demo_notebook

June 3, 2021

Colab <-> Github Credit: https://medium.com/@ashwindesilva/how-to-use-google-colaboratory-to-clone-a-github-repository-e07cf8d3d22b

https://github.com/jakee417/ProbabilisticForecasting for our full repo

[1]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

[3]:
```
% cd gdrive/My Drive/Stat 271/
```

```
/content/gdrive/My Drive/Stat 271
```

[4]:
```
! git clone https://github.com/jakee417/ProbabilisticForecasting
```

```
Cloning into 'ProbabilisticForecasting'…
remote: Enumerating objects: 393, done.
remote: Counting objects: 100% (393/393), done.
remote: Compressing objects: 100% (345/345), done.
remote: Total 393 (delta 47), reused 390 (delta 47), pack-reused 0
Receiving objects: 100% (393/393), 61.22 MiB | 13.94 MiB/s, done.
Resolving deltas: 100% (47/47), done.
Checking out files: 100% (377/377), done.
```

[5]:
```
% cd ProbabilisticForecasting
```

```
/content/gdrive/My Drive/Stat 271/ProbabilisticForecasting
```

[6]:
```
from WindowGenerator import WindowGenerator
import matplotlib.pyplot as plt
from LstmRnn import LstmRnn
import json
import time
import pprint
from datetime import datetime
pp = pprint.PrettyPrinter(indent=4)
```

[7]:
```
# forecast example:
# input_width=96,
# label_width=24,
# shift=24,

# anomaly example:
# input_width=24,
# label_width=24,
# shift=0,
```

[9]:
```
# algorithm parameters
bitcoin = 'data/bitcoin_query.csv'
ethereum = 'data/ethereum_clean.csv'
# distributions: 'normal', 'student_t', 'mix', 'laplace', 'poisson_approx'
params = dict(
        fname=ethereum,
        distribution='normal',  # likelihood model
        lstm_units=32,  # hidden units for lstm and dense cells.      #was 32
        t2v_units=8,  # might leave this to 8 for now.
        dense_cells=None,  # extra dense cells after lstm cell. Can be None.
        resample=None,  # leave this none for more data
        input_width=24,  # past 3 days
        label_width=24,  # one day
        shift=0,  # determines the index of the first forecast point . 0 for␣
 ↪anomaly, label_width for forecast.
        max_epochs=40,
        patience=2,  # reduce_on_plateau=patience, early_stopping=patience*2
        latent_dim=None,  # can be None or 2 and above.     #was 2
        beta=1,  # 1 or greater
        min_df=2.0,  # for the student-t
        number_states=30,  # for the hmm
        batch_size=256,  # 256 => 30s/epoch
        regularization= None  # applies to lstm and dense cell kernels. Can be␣
 ↪None.
    )
```

[10]:
```
now = datetime.now().strftime("%d_%m_%Y_%H_%M_%S")
distribution = params['distribution']
checkpoint_path = f'checkpoints/{distribution}.ckpt'
save_path = f'saved/LstmRnn{distribution}'
train_save_img = f'figures/{now}_{distribution}_train_lstm_rnn.jpg'
val_save_img = f'figures/{now}_{distribution}_val_lstm_rnn.jpg'
test_save_img = f'figures/{now}_{distribution}_test_lstm_rnn.jpg'
global_train_img = f'figures/{now}_train_{distribution}_global_lstm_rnn.jpg'
global_val_img = f'figures/{now}_val_{distribution}_global_lstm_rnn.jpg'
global_test_img = f'figures/{now}_test_{distribution}_global_lstm_rnn.jpg'
loss_img = f'figures/{now}_{distribution}_loss.jpg'
```

```
post_check_img = f'figures/{now}_{distribution}_post_check.jpg'
train_correlation_img = f'figures/{now}_train_{distribution}_correlation'
test_correlation_img = f'figures/{now}_test_{distribution}_correlation'
```

