Software code

Wiem Ben Romdhane

2025-03-26

Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras_tuner as kt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, GRU, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

Import data

```
url="https://raw.githubusercontent.com/Hamrita/Wiem_paper01/refs/heads/main/Data/crypto.csv"
crypto=pd.read_csv(url)
BTC=df.iloc[:,1].values.astype('float32')
BTC=BTC.reshape(BTC.shape[0],1)
ETH=df.iloc[:,2].values.astype('float32')
ETH=ETH.reshape(ETH.shape[0],1)
LTC=df.iloc[:,3].values.astype('float32')
LTC=LTC.reshape(LTC.shape[0],1)
```

Split data

```
# Split into train and test sets
nn=len(BTC)
train_size = int(nn * 0.9)
test_size = nn - train_size
trainBTC, testBTC = BTC[0:train_size], BTC[train_size:nn]
trainETH, testETH = ETH[0:train_size], ETH[train_size:nn]
trainLTC, testLTC = LTC[0:train_size], LTC[train_size:nn]
```

Scale data

```
scalerBTC = MinMaxScaler(feature_range=(0, 1))
trainBTC_scaled = scalerBTC.fit_transform(trainBTC)
testBTC_scaled=scalerBTC.transform(testBTC)
scalerETH = MinMaxScaler(feature_range=(0, 1))
trainETH_scaled = scalerETH.fit_transform(trainETH)
testETH_scaled=scalerETH.transform(testETH)
scalerLTC = MinMaxScaler(feature_range=(0, 1))
trainLTC_scaled = scalerLTC.fit_transform(trainLTC)
testLTC_scaled=scalerLTC.transform(testLTC)
```

Create sequences

```
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), :]  #get look_back sequences
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0]) #get the target after look_back sequences
    return np.array(dataX), np.array(dataY)
```

```
look_back = 10  # Sequence length (tunable)
trainX_BTC, trainY_BTC = create_dataset(trainBTC_scaled, look_back)
testX_BTC, testY_BTC = create_dataset(testBTC_scaled, look_back)

trainX_ETH, trainY_ETH = create_dataset(trainETH_scaled, look_back)
testX_ETH, testY_ETH = create_dataset(testETH_scaled, look_back)

trainX_LTC, trainY_LTC = create_dataset(trainLTC_scaled, look_back)
testX_LTC, testY_LTC = create_dataset(testLTC_scaled, look_back)
