Generate_neural_network_1

August 10, 2023

[ ]:
```
print('-------------------------------------------------')
```

```
-------------------------------------------------
```

1

1.1

(- ***COVID_PSK.csv***)

[ ]:
```
import numpy as np
import pandas as pd

from load_csv_silent import QuickLoad


X,l,ts,df1,df = QuickLoad()
```

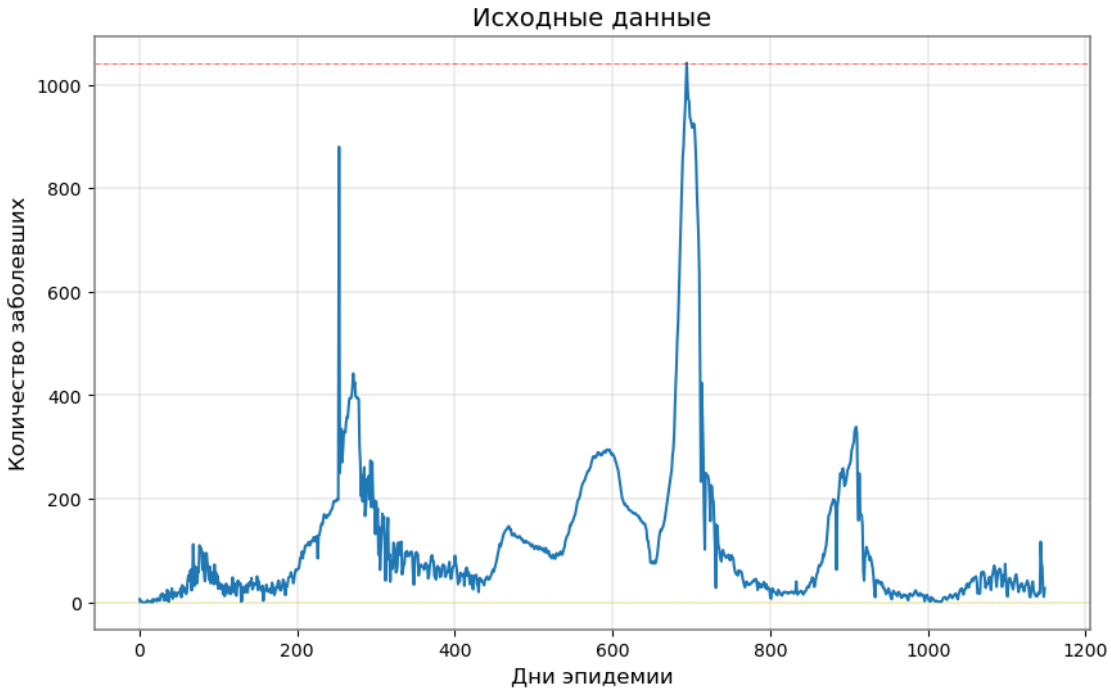
```
e:\Users\Alex\source\repos\TSRevenko\load_csv_silent.py:37:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['Inf_day'] = df1['Infections'].diff().fillna(df1['Infections'])
```

[ ]:
```
import pandas as pd
import numpy as np
import scipy as sp

import matplotlib.pyplot as plt
import seaborn as sns

def set_style(dpi=120,fontsize=12,linewidth=1.
 ↪5,fontsize_xtick=10,fontsize_ytick=10):
    plt.rcParams['figure.dpi'] = dpi
```

```
    # set font size
    plt.rcParams['font.size'] = fontsize
    # set font size on x and y axis
    plt.rcParams['axes.labelsize'] = fontsize-2
    # set font size on thickmark
    plt.rcParams['xtick.labelsize'] = fontsize_xtick
    plt.rcParams['ytick.labelsize'] = fontsize_ytick
    # set grid color
    plt.rcParams['grid.color'] = 'lightgray'
    # set grid line width
    plt.rcParams['grid.linewidth'] = linewidth*0.75
    # set grid alpha
    plt.rcParams['grid.alpha'] = 0.5
    # set border color
    plt.rcParams['axes.edgecolor'] = 'gray'
    # set border width
    plt.rcParams['axes.linewidth'] = linewidth*0.75

def␣
 ↪lineplot(x,y,title=None,xlabel=None,ylabel=None,figsize=(10,6),dpi=120,fontsize=12,linewidt
 ↪5,fontsize_xtick=10,fontsize_ytick=10):
    # set figure size  size
    plt.figure(figsize=figsize)
   ␣
 ↪set_style(dpi=dpi,fontsize=fontsize,linewidth=linewidth,fontsize_xtick=fontsize_xtick,fonts

    plt.plot(x,linewidth=linewidth)
    plt.title(title,fontsize=fontsize+2)
    plt.xlabel(xlabel,fontsize=fontsize)
    plt.ylabel(ylabel,fontsize=fontsize)
    plt.gridcolor='lightblue'
    plt.grid(True)

    # add max value of the line and plot horisontal dotted line at the max value
    plt.axhline(y=max(x),linewidth=linewidth*0.3,color='r',linestyle='-.')
    # add max value of the line and plot horisontal dotted line at the max value
    plt.axhline(y=min(x),linewidth=linewidth*0.3,color='yellow',linestyle='-.')
    plt.show()
def␣
 ↪lineplot2(x1,x2,y,title=None,xlabel=None,ylabel=None,figsize=(10,6),dpi=120,fontsize=12,lin
 ↪5,fontsize_xtick=10,fontsize_ytick=10):
    # set figure size  size
    plt.figure(figsize=figsize)
   ␣
 ↪set_style(dpi=dpi,fontsize=fontsize,linewidth=linewidth,fontsize_xtick=fontsize_xtick,fonts

    plt.plot(x1,linewidth=linewidth)
```

```
    plt.plot(x2,linewidth=linewidth)
    plt.title(title,fontsize=fontsize+2)
    plt.xlabel(xlabel,fontsize=fontsize)
    plt.ylabel(ylabel,fontsize=fontsize)
    plt.gridcolor='lightblue'
    plt.grid(True)

    plt.legend(['x1','x2'],fontsize=fontsize-2,loc='best')

    # add max value of the line and plot horisontal dotted line at the max value
    plt.axhline(y=max(max(x1),max(x2)),linewidth=linewidth*0.
↪3,color='r',linestyle='-.')
    # add max value of the line and plot horisontal dotted line at the max value
    plt.axhline(y=min(min(x1),min(x2)),linewidth=linewidth*0.
↪3,color='yellow',linestyle='-.')
    plt.show()


def lineplot_X_X2(x, x2,␣
↪y,title=None,xlabel=None,ylabel=None,figsize=(10,6),dpi=120,fontsize=12,linewidth=1.
↪5,fontsize_xtick=10,fontsize_ytick=10):
    # set figure size  size
    plt.figure(figsize=figsize)
    ␣
↪set_style(dpi=dpi,fontsize=fontsize,linewidth=linewidth,fontsize_xtick=fontsize_xtick,fonts

    plt.plot(np.append(x,x2),linewidth=linewidth)
    plt.plot(x,linewidth=linewidth)

    plt.title(title,fontsize=fontsize+2)
    plt.xlabel(xlabel,fontsize=fontsize)
    plt.ylabel(ylabel,fontsize=fontsize)
    plt.gridcolor='lightblue'
    plt.grid(True)

    # add max value of the line and plot horisontal dotted line at the max value
    plt.axhline(y=max(x),linewidth=linewidth*0.3,color='r',linestyle='-.')
    # add max value of the line and plot horisontal dotted line at the max value
    plt.axhline(y=max(x2),linewidth=linewidth*0.3,color='orange',linestyle='-.')
    plt.show()
```