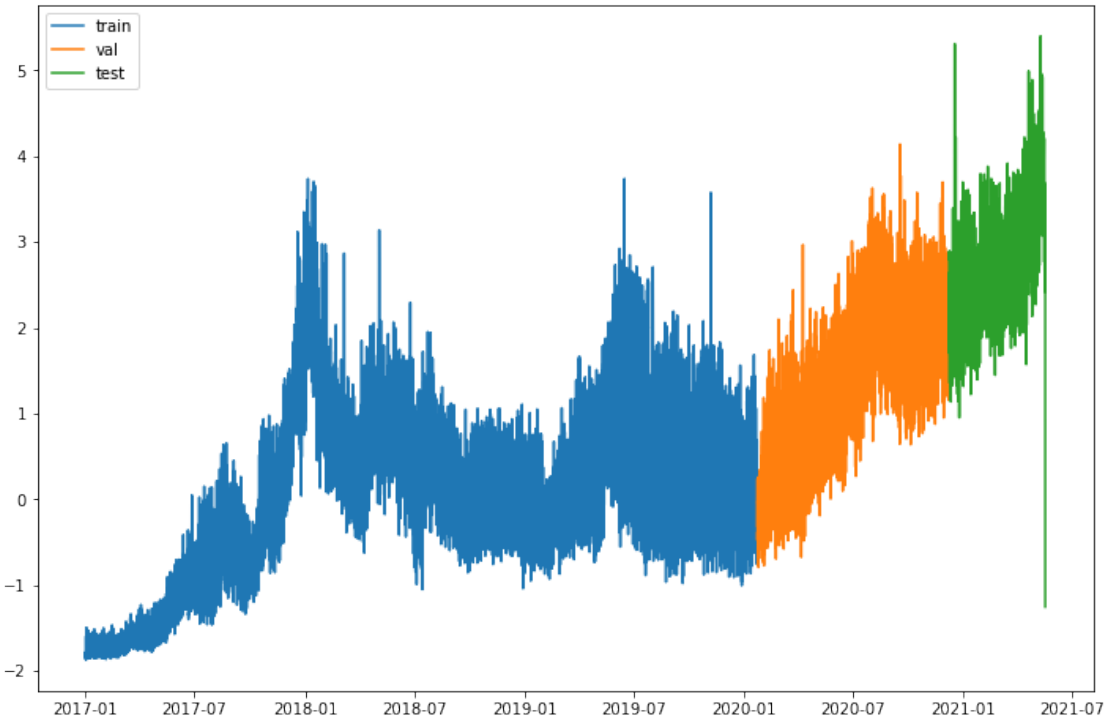
[11]:
```
multi_window = WindowGenerator(params['fname'],
                               input_width=params['input_width'],
                               label_width=params['label_width'],
                               shift=params['shift'],
                               label_columns=['num_transactions'],
                               resample_frequency=params['resample'],
                               standardize=True,
                               batch_size=params['batch_size'])

# multi_window.plot_splits(feedback_model)
print(multi_window)
time_index = multi_window.column_indices['timestamp']
```

```
Total window size: 24
Input indices: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
21 22 23]
Shift (Input) -> (Label): 0
Label indices: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
21 22 23]
Label column name(s): ['num_transactions']
Total time series: 38328
```

[12]:
```
print(len(multi_window.train))
print(len(multi_window.val))
print(len(multi_window.test))
```

```
105
30
15
```

[13]:
```
multi_window.plot_splits()
```



[14]:
```
params.update({'time_index': time_index})
feedback_model = LstmRnn(params)
feedback_model(multi_window.train_example[0])
print(feedback_model.summary())
```

```
Model: "lstm_rnn"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_warmup (LSTMCell)       multiple                  5632
_________________________________________________________________
lstm_forecast (LSTMCell)     multiple                  4480
_________________________________________________________________
rnn (RNN)                    multiple                  5632
_________________________________________________________________
t2v (T2V)                    multiple                  248
_________________________________________________________________
normal (DistributionLambda)  multiple                  0
_________________________________________________________________
dense_extra (Dense)          multiple                  0 (unused)
_________________________________________________________________
dense_main (Dense)           multiple                  66
=================================================================
Total params: 10,434
```

```
Trainable params: 10,426
Non-trainable params: 8

_________________________________________________________________
None
```

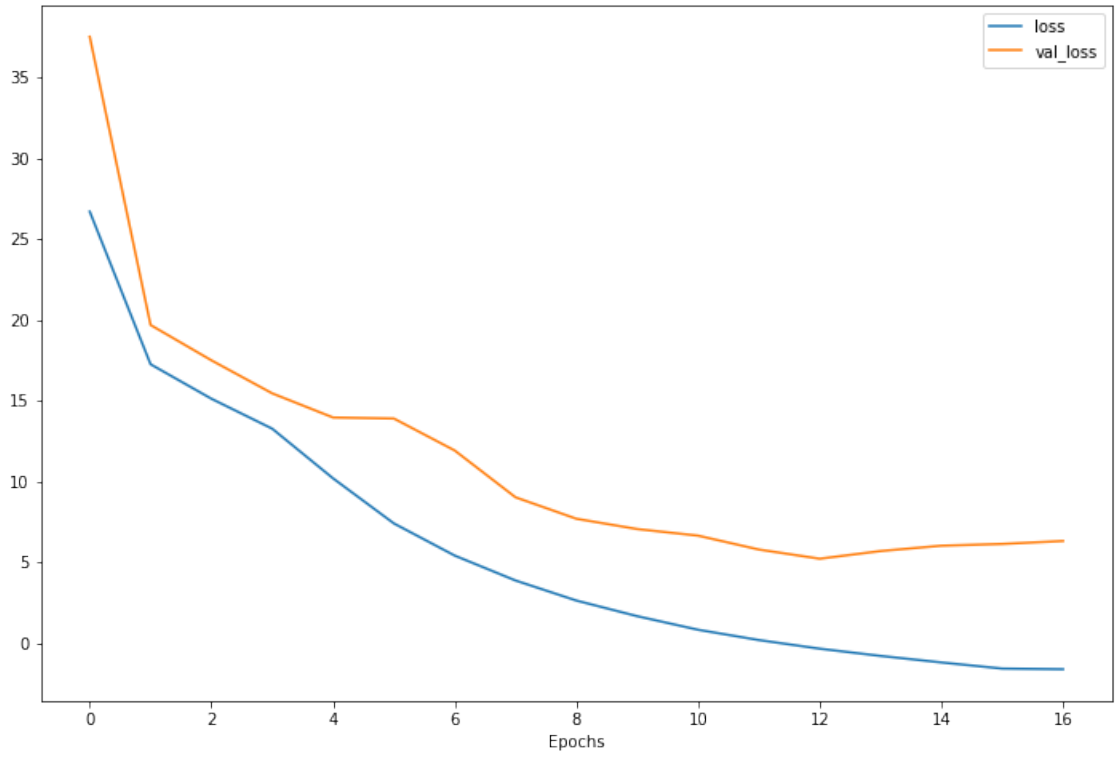
[15]:
```
tic = time.time()
history = feedback_model.compile_and_fit(
    model=feedback_model,
    window=multi_window,
    checkpoint_path=checkpoint_path,
    save_path=save_path,
    max_epochs=params['max_epochs'],
    patience=params['patience']
    )
toc = time.time()
```

```
Epoch 1/40
105/105 [==============================] - 14s 74ms/step - loss: 32.4046 -
reconstruction_loss: 26.6951 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 37.5125 - val_reconstruction_loss: 37.5125 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00001: saving model to checkpoints/normal.ckpt
Epoch 2/40
105/105 [==============================] - 7s 69ms/step - loss: 17.9934 -
reconstruction_loss: 17.2515 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 19.6801 - val_reconstruction_loss: 19.6801 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00002: saving model to checkpoints/normal.ckpt
Epoch 3/40
105/105 [==============================] - 7s 66ms/step - loss: 15.5020 -
reconstruction_loss: 15.1190 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 17.4951 - val_reconstruction_loss: 17.4951 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00003: saving model to checkpoints/normal.ckpt
Epoch 4/40
105/105 [==============================] - 7s 66ms/step - loss: 13.8948 -
reconstruction_loss: 13.2652 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 15.4511 - val_reconstruction_loss: 15.4511 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00004: saving model to checkpoints/normal.ckpt
Epoch 5/40
105/105 [==============================] - 7s 67ms/step - loss: 11.0062 -
reconstruction_loss: 10.1933 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 13.9578 - val_reconstruction_loss: 13.9578 -
```

```
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00005: saving model to checkpoints/normal.ckpt
Epoch 6/40
105/105 [==============================] - 7s 66ms/step - loss: 7.9875 -
reconstruction_loss: 7.4063 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 13.8946 - val_reconstruction_loss: 13.8946 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00006: saving model to checkpoints/normal.ckpt
Epoch 7/40
105/105 [==============================] - 7s 64ms/step - loss: 5.8498 -
reconstruction_loss: 5.4163 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 11.9167 - val_reconstruction_loss: 11.9167 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00007: saving model to checkpoints/normal.ckpt
Epoch 8/40
105/105 [==============================] - 7s 65ms/step - loss: 4.2411 -
reconstruction_loss: 3.8753 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 9.0213 - val_reconstruction_loss: 9.0213 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00008: saving model to checkpoints/normal.ckpt
Epoch 9/40
105/105 [==============================] - 7s 63ms/step - loss: 2.8876 -
reconstruction_loss: 2.6351 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 7.6950 - val_reconstruction_loss: 7.6950 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00009: saving model to checkpoints/normal.ckpt
Epoch 10/40
105/105 [==============================] - 7s 63ms/step - loss: 1.8763 -
reconstruction_loss: 1.6688 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 7.0575 - val_reconstruction_loss: 7.0575 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00010: saving model to checkpoints/normal.ckpt
Epoch 11/40
105/105 [==============================] - 7s 63ms/step - loss: 1.0558 -
reconstruction_loss: 0.8293 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 6.6502 - val_reconstruction_loss: 6.6502 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00011: saving model to checkpoints/normal.ckpt
Epoch 12/40
105/105 [==============================] - 7s 61ms/step - loss: 0.3132 -
reconstruction_loss: 0.1916 - regularization_loss: 0.0000e+00 - kl_loss:
```

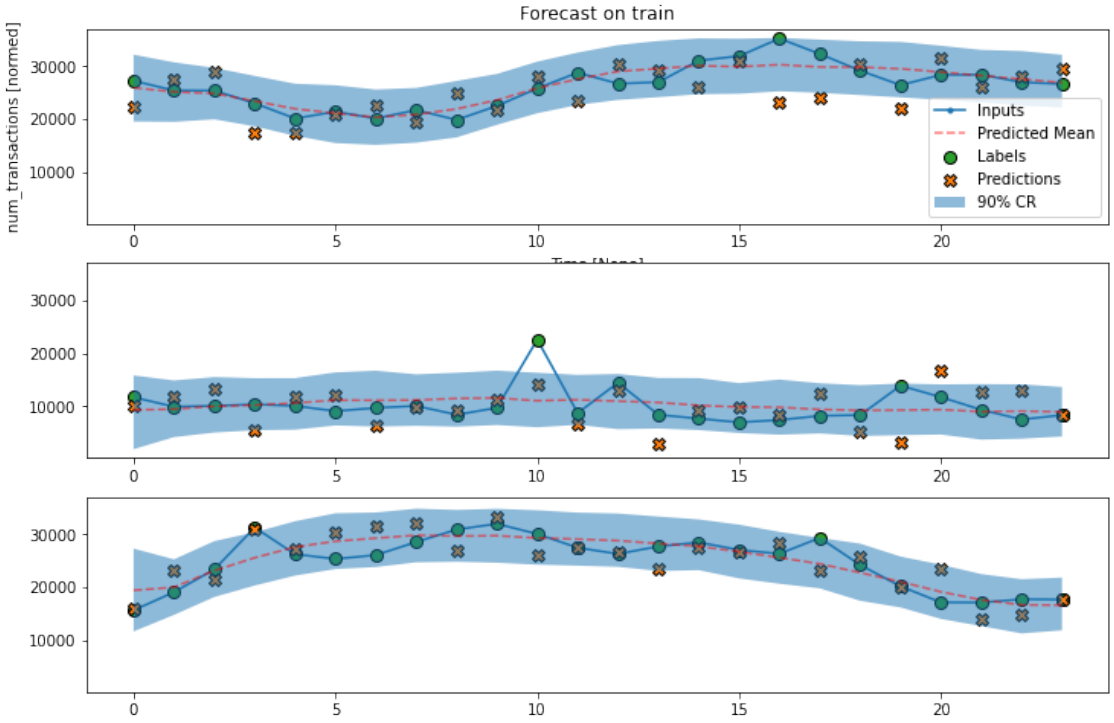
```
0.0000e+00 - val_loss: 5.7913 - val_reconstruction_loss: 5.7913 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00012: saving model to checkpoints/normal.ckpt
Epoch 13/40
105/105 [==============================] - 7s 66ms/step - loss: -0.1923 -
reconstruction_loss: -0.3423 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 5.2229 - val_reconstruction_loss: 5.2229 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00013: saving model to checkpoints/normal.ckpt
Epoch 14/40
105/105 [==============================] - 7s 62ms/step - loss: -0.5982 -
reconstruction_loss: -0.7891 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 5.6969 - val_reconstruction_loss: 5.6969 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00014: saving model to checkpoints/normal.ckpt
Epoch 15/40
105/105 [==============================] - 7s 64ms/step - loss: -1.0749 -
reconstruction_loss: -1.1928 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 6.0275 - val_reconstruction_loss: 6.0275 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00015: ReduceLROnPlateau reducing learning rate to 0.00010100000004749745.

Epoch 00015: saving model to checkpoints/normal.ckpt
Epoch 16/40
105/105 [==============================] - 7s 62ms/step - loss: -1.5423 -
reconstruction_loss: -1.5719 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 6.1370 - val_reconstruction_loss: 6.1370 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00

Epoch 00016: saving model to checkpoints/normal.ckpt
Epoch 17/40
105/105 [==============================] - 7s 63ms/step - loss: -1.5712 -
reconstruction_loss: -1.6138 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00 - val_loss: 6.3181 - val_reconstruction_loss: 6.3181 -
val_regularization_loss: 0.0000e+00 - val_kl_loss: 0.0000e+00
Restoring model weights from the end of the best epoch.

Epoch 00017: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.

Epoch 00017: saving model to checkpoints/normal.ckpt
Epoch 00017: early stopping
```

[16]:
```
# plot history of loss and val_loss
plt.figure(figsize=(12, 8))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.xlabel('Epochs')
plt.savefig(loss_img)
plt.show()
```