```

```
def build_model1(hp):
    model = Sequential()
    model.add(LSTM(hp.Int('input_unit',min_value=50,max_value=200, step=50),return_sequences=True, inpu
    for i in range(hp.Int('n_layers', 1, 3)):
        model.add(LSTM(hp.Int(f'lstm_{i}_units',min_value=50,max_value=200, step=50),return_sequences=T
    model.add(LSTM(hp.Int('layer_2_neurons',min_value=50,max_value=200, step=50)))
    model.add(Dropout(hp.Float('Dropout_rate',min_value=0.1,max_value=0.6,step=0.1)))
    model.add(Dense(1, activation=hp.Choice('dense_activation',values=['relu', 'sigmoid'],default='relu
    learning_rate = hp.Float('learning_rate', min_value=1e-4, max_value=1e-2, sampling='log')
    model.compile(loss='mean_squared_error', optimizer=Adam(learning_rate=learning_rate),metrics = ['ma
    return model
```

```
tuner1= kt.RandomSearch(
        build_model1,
        objective='loss',
        max_trials=10,
        executions_per_trial=1,
```

```
        overwrite=True
        )

# Early stopping callback
early_stopper = EarlyStopping(
    monitor='loss',
    min_delta=0.01,
    patience=5,
    restore_best_weights=True
)
```

```
tuner1.search(
        x=trainX_BTC,
        y=trainY_BTC,
        epochs=100,
        batch_size=32,
        validation_data=(testX_BTC,testY_BTC),
        callbacks=[early_stopper]
)
```

```
best_model_btc = tuner1.get_best_models(num_models=1)[0]
```

```
best_hyperparameters_btc = tuner1.get_best_hyperparameters(num_trials=1)[0]
best_hyperparameters_btc.values
```

```
fit_model_btc=best_model_btc.fit(trainX_BTC,trainY_BTC,
                            epochs=200, verbose=0 )
```

```
fit_btc=best_model_btc.predict(trainX_BTC)
for_btc=best_model_btc.predict(testX_BTC)
btc_fit=scalerBTC.inverse_transform(fit_btc)
btc_pred=scalerBTC.inverse_transform(for_btc)

btc_tr=scalerBTC.inverse_transform(trainY_BTC.reshape(-1,1))
btc_te=scalerBTC.inverse_transform(testY_BTC.reshape(-1,1))
```

```
btc_fit_lstm=pd.DataFrame(btc_fit)
btc_pred_lstm=pd.DataFrame(btc_pred)
btc_fit_lstm.to_csv('Data/btc_fit_lstm.csv', index=False)
btc_pred_lstm.to_csv('Data/btc_pred_lstm.csv', index=False)
btc_tr_lstm=pd.DataFrame(btc_tr)
btc_te_lstm=pd.DataFrame(btc_te)
btc_tr_lstm.to_csv('Data/btc_tr_lstm.csv', index=False)
btc_te_lstm.to_csv('Data/btc_te_lstm.csv', index=False)
```

```
plt.plot(btc_te,label='Actual (Test)')
plt.plot(btc_pred,linestyle="dashed",label='Predicted (Test)')
plt.legend()
plt.show()
```

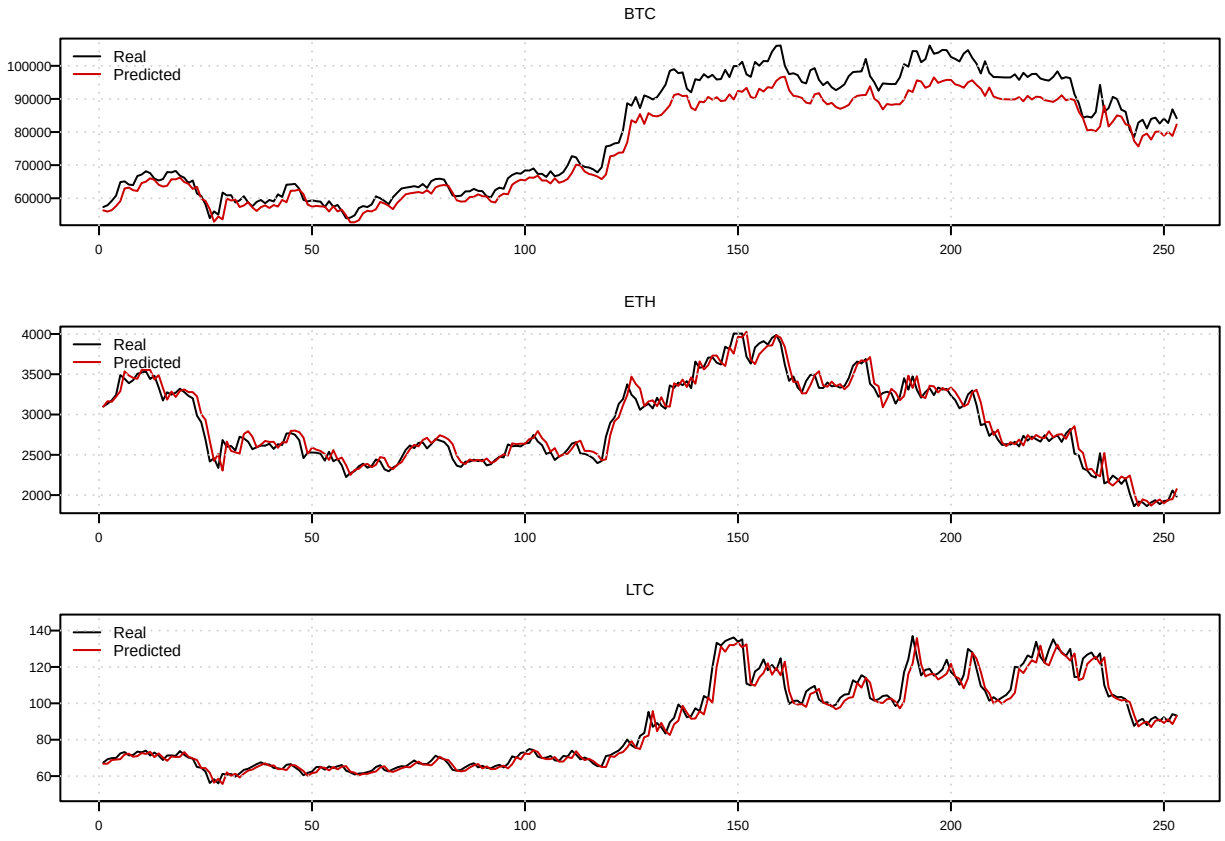
```
lstm100=Sequential()
lstm100.add(LSTM(units=100, return_sequences=True, input_shape=(trainX_BTC.shape[1], 1)))
lstm100.add(LSTM(units=100, return_sequences=True))
lstm100.add(LSTM(units=100))
lstm100.add(Dense(units=1))
lstm100.compile(optimizer='adam', loss='mean_squared_error')
his_100=lstm100.fit(trainX_BTC, trainY_BTC, epochs=100, batch_size=32,
           validation_data=(testX_BTC, testY_BTC))
```

```
fit_btc100=lstm100.predict(trainX_BTC)
for_btc100=lstm100.predict(testX_BTC)
btc_fit100=scalerBTC.inverse_transform(fit_btc100)
btc_pred100=scalerBTC.inverse_transform(for_btc100)

btc_tr=scalerBTC.inverse_transform(trainY_BTC.reshape(-1,1))
btc_te=scalerBTC.inverse_transform(testY_BTC.reshape(-1,1))

btc_fit100.to_csv('Data/btc_fit_lstm100.csv', index=False)
btc_pred100.to_csv('Data/btc_pred_lstm100.csv', index=False)
```

Combined Methods

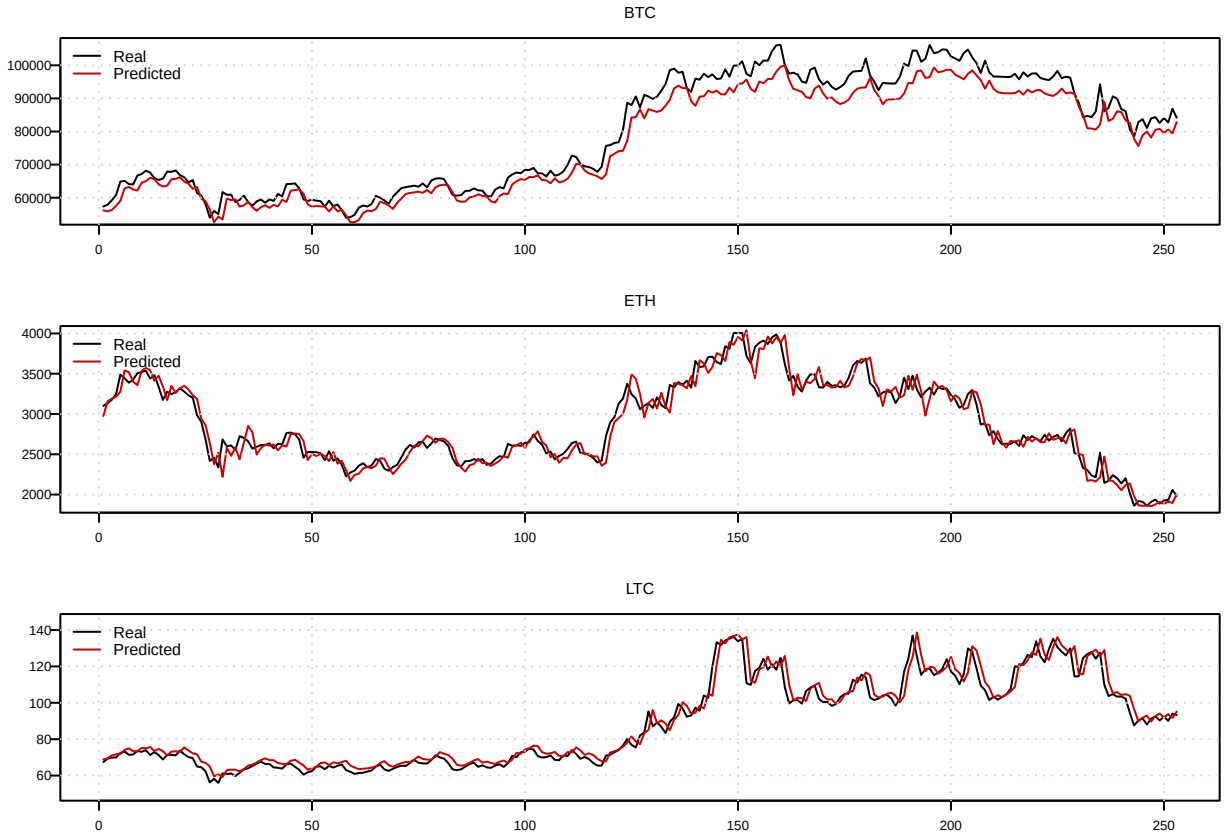
```
# R code
train_btc=read.csv("Data/btc_tr_lstm.csv")$X0
test_btc=read.csv("Data/btc_te_lstm.csv")$X0
LSTM_fit_btc=read.csv("Data/btc_fit_lstm100.csv")$X0
LSTM_pred_btc=read.csv("Data/btc_pred_lstm100.csv")$X0
GRU_fit_btc=read.csv("Data/btc_fit_gru100.csv")$X0
GRU_pred_btc=read.csv("Data/btc_pred_gru100.csv")$X0
############
train_eth=read.csv("Data/eth_tr_lstm.csv")$X0
test_eth=read.csv("Data/eth_te_lstm.csv")$X0
LSTM_fit_eth=read.csv("Data/eth_fit_lstm.csv")$X0
LSTM_pred_eth=read.csv("Data/eth_pred_lstm.csv")$X0
GRU_fit_eth=read.csv("Data/eth_fit_gru.csv")$X0
GRU_pred_eth=read.csv("Data/eth_pred_gru.csv")$X0
##############
train_ltc=read.csv("Data/ltc_tr_gru.csv")$X0
test_ltc=read.csv("Data/ltc_te_gru.csv")$X0
LSTM_fit_ltc=read.csv("Data/ltc_fit_lstm.csv")$X0
LSTM_pred_ltc=read.csv("Data/ltc_pred_lstm.csv")$X0
GRU_fit_ltc=read.csv("Data/ltc_fit_gru.csv")$X0
GRU_pred_ltc=read.csv("Data/ltc_pred_gru.csv")$X0
#####################################
# Graphs
#####################################
options(scipen = 999)
par(mfrow=c(3,1), mar=c(2,2.5,2,1), mgp=c(0.8,0.35,0), cex.axis=0.6, tck = -0.05)
plot(test_btc, type="l", las=1 , xlab="", ylab="" )
lines(LSTM_pred_btc , col="red3" )
```

```
legend("topleft", bty="n", col=c(1,"red3"), legend=c("Real", "Predicted"), lty=1, cex=0.7 )
grid()
title("BTC", cex.main=0.8, font.main=1)
################################################
plot(test_eth, type="l", las=1 , xlab="", ylab="" )
lines(LSTM_pred_eth, col="red3" )
legend("topleft", bty="n", col=c(1,"red3"), legend=c("Real", "Predicted"), lty=1, cex=0.7 )
grid()
title("ETH", cex.main=0.8, font.main=1)
################################################
plot(test_ltc, type="l", las=1 , xlab="", ylab="", ylim=c(50,145) )
lines(LSTM_pred_ltc, col="red3" )
legend("topleft", bty="n", col=c(1,"red3"), legend=c("Real", "Predicted"), lty=1, cex=0.7 )
grid()
title("LTC", cex.main=0.8, font.main=1)
```




```
################################################

par(mfrow=c(3,1), mar=c(2,2.5,2,1), mgp=c(0.8,0.35,0), cex.axis=0.6, tck = -0.05)
plot(test_btc, type="l", las=1 , xlab="", ylab="" )
lines(GRU_pred_btc , col="red3" )
legend("topleft", bty="n", col=c(1,"red3"), legend=c("Real", "Predicted"), lty=1, cex=0.7 )
grid()
title("BTC", cex.main=0.8, font.main=1)
################################################
```

```r
plot(test_eth, type="l", las=1 , xlab="", ylab="" )
lines(GRU_pred_eth, col="red3" )
legend("topleft", bty="n", col=c(1,"red3"), legend=c("Real", "Predicted"), lty=1, cex=0.7 )
grid()
title("ETH", cex.main=0.8, font.main=1)
################################################
plot(test_ltc, type="l", las=1 , xlab="", ylab="", ylim=c(50,145))
lines(GRU_pred_ltc, col="red3" )
legend("topleft", bty="n", col=c(1,"red3"), legend=c("Real", "Predicted"), lty=1, cex=0.7 )
grid()
title("LTC", cex.main=0.8, font.main=1)
```

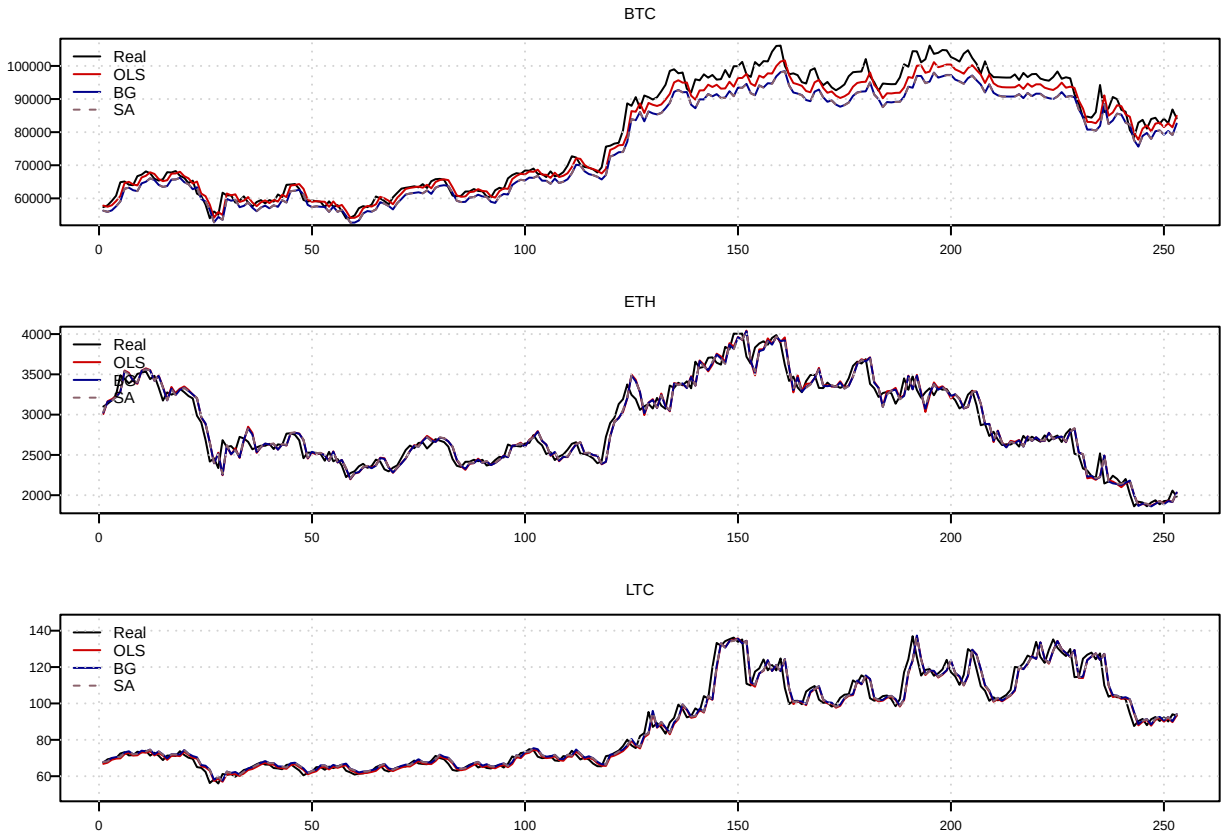



```r
#####################################
btc_comb_tr=cbind(LSTM_fit_btc, GRU_fit_btc )
btc_comb_tst=cbind(LSTM_pred_btc, GRU_pred_btc)
##############
eth_comb_tr=cbind(LSTM_fit_eth, GRU_fit_eth )
eth_comb_tst=cbind(LSTM_pred_eth, GRU_pred_eth)
################################################
ltc_comb_tr=cbind(LSTM_fit_ltc, GRU_fit_ltc )
ltc_comb_tst=cbind(LSTM_pred_ltc, GRU_pred_ltc)
require(ForecastComb, quietly=T)
```

```
## Warning: le package 'ForecastComb' a été compilé avec la version R 4.4.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
btc_df=foreccomb(train_btc,btc_comb_tr,test_btc, btc_comb_tst)
eth_df=foreccomb(train_eth,eth_comb_tr,test_eth, eth_comb_tst)
ltc_df=foreccomb(train_ltc,ltc_comb_tr,test_ltc, ltc_comb_tst)
# OLS
ols_btc=comb_OLS(btc_df)
ols_eth=comb_OLS(eth_df)
ols_ltc=comb_OLS(ltc_df)
# BG
bg_btc=comb_BG(btc_df)
bg_eth=comb_BG(eth_df)
bg_ltc=comb_BG(ltc_df)
# SA
sa_btc=comb_SA(btc_df)
sa_eth=comb_SA(eth_df)
sa_ltc=comb_SA(ltc_df)

options(scipen = 999)
par(mfrow=c(3,1), mar=c(2,2.5,2,1), mgp=c(0.8,0.35,0), cex.axis=0.6, tck = -0.05)
plot(test_btc, type="l", las=1 , xlab="", ylab="" )
lines(ols_btc$Forecasts_Test, col="red3" )
lines(bg_btc$Forecasts_Test, col="blue4" )
lines(sa_btc$Forecasts_Test, col="pink4", lty=2 )
legend("topleft", bty="n", col=c(1,"red3", "blue4", "pink4"), legend=c("Real", "OLS", "BG", "SA"), lty=
grid()
title("BTC", cex.main=0.8, font.main=1)
#####################################################
plot(test_eth, type="l", las=1 , xlab="", ylab="" )
lines(ols_eth$Forecasts_Test, col="red3" )
lines(bg_eth$Forecasts_Test, col="blue4" )
lines(sa_eth$Forecasts_Test, col="pink4", lty=2 )
legend("topleft", bty="n", col=c(1,"red3", "blue4", "pink4"), legend=c("Real", "OLS", "BG", "SA"), lty=
grid()
title("ETH", cex.main=0.8, font.main=1)
#####################################################
plot(test_ltc, type="l", las=1 , xlab="", ylab="", ylim=c(50,145) )
lines(ols_ltc$Forecasts_Test, col="red3" )
lines(bg_ltc$Forecasts_Test, col="blue4" )
lines(sa_ltc$Forecasts_Test, col="pink4", lty=2 )
legend("topleft", bty="n", col=c(1,"red3", "blue4", "pink4"), legend=c("Real", "OLS", "BG", "SA"), lty=
grid()
title("LTC", cex.main=0.8, font.main=1)
```




```
metriks=function(x,y){
  require(Metrics, quietly = T)
  rr=c(RMSE=rmse(x,y), RMAE=mae(x,y)**0.5, MAPE=mape(x,y)*100)
  rr
  }
```

```
acc_btc_lstm=c(metriks(train_btc,LSTM_fit_btc),metriks(test_btc,LSTM_pred_btc))
acc_btc_gru=c(metriks(train_btc,GRU_fit_btc),metriks(test_btc,GRU_pred_btc))
acc_btc_ols=c(metriks(train_btc,ols_btc$Fitted),metriks(test_btc,ols_btc$Forecasts_Test))
acc_btc_bg=c(metriks(train_btc,bg_btc$Fitted),metriks(test_btc,bg_btc$Forecasts_Test))
acc_btc_sa=c(metriks(train_btc,sa_btc$Fitted),metriks(test_btc,sa_btc$Forecasts_Test))
t1=rbind(acc_btc_lstm, acc_btc_gru, acc_btc_ols, acc_btc_bg, acc_btc_sa)
```

```
acc_eth_lstm=c(metriks(train_eth,LSTM_fit_eth),metriks(test_eth,LSTM_pred_eth))
acc_eth_gru=c(metriks(train_eth,GRU_fit_eth),metriks(test_eth,GRU_pred_eth))
acc_eth_ols=c(metriks(train_eth,ols_eth$Fitted),metriks(test_eth,ols_eth$Forecasts_Test))
acc_eth_bg=c(metriks(train_eth,bg_eth$Fitted),metriks(test_eth,bg_eth$Forecasts_Test))
acc_eth_sa=c(metriks(train_eth,sa_eth$Fitted),metriks(test_eth,sa_eth$Forecasts_Test))
t2=rbind(acc_eth_lstm, acc_eth_gru, acc_eth_ols, acc_eth_bg, acc_eth_sa)
```

```
acc_ltc_lstm=c(metriks(train_ltc,LSTM_fit_ltc),metriks(test_ltc,LSTM_pred_ltc))
acc_ltc_gru=c(metriks(train_ltc,GRU_fit_ltc),metriks(test_ltc,GRU_pred_ltc))
acc_ltc_ols=c(metriks(train_ltc,ols_ltc$Fitted),metriks(test_ltc,ols_ltc$Forecasts_Test))
acc_ltc_bg=c(metriks(train_ltc,bg_ltc$Fitted),metriks(test_ltc,bg_ltc$Forecasts_Test))
```

```r
acc_ltc_sa=c(metriks(train_ltc,sa_ltc$Fitted),metriks(test_ltc,sa_ltc$Forecasts_Test))
t3=rbind(acc_ltc_lstm, acc_ltc_gru, acc_ltc_ols, acc_ltc_bg, acc_ltc_sa)
```

```r
cn=c("LSTM", "GRU", "Comb_OLS", "Comb_BG", "Comb_SA")
row.names(t1)=row.names(t2)=row.names(t3)=NULL
rn=rep(cn,3)
tab=data.frame(cbind(rn,round(rbind(t1,t2,t3),3)))
require(dplyr, quietly = T)
```

```
##
## Attachement du package : 'dplyr'
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
require(kableExtra, quietly = T)
```

```
## Warning: le package 'kableExtra' a été compilé avec la version R 4.4.3
```

```
##
## Attachement du package : 'kableExtra'
```

```
## L'objet suivant est masqué depuis 'package:dplyr':
##
##     group_rows
```

```r
colnames(tab)[1]=""
kbl(tab, digits=2, align="c", booktabs = T) %>%
  pack_rows("BTC", 1, 5) %>%
  pack_rows("ETH", 6, 10) %>%
  pack_rows("LTC", 11, 15) %>%
  add_header_above(c(" " = 1, "Train" = 3, "Test" = 3))
```

	Train			Test		
	RMSE	RMAE	MAPE	RMSE.1	RMAE.1	MAPE.1
BTC						
LSTM	1390.264	31.308	4.563	5358.839	66.428	5.114
GRU	1308.704	28.497	3.073	4311.471	59.87	4.282
Comb_OLS	1033.683	24.165	2.396	2987.408	47.695	2.693
Comb_BG	1340.918	29.712	3.662	4786.434	63.029	4.672
Comb_SA	1343.37	29.799	3.709	4818.105	63.227	4.697
ETH						
LSTM	74.317	7.253	8.811	114.527	9.216	2.99
GRU	62.238	6.376	4.426	125.011	9.693	3.264
Comb_OLS	59.769	6.114	4.288	120.632	9.469	3.115
Comb_BG	62.258	6.478	5.764	117.045	9.313	3.018
Comb_SA	63.24	6.556	6.15	115.978	9.262	2.987
LTC						
LSTM	6.586	1.948	3.781	4.845	1.792	3.341
GRU	6.118	2.02	4.749	4.977	1.856	3.81
Comb_OLS	5.801	1.845	3.424	4.728	1.755	3.214
Comb_BG	5.974	1.914	3.981	4.698	1.758	3.261
Comb_SA	5.99	1.91	3.941	4.691	1.755	3.244

```
require(dplyr, quietly = T)
require(kableExtra, quietly = T)
collapse_rows_dt <- data.frame(C1 = c(rep("a", 10), rep("b", 5)),
C2 = c(rep("c", 7), rep("d", 3), rep("c", 2), rep("d", 3)),
C3 = 1:15,
C4 = sample(c(0,1), 15, replace = TRUE))
kbl(collapse_rows_dt[-1], align = "c", booktabs = T) %>%
column_spec(1, bold = T, width = "5em") %>%
row_spec(c(1:7, 11:12) - 1, extra_latex_after = "\\rowcolor{gray!6}") %>%
collapse_rows(1, latex_hline = "none")
```