[ ]:
```
lineplot(X,l,figsize=(10,6),xlabel="       ",ylabel="            ", title =␣
↪"         ")
```



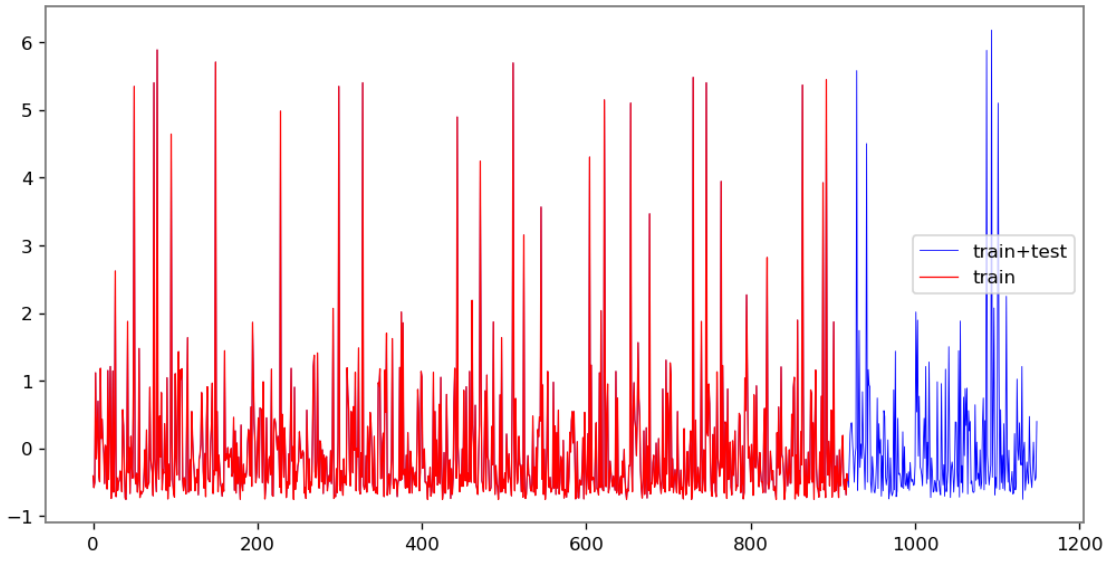
1.2 ,

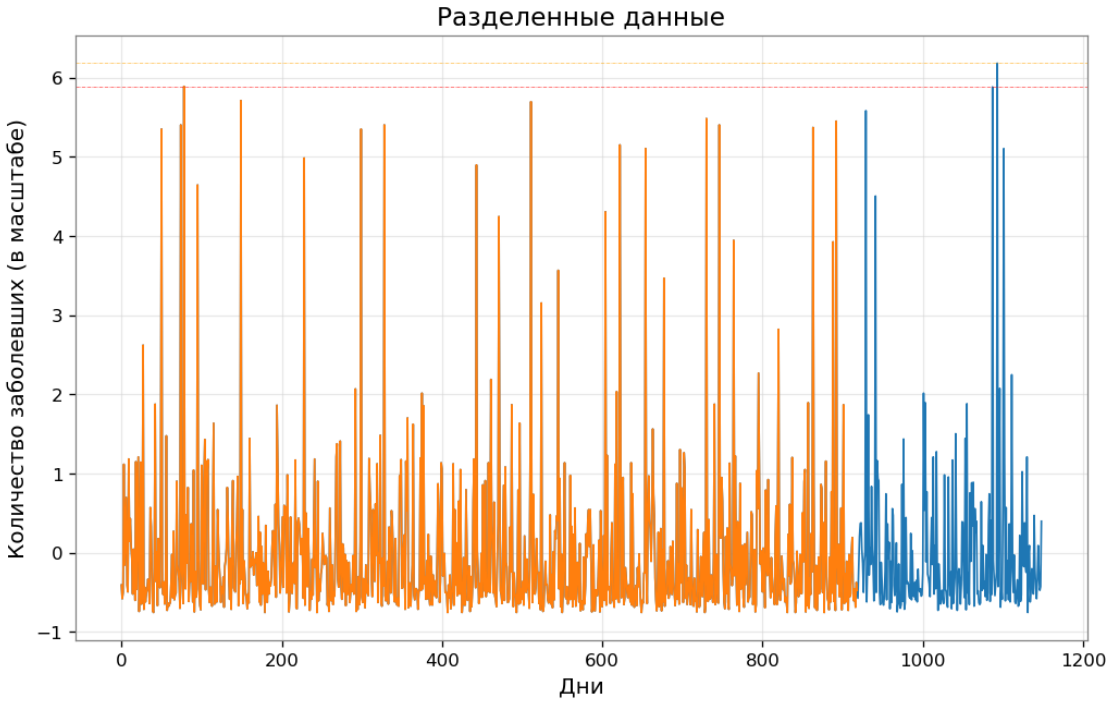
[ ]:
```
# Normalize the data
import numpy as np
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from sklearn.preprocessing import (
    MaxAbsScaler,
    MinMaxScaler,
    Normalizer,
    PowerTransformer,
    QuantileTransformer,
    RobustScaler,
    StandardScaler,
    minmax_scale,
)


scaler = TimeSeriesScalerMeanVariance()
# scaler = PowerTransformer(method='yeo-johnson')
# scaler = StandardScaler()
# scaler = QuantileTransformer()
# xx = np.array(X).reshape(-1, 1)
```

```
X_normalized = scaler.fit_transform([X])

print(X_normalized)

X_n = X_normalized[0]

print(X_n)
lineplot(X_n,l,figsize=(10,6),xlabel="       ",ylabel="              (      )",␣
 ↪title = "             ")
```

```
[[[-0.7141232 ]
  [-0.75404669]
  [-0.73408494]
  …
  [-0.60100664]
  [-0.68085362]
  [-0.57439098]]]
[[-0.7141232 ]
 [-0.75404669]
 [-0.73408494]
 …
 [-0.60100664]
 [-0.68085362]
 [-0.57439098]]
```

[ ]:
```
from tslearn.utils import to_time_series_dataset
from sklearn.model_selection import train_test_split

import numpy as np
import pandas as pd

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_n, X_n, test_size=0.2,␣
 ↪random_state=42)
# set figure size
plt.figure(figsize=(10, 5))

plt.plot(np.append(X_train, X_test), 'b', label='train+test',linewidth=0.5)
plt.plot(X_train, 'r', label='train',linewidth=0.75)
plt.legend(loc='best', fontsize=10)
plt.show()

lineplot_X_X2(X_train, X_test, y_train,figsize=(10,6),xlabel="  ",ylabel="
 ↪      (      )", title = "           ",linewidth = 1)
```





1.3

. tslearn, kerass (tensorflow)
. . .
. tslearn
(inverse transform), . Models ,
.

: - GRU - LSTM

- 1 2

- adam

( ) - MSE

:

- L1_L2 (ElasticNET)
- Dropout

,

LSTM / GRU | 1 / 2 | Units1 | Units2 | / | Dropout / c Dropout |

,

- Arch_{ }
- Layers_{ }

- U1_{ 1}
- U2_{ 2}
- Regul1_{ }
- Dropout1_{ Droput}
- Regul2_{ }
- Dropout2_{ Droput}

[ ]:
```
from keras.models import Sequential
from keras.layers import GRU, LSTM,Dropout,Dense, GaussianNoise
from keras.regularizers import l1_l2
from keras.callbacks import Callback
from keras.callbacks import EarlyStopping
```

, . . - GRU ( ) LSTM - , ( ) . Units, 20 100 ( ).

(ElasticNET, . . L1+L2).

, Dropout ( )

, , .

[ ]:
```
class ArchEnum:
    GRU = 'GRU'
    LSTM = 'LSTM'
    pass

# class layer

class Layer:
    def __init__(self):
        self.Architecture = ArchEnum.GRU
        self.U = 50
        self.Regul = False
        self.L1  = 0.01
        self.L2  = 0.01
        self.Drop = False
        self.Drop_rate = 0.2

    def __str__(self):
        return f"Architecture: {self.Architecture}, U: {self.U}, Regul: {self.
↪Regul}, L1: {self.L1}, L2: {self.L2}, Drop: {self.Drop}, Drop_rate: {self.
↪Drop_rate}"
    def ToString(self):
        return f"Arch_{self.Architecture}_U_{self.U}_Reg_{self.Regul}_L1_{self.
↪L1}_L2_{self.L2}_Drop_{self.Drop}_rate_{self.Drop_rate}"
```

```
  def Constructor(self,architecture=ArchEnum.GRU,U=50,Regul=False,L1=0.
↪01,L2=0.01,Drop=False,drop_level=0.2):
      self.Architecture = architecture
      self.U = U
      self.Regul = Regul
      self.L1 = L1
      self.L2 = L2
      self.Drop = Drop
      self.Drop_rate = drop_level
      pass

  def SimpleGRU(self,U=50):
      self.Constructor(ArchEnum.GRU,U)
      pass
  def SimpleLSTM(self,U=50):
      self.Constructor(ArchEnum.LSTM,U)
      pass
  def GRU_With_Regul(self,U=50,L1=0.01,L2=0.01):
      self.Constructor(ArchEnum.GRU,U,Regul=True,L1=L1,L2=L2)
      pass
  def LSTM_With_Regul(self,U=50,L1=0.01,L2=0.01):
      self.Constructor(ArchEnum.LSTM,U,Regul=True,L1=L1,L2=L2)
      pass
  def GRU_With_Drop(self,U=50,Drop_rate=0.2):
      self.Constructor(ArchEnum.GRU,U,Drop=True,drop_level=Drop_rate)
      pass
  def LSTM_With_Drop(self,U=50,Drop_rate=0.2):
      self.Constructor(ArchEnum.LSTM,U,Drop=True,drop_level=Drop_rate)
      pass
  def GRU_With_Regul_And_Drop(self,U=50,L1=0.01,L2=0.01,drop_level=0.2):
      self.Constructor(ArchEnum.
↪GRU,U,Regul=True,L1=L1,L2=L2,Drop=True,drop_level=Drop_rate)
      pass
  def LSTM_With_Regul_And_Drop(self,U=50,L1=0.01,L2=0.01,Drop_rate=0.2):
      self.Constructor(ArchEnum.
↪LSTM,U,Regul=True,L1=L1,L2=L2,Drop=True,drop_level=Drop_rate)
      pass
```

.

( ) , .

1.
2. (PrepareData,). , , csv (.)
3. (Layer) . , , Dropout, . , Layer 1 2 .
4. Build .

5. Fit.
6. ( ) batch. adam ( ) MSE
7. 0,001 (Stop criteria) 15 (Pation) ,
8. FitAndPlot

:

NeuralNetwork1 = NeuralNetwork()

Layer1 = Layer()

Layer1.GRU_With_Drop()

Layer2 = Layer()

Layer2.SimpleGRU()

NeuralNetwork1.Layers.append(Layer1)

NeuralNetwork1.Layers.append(Layer2)

NeuralNetwork1.PrepareData(X_unscaled=X)

NeuralNetwork1.Build()

NeuralNetwork1.FitAndPlot()

[ ]:
```
from keras.optimizers import SGD
from keras.optimizers import RMSprop
from keras.optimizers import Adadelta
from keras.optimizers import Nadam
from keras.optimizers import Adam
from keras.optimizers import SGD
from keras.optimizers import RMSprop
from keras.optimizers import Adagrad
from keras.optimizers import Adadelta
from keras.optimizers import Adamax

class PlotLearningCurve(Callback):
    def on_train_begin(self, logs=None):
        self.losses = []
        self.val_losses = []
        self.fig, self.ax = plt.subplots()

    def on_epoch_end(self, epoch, logs=None):
        self.losses.append(logs.get('loss'))
        self.val_losses.append(logs.get('val_loss'))

        self.ax.clear()
        self.ax.plot(self.losses, label='train_loss')
        self.ax.plot(self.val_losses, label='val_loss')
```

```
        self.ax.legend()
        self.ax.set_xlabel('Epoch')
        self.ax.set_ylabel('Loss')
        self.fig.canvas.draw()

class NeuralNetwork:
    # STEP1 : INITIALIZE
    def __init__(self):
        self.Name = "NN"
        self.Layers = []
        self.Model = None

        self.X = None
        self.X_normalized = None
        self.X_n = None
        self.X_train  = None
        self.X_test = None

        self.y_train = None
        self.y_test = None

        self.EarlyStop = True

        self.Epochs = 200
        self.Batch_size = 32

        self.Stop_criteria = 0.001
        self.Patience = 15

        pass

    def __str__(self):
        return f"Name: {self.Name}, Layers: {self.Layers}  NLayers: {len(self.
 ↪Layers)}"
        pass
    def ToString(self):
        res = self.Name+'_'
        for layer in self.Layers:
            res += layer.ToString()
            res += "_"
        return res
    # STEP2 : ADD PREPARED DATA
    def␣
 ↪AddData(self,X_unscaled=None,X_normalized=None,X_n=None,X_n_train=None,X_n_test=None,␣
 ↪y_train = None, y_test = None):
        self.X = X_unscaled
        self.X_normalized = X_normalized
```

```
        self.X_n = X_n
        self.X_train = X_n_train
        self.X_test = X_n_test
        self.y_train = y_train
        self.y_test = y_test
        pass
    # STEP2 : ADD DATA, based on single column X from QuickLoad
    def PrepareData(self,X_unscaled):
        self.X = X_unscaled
        scaler = TimeSeriesScalerMeanVariance()
        self.X_normalized = scaler.fit_transform([X])
        self.X_n = self.X_normalized[0]
        # Split the data into training and testing sets
        self.X_train, self.X_test, self.y_train,self. y_test =␣
↪train_test_split(self.X_n, self.X_n, test_size=0.2, random_state=42)
        pass
    # STEP3 : ADD LAYERS
    def AddLayer(self,layer):
        self.Layers.append(layer)
        pass
    # STEP4 : BUILD MODEL WITH PREPARED DATA AND ADDED LAYERS
    def Build(self):
        self.Model = Sequential()
        if len(self.Layers) == 0:
            print("No layers")
            return
        if self.X.any() == None:
            print("No unscaled data")
            return
        if self.X_n.any() == None:
            print("No scaled data")
            return
        if self.X_train.any() == None:
            print("No train data")
            return
        if self.X_test.any() == None:
            print("No data")
            return

        self.Model.add(GaussianNoise(0.1,input_shape=(self.X_normalized.
↪shape[1], X_normalized.shape[2])))


        for i in range(0,len(self.Layers)):
            layer = self.Layers[i]
            if layer.Architecture == ArchEnum.GRU and layer.Regul == False:
                self.Model.add(
```

```
                        GRU(units=layer.U,
                        activation='relu',
                        return_sequences=True,
                        input_shape=(self.X_normalized.shape[1], X_normalized.
↪shape[2])))
            if layer.Architecture == ArchEnum.GRU and layer.Regul == True:
                self.Model.add(
                        GRU(units=layer.U,
                        activation='relu',
                        return_sequences=True,
                        input_shape=(self.X_normalized.shape[1], X_normalized.
↪shape[2]),
                        kernel_regularizer=l1_l2(l1=0.01, l2=0.01)))
            if layer.Architecture == ArchEnum.LSTM and layer.Regul == False:
                self.Model.add(
                        LSTM(units=layer.U,
                        activation='relu',
                        return_sequences=True,
                        input_shape=(self.X_normalized.shape[1], X_normalized.
↪shape[2])))
            if layer.Architecture == ArchEnum.GRU and layer.Regul == True:
                self.Model.add(
                        LSTM(units=layer.U,
                        activation='relu',
                        return_sequences=True,
                        input_shape=(self.X_normalized.shape[1], X_normalized.
↪shape[2]),
                        kernel_regularizer=l1_l2(l1=0.01, l2=0.01)))
            if layer.Drop == True:
                self.Model.add(Dropout(layer.Drop_rate))

        self.Model.add(Dense(1))  # Output layer for regression
        self.Model.compile(optimizer='adam', loss='mse')

    def SetOpt(self, name):
        if name == "adam":
            Opt = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None,␣
↪decay=0.0, amsgrad=False)
        if name == "sgd":
            Opt = SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
        if name == "rmsprop":
            Opt = RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)
        if name == "adagrad":
            Opt = Adagrad(lr=0.01, epsilon=None, decay=0.0)
        if name == "adadelta":
            Opt = Adadelta(lr=1.0, rho=0.95, epsilon=None, decay=0.0)
        if name == "adamax":
```

```
            Opt = Adamax(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None,␣
↪decay=0.0)
        if name == "nadam":
            Opt = Nadam(learning_rate=0.002, beta_1=0.9, beta_2=0.999,␣
↪epsilon=None, schedule_decay=0.004)

        self.Model.compile(optimizer=Opt, loss='mse')
        pass

    def Fit(self):
        # Define the EarlyStopping callback
        early_stopping = EarlyStopping(monitor='loss', patience=self.Patience,␣
↪mode='min', min_delta=self.Stop_criteria)
        # Train the model
        if self.EarlyStop == True:
            self.Model.fit(
                self.X_train, self.y_train, epochs=self.Epochs, batch_size=self.
↪Batch_size, validation_data=(self.X_test, self.y_test),
                callbacks=[early_stopping]
                )
        else:
            self.Model.fit(
                self.X_train, self.y_train, epochs=self.Epochs, batch_size=self.
↪Batch_size, validation_data=(self.X_test, self.y_test)
                )
        pass

    def FitAndPlot(self):
        # Define the EarlyStopping callback
        early_stopping = EarlyStopping(monitor='loss', patience=self.Patience,␣
↪mode='min', min_delta=self.Stop_criteria)
        # Define the custom callback
        plot_callback = PlotLearningCurve()
        # Train the model
        if self.EarlyStop == True:
            self.Model.fit(
            self.X_train, self.y_train, epochs=self.Epochs, batch_size=self.
↪Batch_size, validation_data=(self.X_test, self.y_test),
            callbacks=[plot_callback, early_stopping])
        else:
            self.Model.fit(
            self.X_train, self.y_train, epochs=self.Epochs, batch_size=self.
↪Batch_size, validation_data=(self.X_test, self.y_test),
            callbacks=[plot_callback])

    def BuidAndFit(self):
```

```
        self.Build()
        self.Fit()

    def BuidAndFitAndPlot(self):
        self.Build()
        self.FitAndPlot()

    def Predict(self, plot_data = True, plot_residuals = True):
        self.y_pred =  (self.Model.predict(self.X_normalized))[0]
        self.residuals =(X_normalized[0] - y_pred)

        if plot_data == True:
            lineplot2(self.X_n,self.
↪y_pred,_,title="     ",xlabel="   ",ylabel="     (          )")
        if plot_residuals == True:
            lineplot(self.residuals,_, title="    ",xlabel="   ",ylabel="    ␣
↪(          )")
        pass

    def Postprocess(self, plot_data = True, plot_residuals = True):
        mu = np.mean(self.X)
        var = np.var(self.X)
        sd = np.sqrt(var)
        print(mu,var,sd)
        X_hat = self.y_pred * sd + mu
        print(X_hat)
        Residuals = (X_hat - self.X)[0]
        if plot_data == True:
            lineplot2(self.X,X_hat,_,title="     ",xlabel="   ",ylabel="    ␣
↪(          )")
        if plot_residuals == True:
            lineplot(Residuals,_, title="    ",xlabel="   ",ylabel="     (       ␣
↪    )")
        pass


        pass

    def SaveModel(self,path = None):
        if path == None:
            path = "./Mdl/"+self.ToString()+".h5"
        else:
            path = "./Mdl/" + self.ToString() + ".h5"

        self.Model.save(path)
        pass
```

[ ]:
```
NeuralNetwork1 = NeuralNetwork()
# Typical values

NeuralNetwork1.Epochs = 256
NeuralNetwork1.BatchSize = 64

NeuralNetwork1.EarlyStop = True
NeuralNetwork1.Stop_criteria = 0.000001
NeuralNetwork1.Patience = 25

Layer1 = Layer()
Layer1.GRU_With_Drop(Drop_rate = 0.0125)
Layer1.U = 100

Layer2 = Layer()
Layer2.SimpleGRU()
Layer2.U = 60

NeuralNetwork1.Layers.append(Layer1)
#NeuralNetwork1.Layers.append(Layer2)
```

SetOpt - ( )

[ ]:
```
# NeuralNetwork1.
  ↪AddData(X_unscaled=X,X_normalized=X_normalized,X_n=X_n,X_n_train=X_train,X_n_test=X_test,y_
NeuralNetwork1.PrepareData(X_unscaled=X)
NeuralNetwork1.Build()
NeuralNetwork1.SetOpt(name="nadam")
```

[ ]:
```
# NeuralNetwork1.Fit()
NeuralNetwork1.FitAndPlot()
```

```
Epoch 1/256
WARNING:tensorflow:Model was constructed with shape (None, 1149, 1) for input
KerasTensor(type_spec=TensorSpec(shape=(None, 1149, 1), dtype=tf.float32,
name='gaussian_noise_input'), name='gaussian_noise_input', description="created
by layer 'gaussian_noise_input'"), but it was called on an input with
incompatible shape (None, 1, 1).
WARNING:tensorflow:Model was constructed with shape (None, 1149, 1) for input
KerasTensor(type_spec=TensorSpec(shape=(None, 1149, 1), dtype=tf.float32,
name='gaussian_noise_input'), name='gaussian_noise_input', description="created
by layer 'gaussian_noise_input'"), but it was called on an input with
incompatible shape (None, 1, 1).
14/29 [=============>…] - ETA: 0s - loss: 0.9331
WARNING:tensorflow:Model was constructed with shape (None, 1149, 1) for input
```

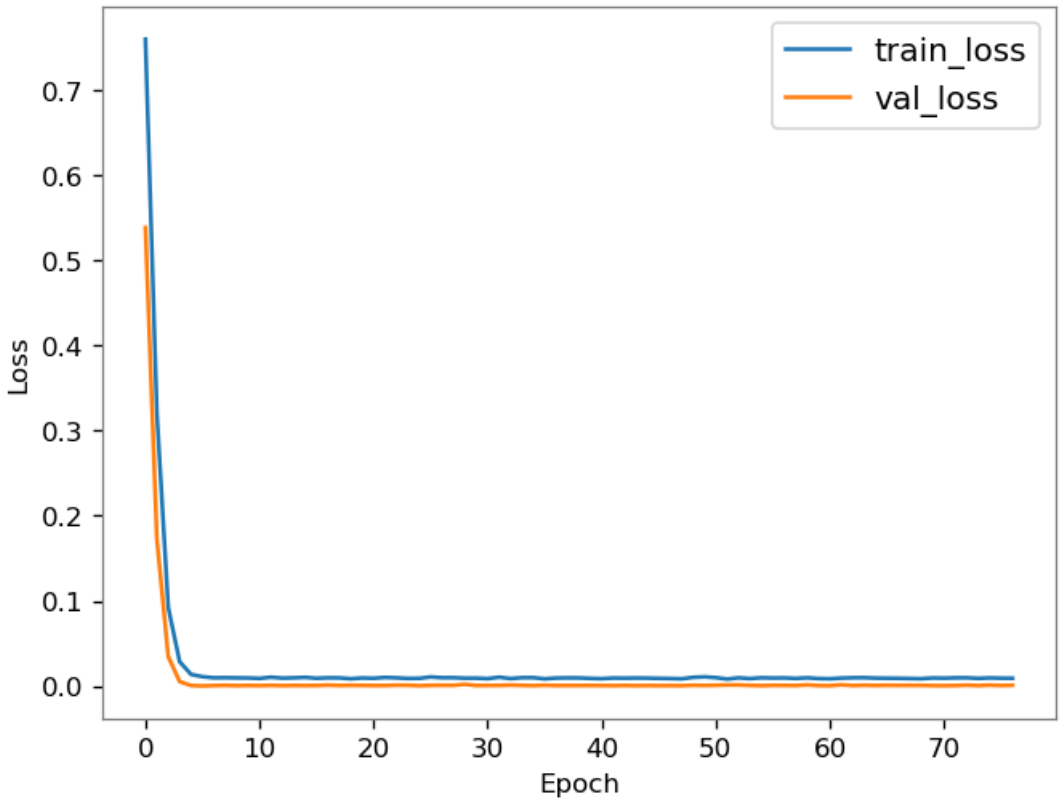
```
KerasTensor(type_spec=TensorSpec(shape=(None, 1149, 1), dtype=tf.float32,
name='gaussian_noise_input'), name='gaussian_noise_input', description="created
by layer 'gaussian_noise_input'"), but it was called on an input with
incompatible shape (None, 1, 1).
29/29 [==============================] - 4s 24ms/step - loss: 0.7591 - val_loss:
0.5380
Epoch 2/256
29/29 [==============================] - 0s 10ms/step - loss: 0.3207 - val_loss:
0.1727
Epoch 3/256
29/29 [==============================] - 0s 13ms/step - loss: 0.0922 - val_loss:
0.0346
Epoch 4/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0286 - val_loss:
0.0052
Epoch 5/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0137 - val_loss:
6.6718e-04
Epoch 6/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0110 - val_loss:
2.5457e-04
Epoch 7/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0098 - val_loss:
6.2039e-04
Epoch 8/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0099 - val_loss:
8.5780e-04
Epoch 9/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0097 - val_loss:
5.6124e-04
Epoch 10/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0096 - val_loss:
7.3099e-04
Epoch 11/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0091 - val_loss:
5.6628e-04
Epoch 12/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0104 - val_loss:
8.3201e-04
Epoch 13/256
29/29 [==============================] - 0s 12ms/step - loss: 0.0094 - val_loss:
5.5936e-04
Epoch 14/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0097 - val_loss:
7.6577e-04
Epoch 15/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0103 - val_loss:
6.2784e-04
```

```
Epoch 16/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0092 - val_loss:
6.9808e-04
Epoch 17/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0098 - val_loss:
0.0012
Epoch 18/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0097 - val_loss:
7.4767e-04
Epoch 19/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0086 - val_loss:
8.8269e-04
Epoch 20/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0096 - val_loss:
8.3192e-04
Epoch 21/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0092 - val_loss:
6.4103e-04
Epoch 22/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0102 - val_loss:
6.7126e-04
Epoch 23/256
29/29 [==============================] - 0s 11ms/step - loss: 0.0097 - val_loss:
0.0010
Epoch 24/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0090 - val_loss:
0.0010
Epoch 25/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0091 - val_loss:
4.7035e-04
Epoch 26/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0107 - val_loss:
8.9990e-04
Epoch 27/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0099 - val_loss:
9.7186e-04
Epoch 28/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0099 - val_loss:
8.8923e-04
Epoch 29/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0093 - val_loss:
0.0021
Epoch 30/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0094 - val_loss:
6.9125e-04
Epoch 31/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0088 - val_loss:
7.8890e-04
```

```
Epoch 32/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0105 - val_loss:
8.0273e-04
Epoch 33/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0088 - val_loss:
0.0012
Epoch 34/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0099 - val_loss:
8.3191e-04
Epoch 35/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0099 - val_loss:
5.6771e-04
Epoch 36/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0086 - val_loss:
9.6468e-04
Epoch 37/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0095 - val_loss:
6.6044e-04
Epoch 38/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0097 - val_loss:
7.0050e-04
Epoch 39/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0096 - val_loss:
6.8314e-04
Epoch 40/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0091 - val_loss:
7.3654e-04
Epoch 41/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0088 - val_loss:
6.7482e-04
Epoch 42/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0095 - val_loss:
5.3725e-04
Epoch 43/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0094 - val_loss:
7.3776e-04
Epoch 44/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0095 - val_loss:
5.3691e-04
Epoch 45/256
29/29 [==============================] - 1s 24ms/step - loss: 0.0094 - val_loss:
6.2655e-04
Epoch 46/256
29/29 [==============================] - 0s 11ms/step - loss: 0.0091 - val_loss:
5.5859e-04
Epoch 47/256
29/29 [==============================] - 0s 11ms/step - loss: 0.0089 - val_loss:
5.8721e-04
```

```
Epoch 48/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0086 - val_loss:
5.0396e-04
Epoch 49/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0103 - val_loss:
9.5679e-04
Epoch 50/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0109 - val_loss:
7.9150e-04
Epoch 51/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0100 - val_loss:
9.4227e-04
Epoch 52/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0083 - val_loss:
0.0013
Epoch 53/256
29/29 [==============================] - 1s 27ms/step - loss: 0.0097 - val_loss:
0.0013
Epoch 54/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0088 - val_loss:
8.6254e-04
Epoch 55/256
29/29 [==============================] - 0s 12ms/step - loss: 0.0098 - val_loss:
5.0834e-04
Epoch 56/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0095 - val_loss:
8.7012e-04
Epoch 57/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0096 - val_loss:
7.9879e-04
Epoch 58/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0091 - val_loss:
6.8777e-04
Epoch 59/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0098 - val_loss:
0.0014
Epoch 60/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0089 - val_loss:
6.0269e-04
Epoch 61/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0086 - val_loss:
5.2794e-04
Epoch 62/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0094 - val_loss:
0.0015
Epoch 63/256
29/29 [==============================] - 1s 31ms/step - loss: 0.0098 - val_loss:
6.4570e-04
```

```
Epoch 64/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0099 - val_loss:
0.0010
Epoch 65/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0094 - val_loss:
7.2651e-04
Epoch 66/256
29/29 [==============================] - 0s 11ms/step - loss: 0.0093 - val_loss:
9.4377e-04
Epoch 67/256
29/29 [==============================] - 0s 11ms/step - loss: 0.0091 - val_loss:
8.4287e-04
Epoch 68/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0089 - val_loss:
8.5765e-04
Epoch 69/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0087 - val_loss:
8.6311e-04
Epoch 70/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0096 - val_loss:
5.5718e-04
Epoch 71/256
29/29 [==============================] - 0s 8ms/step - loss: 0.0094 - val_loss:
5.0001e-04
Epoch 72/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0097 - val_loss:
6.5762e-04
Epoch 73/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0098 - val_loss:
0.0011
Epoch 74/256
29/29 [==============================] - 0s 7ms/step - loss: 0.0091 - val_loss:
5.6626e-04
Epoch 75/256
29/29 [==============================] - 0s 10ms/step - loss: 0.0096 - val_loss:
0.0011
Epoch 76/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0093 - val_loss:
6.7836e-04
Epoch 77/256
29/29 [==============================] - 0s 9ms/step - loss: 0.0092 - val_loss:
9.8390e-04
```



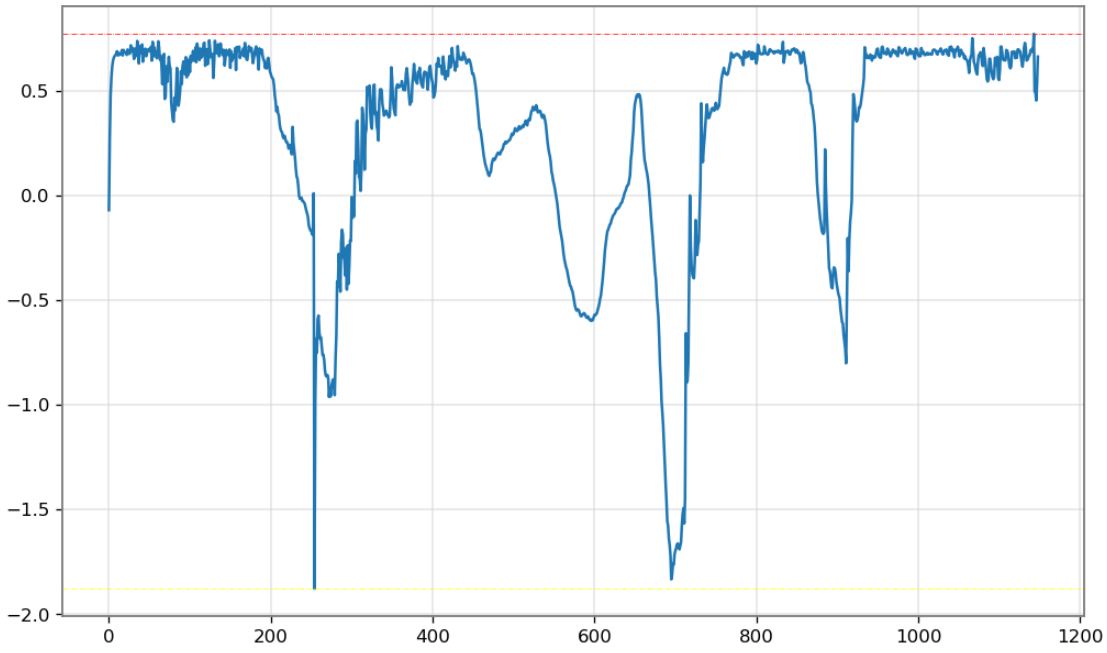
[ ]:
```
NeuralNetwork1.Model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gaussian_noise (GaussianNoi  (None, 1149, 1)          0
 se)

 gru (GRU)                   (None, 1149, 100)         30900

 dropout (Dropout)           (None, 1149, 100)         0
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gaussian_noise (GaussianNoi  (None, 1149, 1)          0
 se)

 gru (GRU)                   (None, 1149, 100)         30900

 dropout (Dropout)           (None, 1149, 100)         0
```

```
 dense (Dense)               (None, 1149, 1)           101

=================================================================
Total params: 31,001
Trainable params: 31,001
Non-trainable params: 0
_________________________________________________________________
```

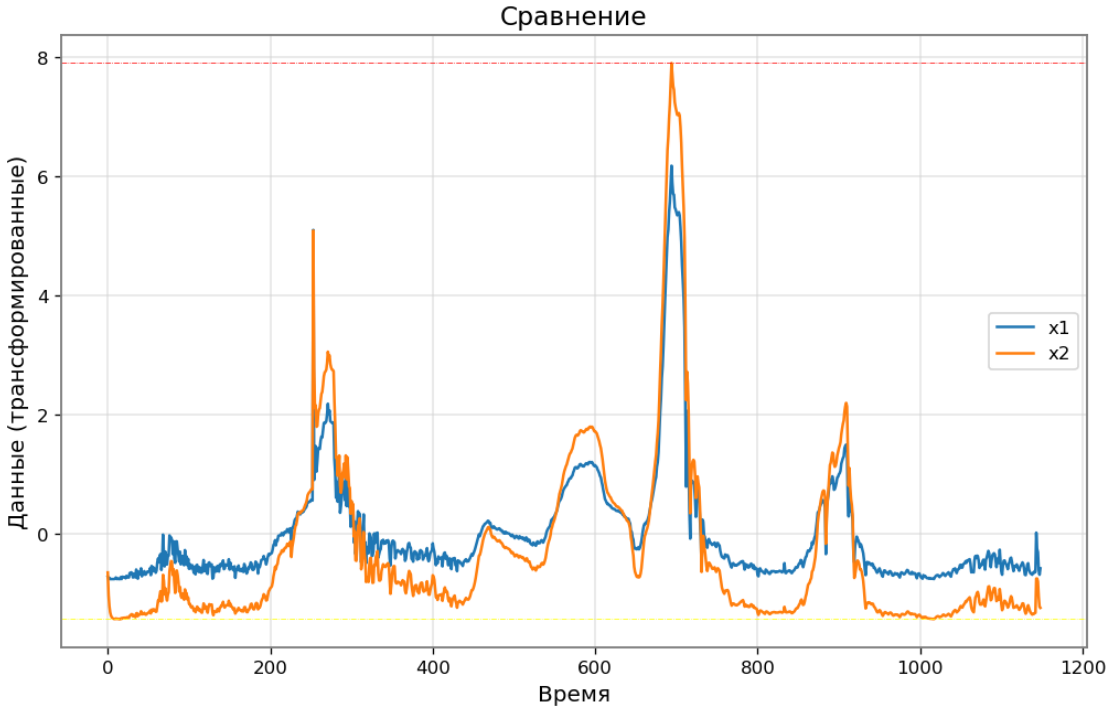
[ ]:
```
y_pred =  (NeuralNetwork1.Model.predict(X_normalized))[0]
residuals =(X_normalized[0] - y_pred)

# Print the accuracy
# lineplot(X_normalized[0],y_pred)
lineplot(residuals,X_normalized[0])
```

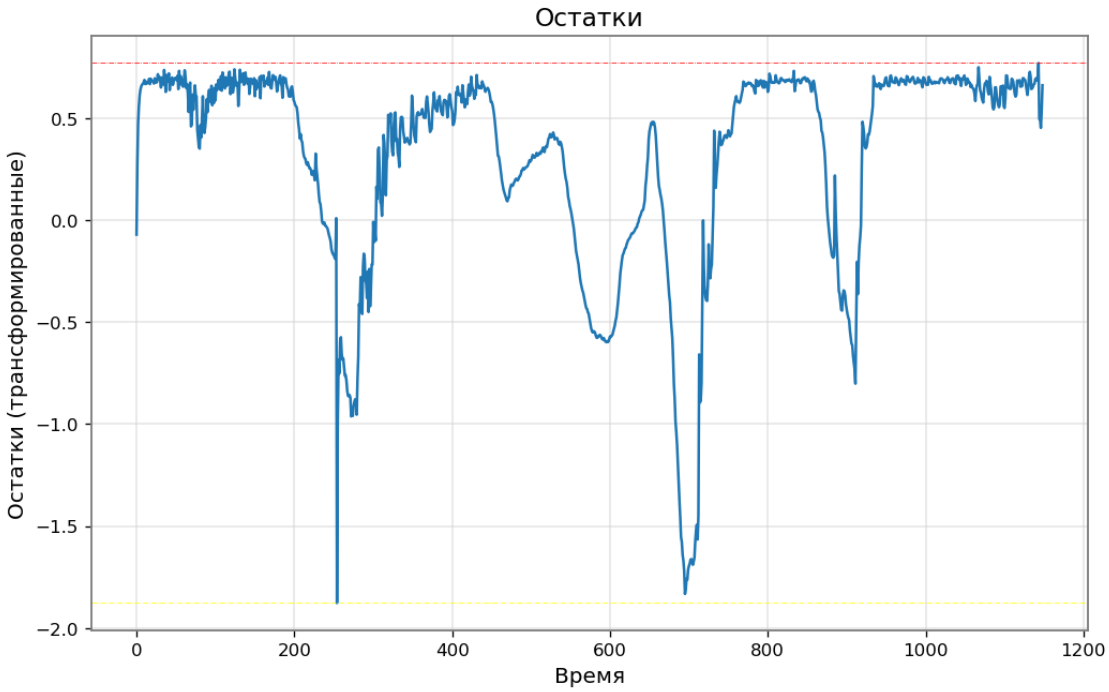
```
1/1 [==============================] - 1s 690ms/step
```



[ ]:
```
NeuralNetwork1.Predict(plot_data = True, plot_residuals = True)
```

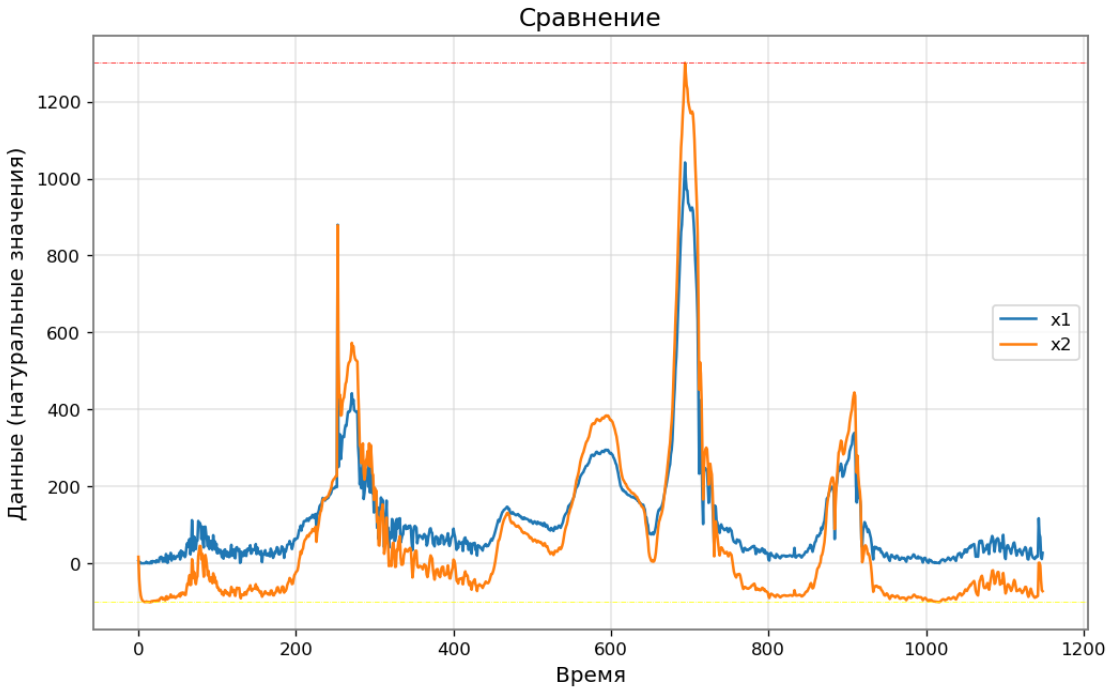
```
1/1 [==============================] - 0s 122ms/step
```

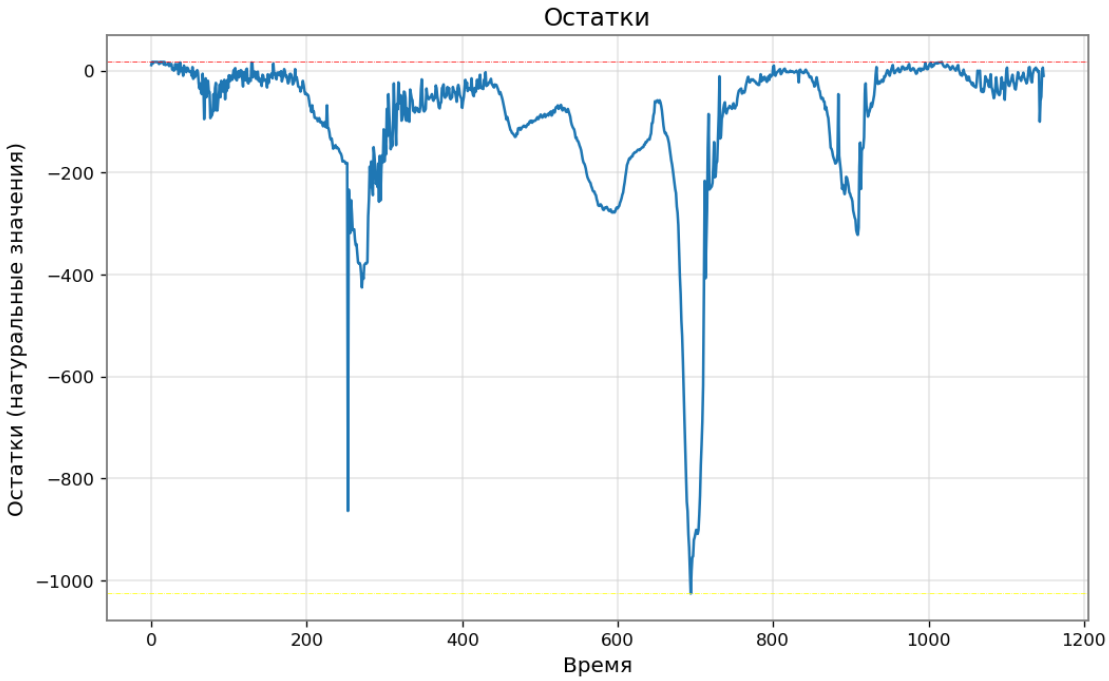





```
[ ]: NeuralNetwork1.Postprocess(plot_data = True, plot_residuals = True)
```

```
113.32375979112271 22586.319896742993 150.2874575496671
[[ 16.639465]
 [-40.718872]
 [-68.73741 ]
 …
 [-45.054337]
 [-69.70392 ]
 [-72.451614]]
```





[ ]:
```
NeuralNetwork1.SaveModel()
```