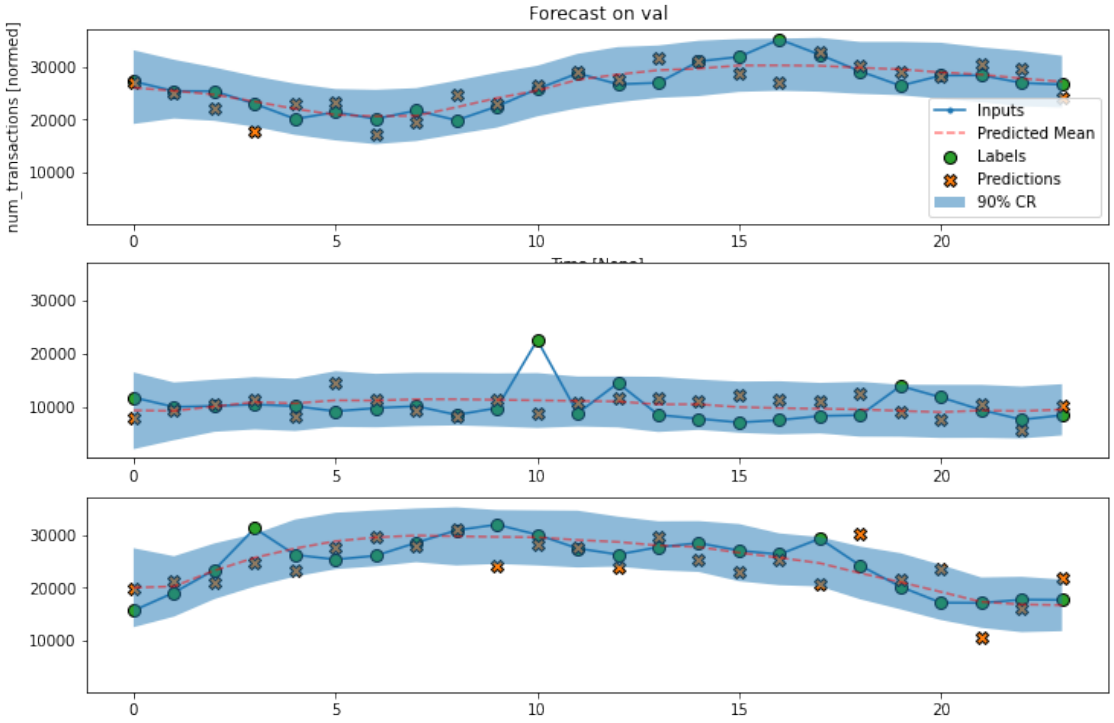


[17]:
```
samples = 500
```

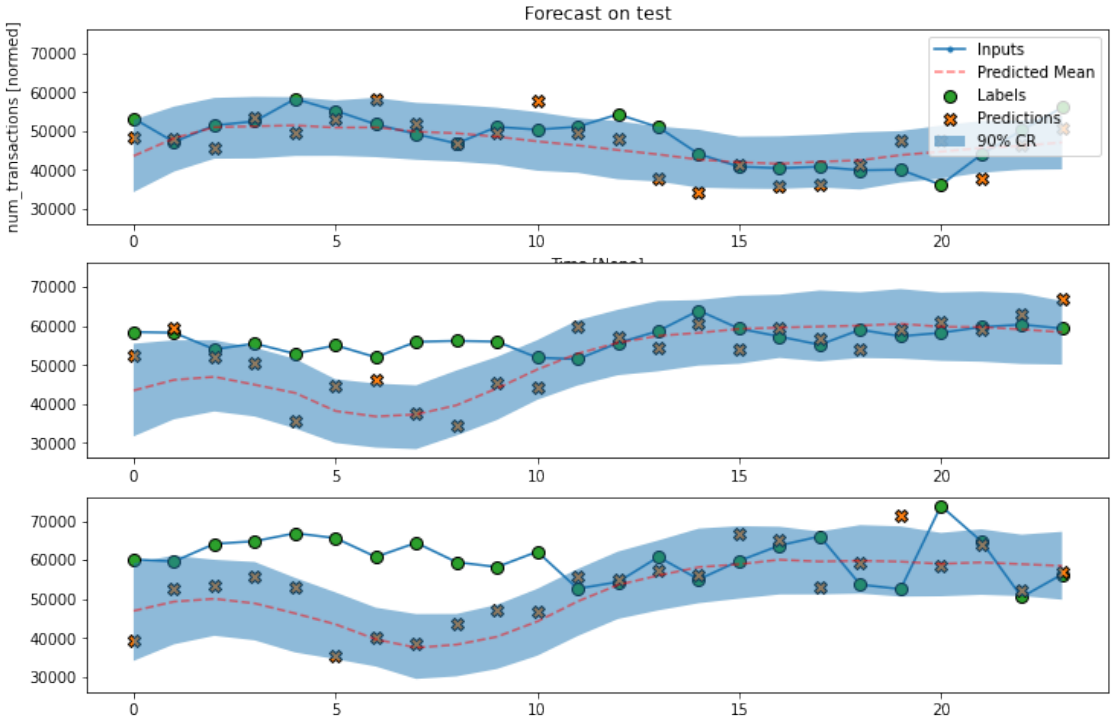
[18]:
```
multi_window.plot(feedback_model,
                  save_path=train_save_img,
                  max_subplots=3,
                  samples=samples,
                  mode='train')
```



[19]:
```
multi_window.plot(feedback_model,
                  save_path=val_save_img,
                  max_subplots=3,
                  samples=samples,
                  mode='val')
```



[20]:
```
multi_window.plot(feedback_model,
                  save_path=test_save_img,
                  max_subplots=3,
                  samples=samples,
                  mode='test')
```



[21]:
```
forecasts = []
```

[22]:
```
# This takes a while - also don't need for now
train_forecast = multi_window.forecast(
        model=feedback_model,
        dataset_name='train',
        samples=samples
    )
forecasts.append(train_forecast)
```

```
Sampling forecast values: train
Sampling complete: 20.49944806098938
Assign samples time index: 103.4306411743164
```

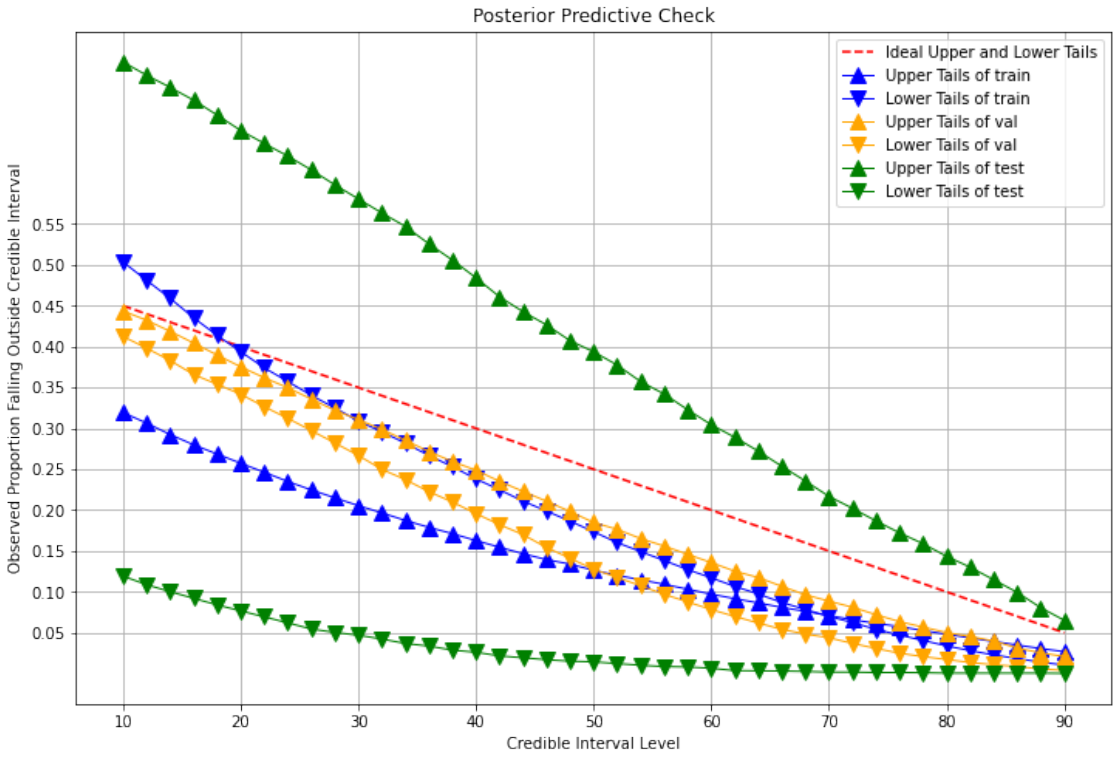
[23]:
```
val_forecast = multi_window.forecast(
        model=feedback_model,
        dataset_name='val',
        samples=samples
    )
forecasts.append(val_forecast)
```

```
Sampling forecast values: val
Sampling complete: 5.8086793422698975
Assign samples time index: 18.315253257751465
```

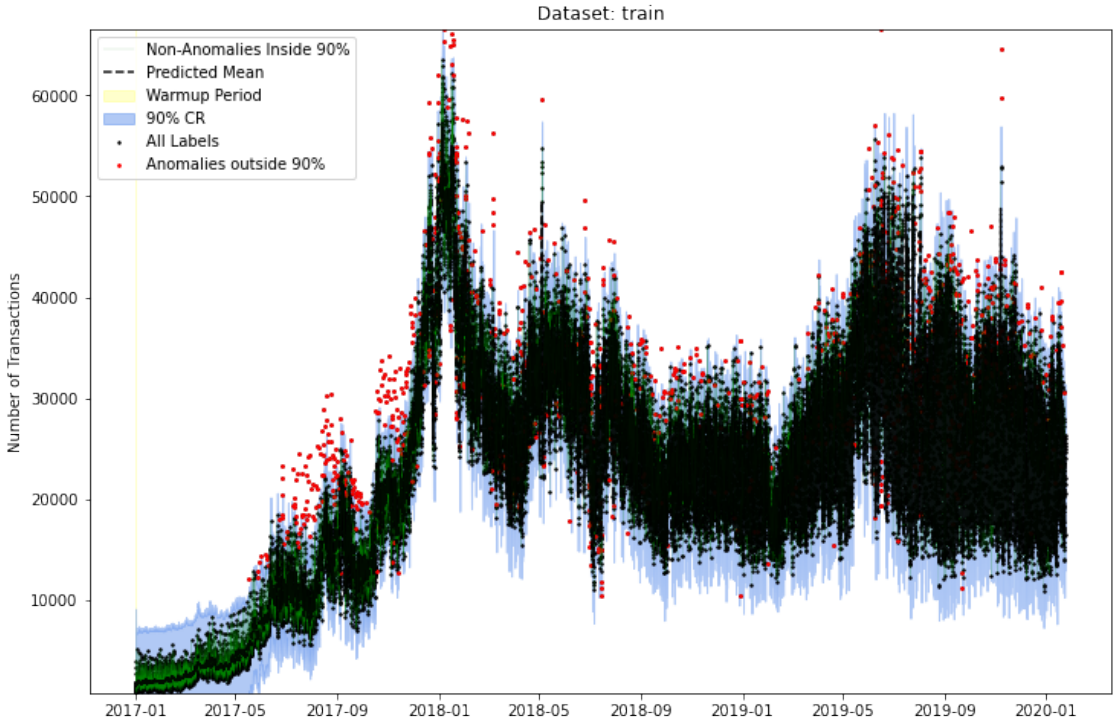
[24]:
```
test_forecast = multi_window.forecast(
    model=feedback_model,
    dataset_name='test',
    samples=samples
)
forecasts.append(test_forecast)
```

```
Sampling forecast values: test
Sampling complete: 5.136912822723389
Assign samples time index: 10.379784345626831
```

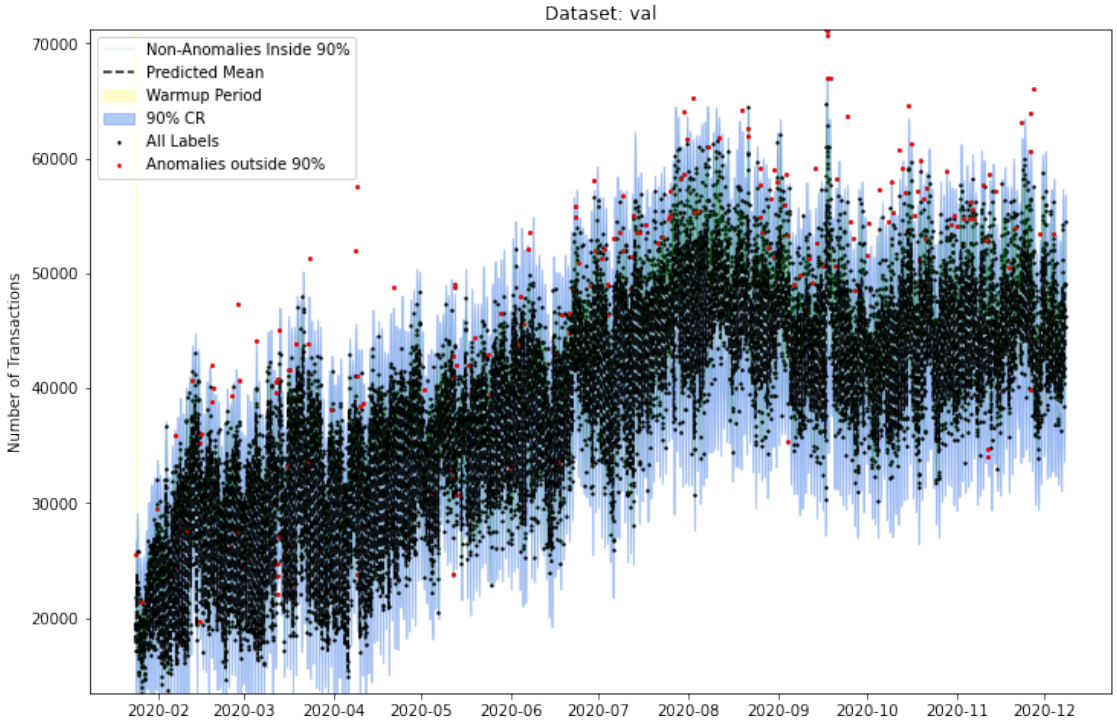
[25]:
```
post_checks = multi_window.plot_posterior_predictive_check(forecasts,␣
↪post_check_img)
```



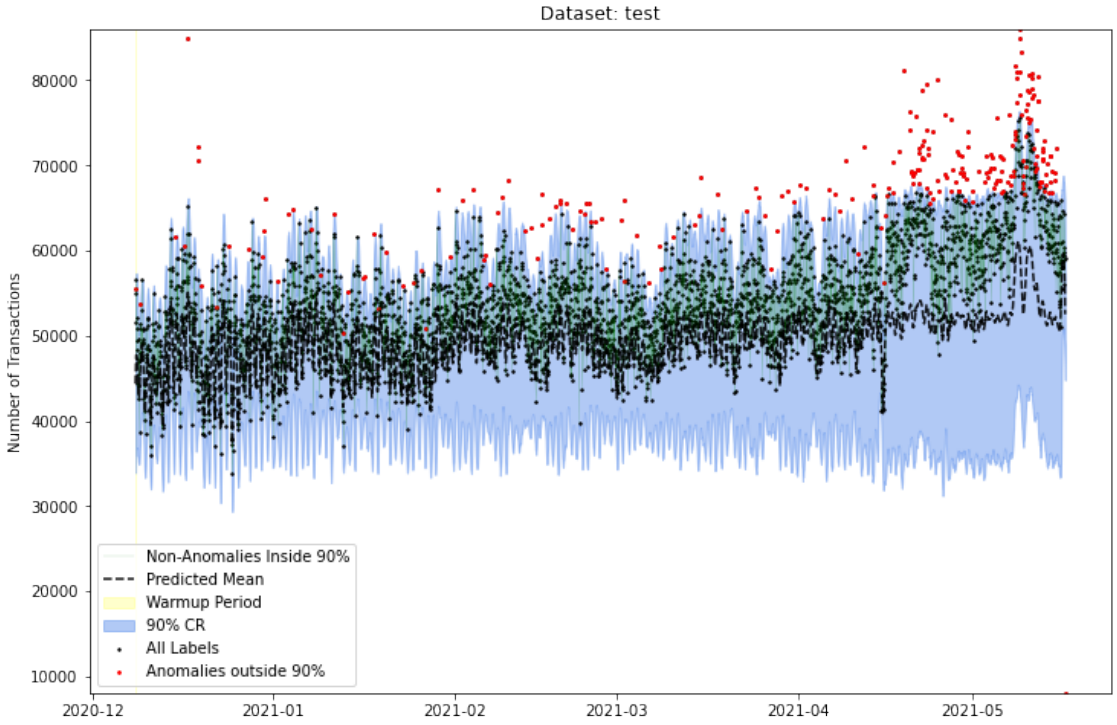
[26]:
```
multi_window.plot_global_forecast(
        train_forecast,
        save_path=global_train_img
    )
```



[27]:
```
multi_window.plot_global_forecast(
        val_forecast,
        save_path=global_val_img
    )
```



[28]:
```
multi_window.plot_global_forecast(
        test_forecast,
        save_path=global_test_img
    )
```



[29]:
```
# val_total_area, val_loss, test_total_area, and test_loss are the ones we care␣
 ↪about for results
performance = dict()
performance['time'] = toc - tic
performance['now'] = now
performance.update(params)
performance['train_size'] = len(multi_window.train)
performance['val_size'] = len(multi_window.val)
performance['test_size'] = len(multi_window.test)
performance['total_data'] = len(multi_window.df)
performance['train'] = feedback_model.evaluate(multi_window.train)
performance['val'] = feedback_model.evaluate(multi_window.val)
performance['test'] = feedback_model.evaluate(multi_window.test, verbose=0)
performance.update(post_checks)
pp.pprint(performance)

# cache performance - saves automatically organized by datetime, distribution
out_file = open(f'metrics/{now}_{distribution}.json', "w")
json.dump(performance, out_file, indent=6)
out_file.close()
```

```
105/105 [==============================] - 4s 35ms/step - loss: -0.5770 -
reconstruction_loss: -0.6197 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00
```

```
30/30 [==============================] - 1s 36ms/step - loss: 5.2584 -
reconstruction_loss: 5.2157 - regularization_loss: 0.0000e+00 - kl_loss:
0.0000e+00
{   'batch_size': 256,
    'beta': 1,
    'dense_cells': None,
    'distribution': 'normal',
    'fname': 'data/ethereum_clean.csv',
    'input_width': 24,
    'label_width': 24,
    'latent_dim': None,
    'lstm_units': 32,
    'max_epochs': 40,
    'min_df': 2.0,
    'now': '02_06_2021_03_07_35',
    'number_states': 30,
    'patience': 2,
    'regularization': None,
    'resample': None,
    'shift': 0,
    't2v_units': 8,
    'test': [31.77865982055664, 0.0, 31.77865982055664, 0.0],
    'test_area_above': 6.155948343334203,
    'test_area_below': 9.062157578919907,
    'test_size': 15,
    'test_total_area': 15.22,
    'time': 140.401602268219,
    'time_index': 1,
    'total_data': 38328,
    'train': [-0.6197210550308228, 0.0, -0.6197210550308228, 0.0],
    'train_area_above': 4.389699578813969,
    'train_area_below': 2.3493067203399307,
    'train_size': 105,
    'train_total_area': 6.74,
    'val': [5.215722560882568, 0.0, 5.215722560882568, 0.0],
    'val_area_above': 1.8889251239238196,
    'val_area_below': 3.6803417688494653,
    'val_size': 30,
    'val_total_area': 5.57}
```

[ ]: