: 0.0000e+00
```

saving model

In [9]:

```python
print('Saving Model')
model.save('models/mlp')
```
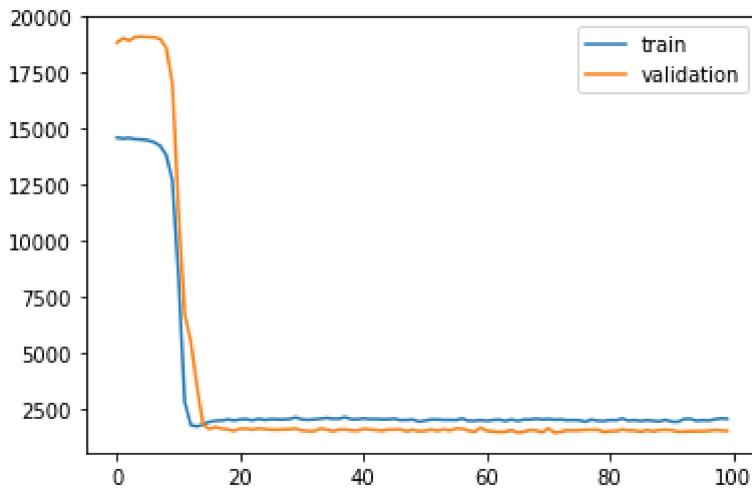
```
Saving Model
INFO:tensorflow:Assets written to: models/mlp\assets
```
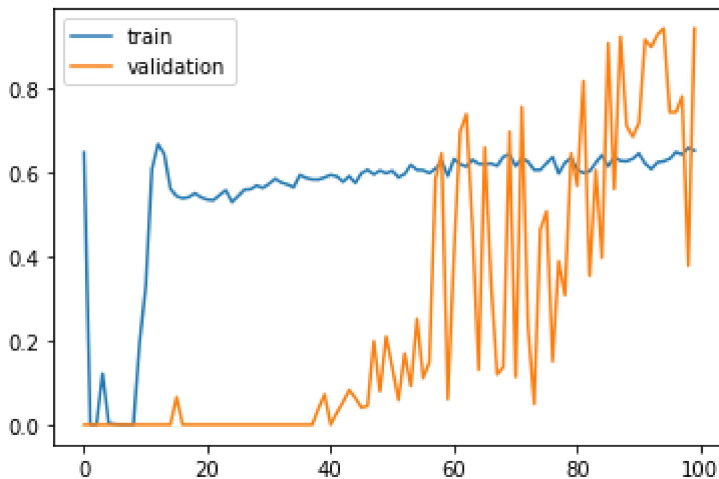
loss training

In [10]:

```python
# accuracy


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();
```


accuracy

In [11]:

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


predicting

In [12]:

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

```
WARNING:tensorflow:Model was constructed with shape (None, 2) f
or input KerasTensor(type_spec=TensorSpec(shape=(None, 2), dtyp
e=tf.float32, name='dense_input'), name='dense_input', descript
ion="created by layer 'dense_input'"), but it was called on an
input with incompatible shape (None, 1, 2).
```

unormalizing

In [13]:

```python
f_columns = ['temperature','label']
scaler1 = StandardScaler().fit(train[f_columns])
scaler2 = StandardScaler().fit(train[f_columns])

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


#normalizando test
scaler3 = StandardScaler().fit(test[f_columns])
scaler4 = StandardScaler().fit(test[f_columns])

scaler3 = scaler3.fit(test[f_columns].to_numpy())
scaler4 = scaler4.fit(test[['delay']])
```
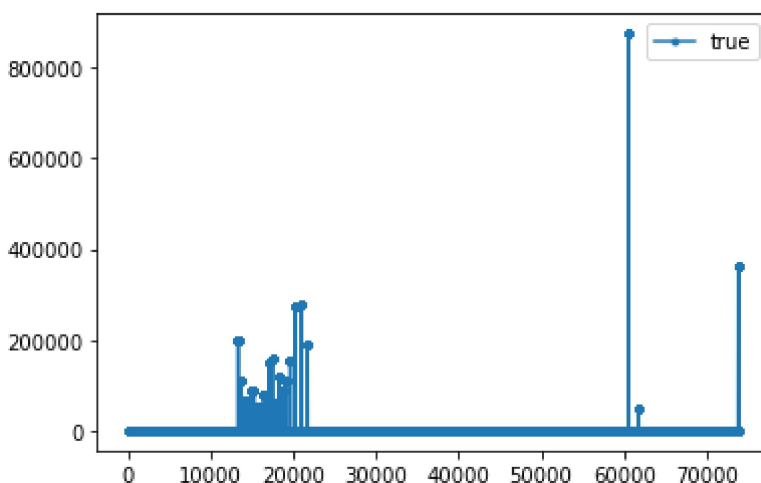
In [14]:

```python
y_test_inv = scaler4.inverse_transform(Y_test.reshape(1,-1))
y_pred_inv = scaler4.inverse_transform(y_pred)
```
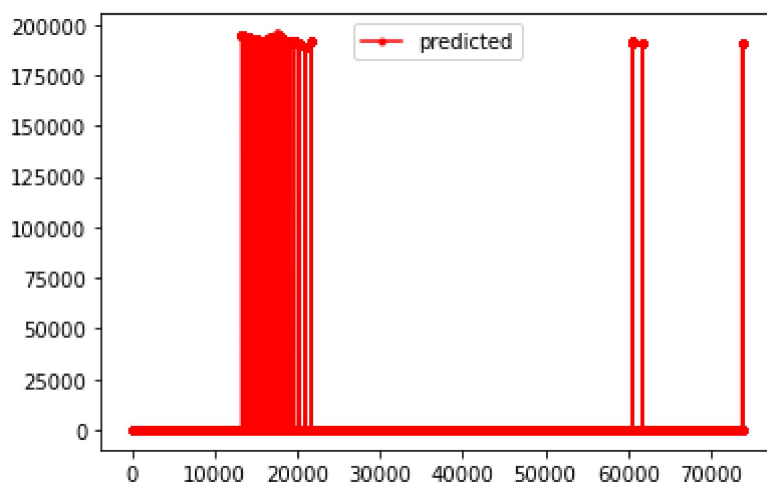
In [15]:

```python
fig2 = plt.figure()
a2 = fig2.add_subplot(1,1,1)
a2.plot(y_test_inv.flatten(), marker='.', label='true')
a2.legend();
```
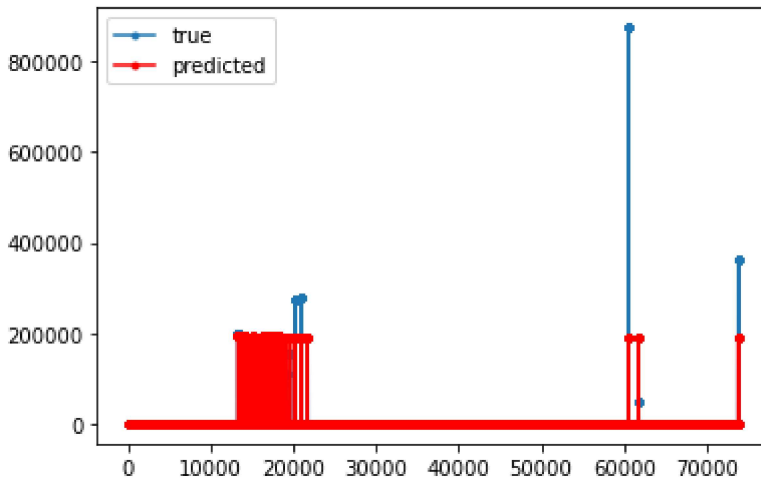
In [16]:

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

In [17]:

```python
plt.plot(y_test_inv.flatten(), marker='.', label='true')

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


In [18]:

```python
from sklearn.metrics import mean_squared_error
```

In [19]:

```python
np.max(y_test_inv)
```

Out[19]:

875000.6517809001

In [20]:

```python
np.max(y_pred_inv[:,0])
```

Out[20]:

195555.28

In [21]:

```python
mean_squared_error(y_test_inv[0], y_pred_inv[:,0])
```

Out[21]:

514186540.9281615

In [ ]:

In [ ]: