BITCAMP: AI FINTECH DEVELPER TRAINING COURSE 1(2021.7-2022.1)

A Prediction Model for EU ETS Carbon Credit

- Application of Machine Learning & Deep Learning

Park Sinae

Contents

Motivation

Motivation

종목명	현재가
KODEX 유럽탄소배출권선물ICE(H) 코스피	13,810
SOL 유럽탄소배출권선물S&P(H) 코스피	13,800
SOL 글로벌탄소배출권선물IHS(합성) 코스피	13,025
HANARO 글로벌탄소배출권선물ICE(합성) 코스피	12,760
미래에셋 S&P 유럽탄소배출권 선물 ETN 코스피	14,670
TRUE S&P 유럽탄소배출권 선물 ETN(H) 코스피	14,700
메리츠 S&P 유럽탄소배출권 선물 ETN(H) 코스피	14,700
메리츠 S&P 유럽탄소배출권 선물 ETN 코스피	14,675

ETFs

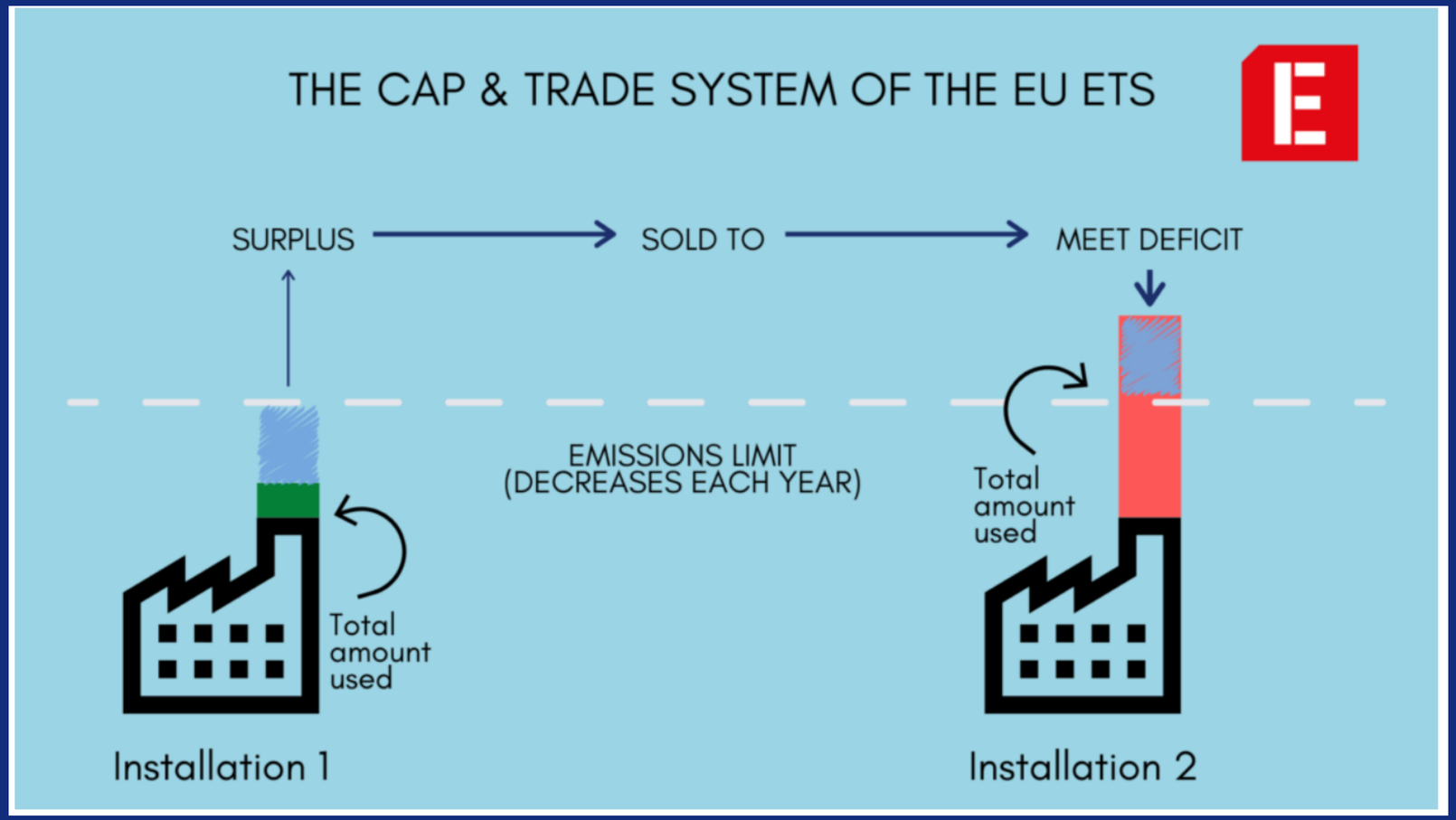
ETNs

Recent enlisting of ETFs/ETNs exposure to global carbon credits

- **4 ETFs exposure to EU ETS Carbon Credit(inception date: 21/9/30)**
- **4 ETNS exposure to EU ETS Carbon Credit(inception date: 21/11/8)**

Background

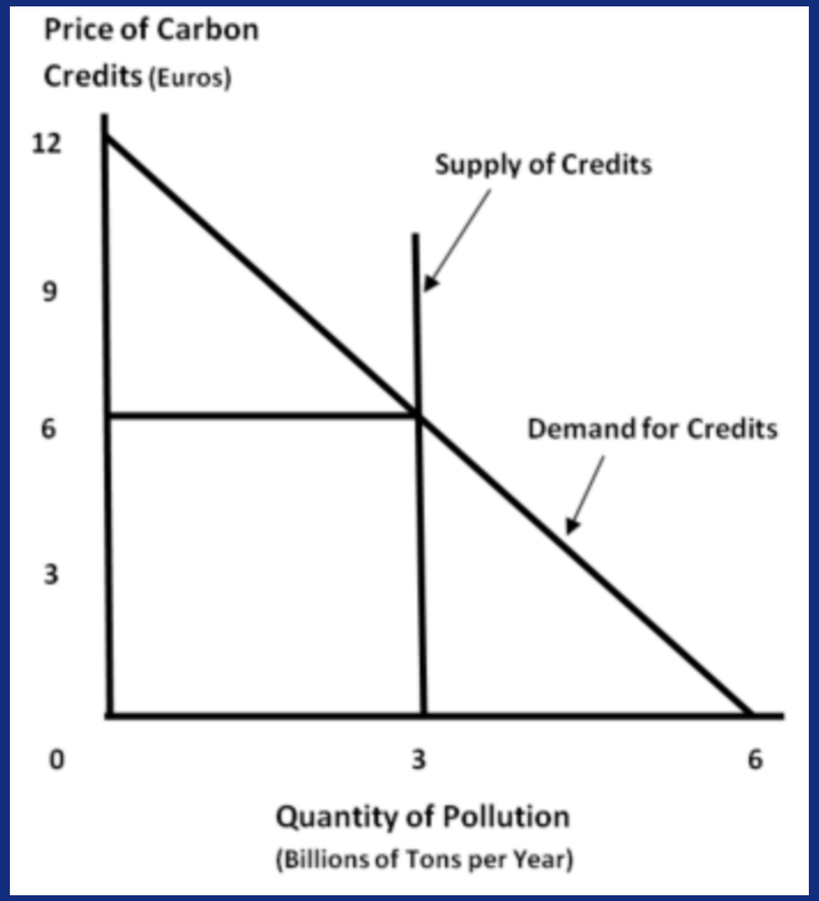
Background: Cap and Trade System



- Total emission cap is fixed by institutions
- Yearly emission limit is allocated to individual firms
- Firms can trade the allowance upon their affordability (capability of tech, yearly production strategy and etc.)

https://www.investigate-europe.eu/en/2020/eu-emissions-trading-scheme-explained/

Background: Cap and Trade System

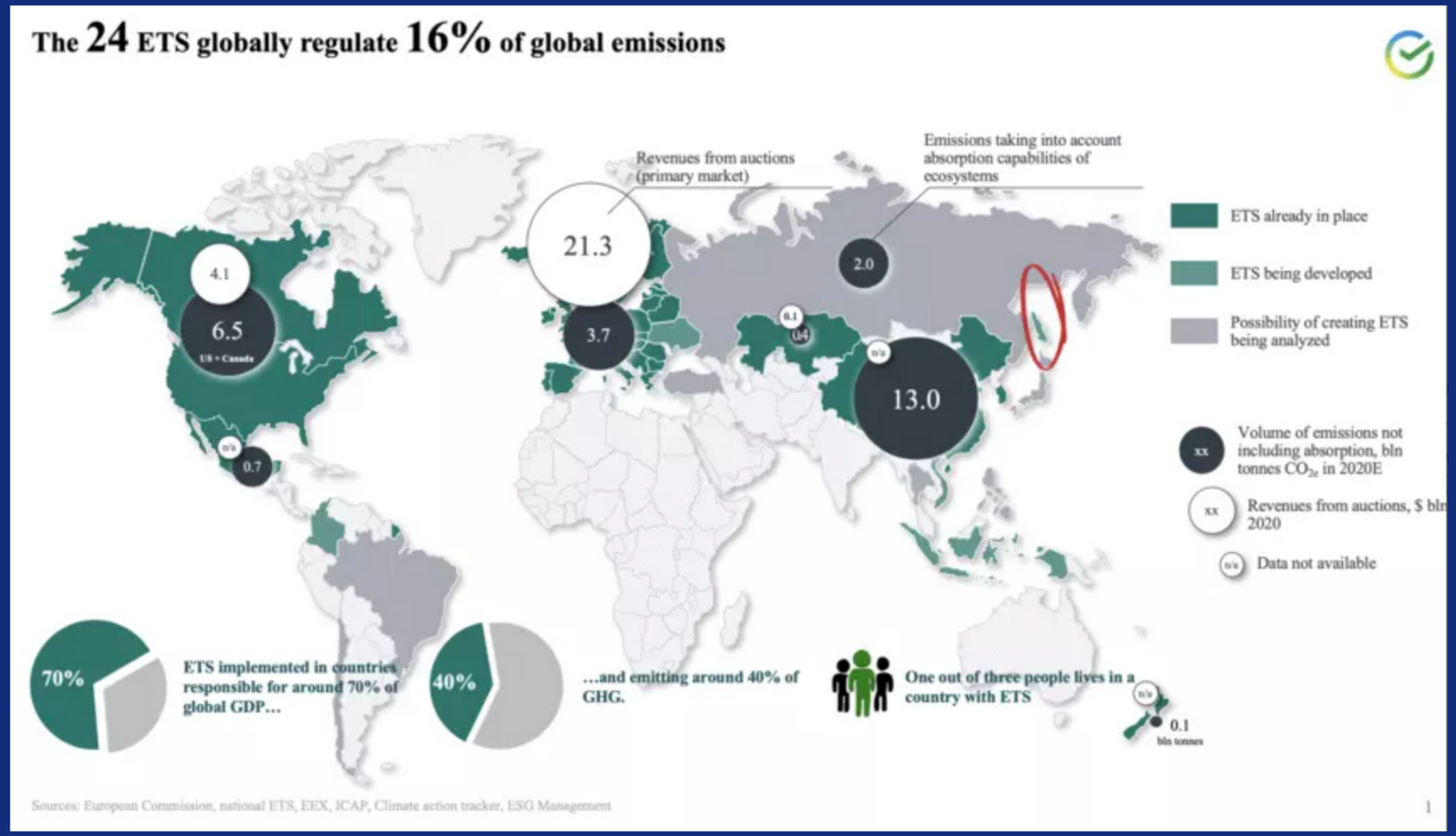


Demand and supply in the Carbon Market

- The quantity of the whole market supply of credits is fixed by institutions
- The quantity of demand can be differ by each firms depending techs, energy prices, climate factors and other economic fundamentals
- The Carbon Market allows the agents trade carbon credit at the price determined by interaction of the market agents

https://study.com/academy/answer/the-graph-below-shows-the-market-for-carbon-credits-consider-the-demand-for-pollution-rights-in-the-absence-of-constraints-on-pollution-rights-the-cost-of-polluting-is-zero-and-the-firms-in-annex-i.html

Background: Cap and Trade System



https://www.weforum.org/agenda/2021/07/how-to-build-a-eurasian-emissions-trading-system/

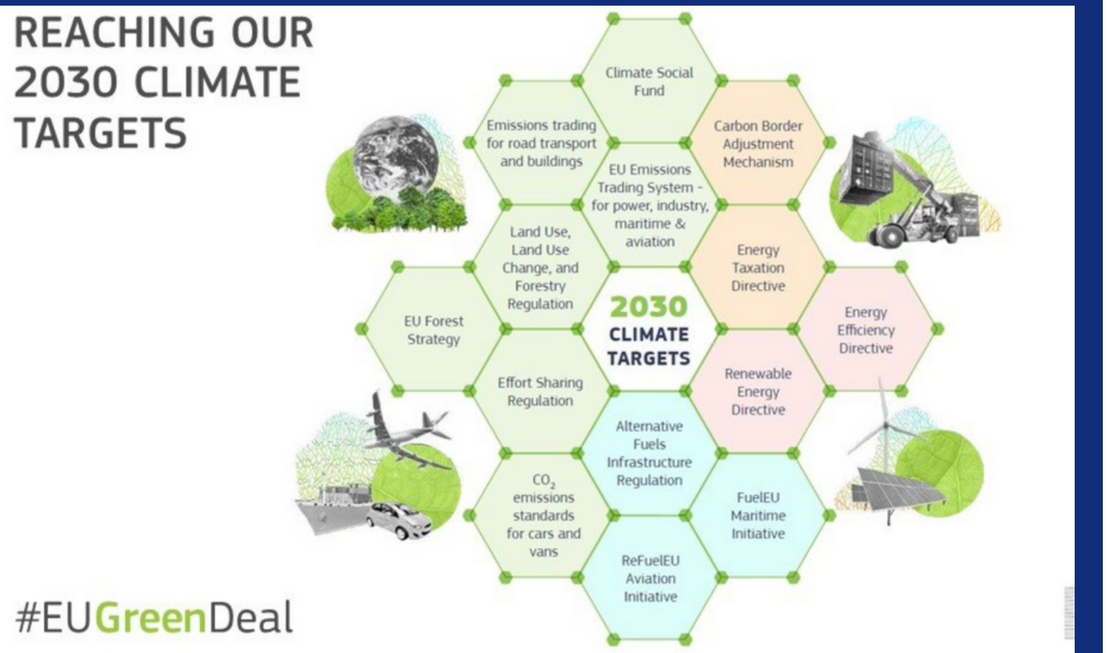
- The 24 emission trade systems(ETSs) are in operation globally: they are all an effort to regulate air pollution with economic instrument and have subtle distinction reflecting their own industrial properties
- Korea has also adopted ETS in 2015

Background:

EU net-zero greenhouse gas emission policies

Green Deal(19.11) ▶ Fit for 55(21.7)

https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52019DC0640&from=ET



Pricing	Targets	Rules
• Stronger Emissions Trading System including in aviation • Extending Emissions Trading to maritime, road transport, and buildings • Updated Energy taxation Directive • New Carbon Border Adjustment Mechanism	• Updated Effort Sharing Regulation • Updated Land Use Land Use Change and Forestry Regulation • Updated Renewable Energy Directive • Updated Energy Efficiency Directive	• Stricter CO_2 performance for cars & vans • New infrastructure for alternative fuels • ReFuelEU: More sustainable aviation fuels • FuelEU: Cleaner maritime fuels

Support measures

- Using revenues and regulations to promote innovation, build solidarity and mitigate impacts for the vulnerable, notably through the new **Social Climate Fund and enhanced Modernisation and Innovation Funds.**

Source: European Commission

Background:
EU net-zero greenhouse gas emission policies

Recent spur to Carbon neutral in Europe

- European Green Deal(2019. 11):
  - A comprehensive plan to transform economic activities to more sustainable way

- Fit for 55(2021.7)
  - A plan for carbon reduction in EU
  - Announced to reduce the net greenhouse gas emissions by at least 55% by 2030
  - Referred Carbon Boarder Adjusting Mechanism(CBAM) and articulated **that the free allowance will be gradually phased out as from 2026**

Background:

EU net-zero greenhouse gas emission policies

Carbon credit becomes a good traded in the global market!

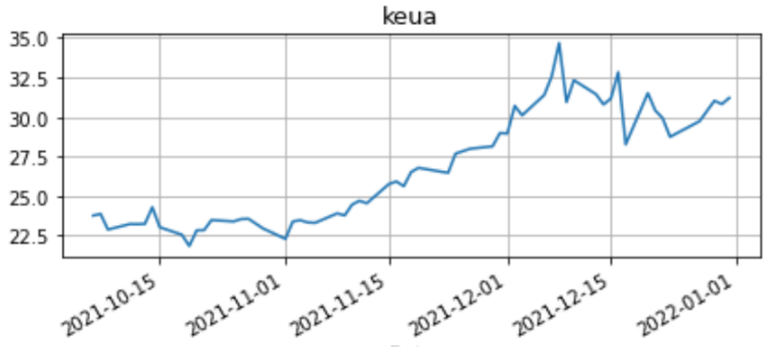
- Once a tool that incentivized market agent to reduce environmental pollution,
- now became a financial products in global market, fruiting the efforts to internalize the by-products of economic prosperity

Carbon Credit ETFs in the U.S Stock Market

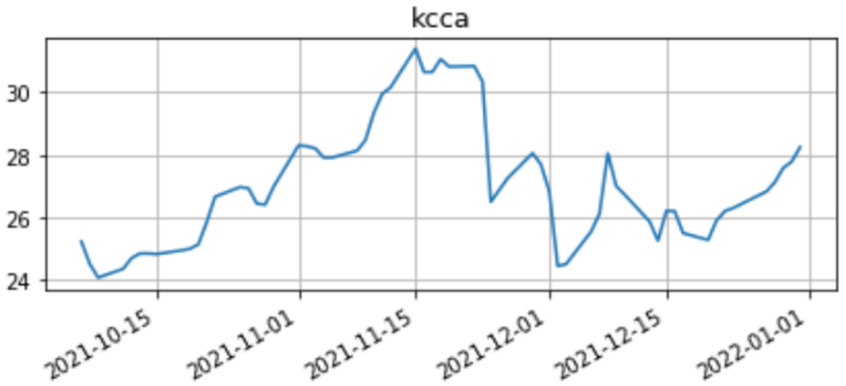
Carbon Credit in U.S Financial Market

- **KRBN(inception date: 2020.7.30)**
  - **Provides exposure to the EU ETS carbon credits, California's CCA carbon credits, and the RGGI carbon credits of the northeastern United States. Though current portfolio weighting heavily favours European Union Allowances**

- **KEUA(inception date: 2021.10.5)**
  - **Direct exposure to the European Union Allowances that trade under the EU's Emissions Trading Scheme. As a result, this ETF will closely follow the price performance of EU ETS carbon credits**



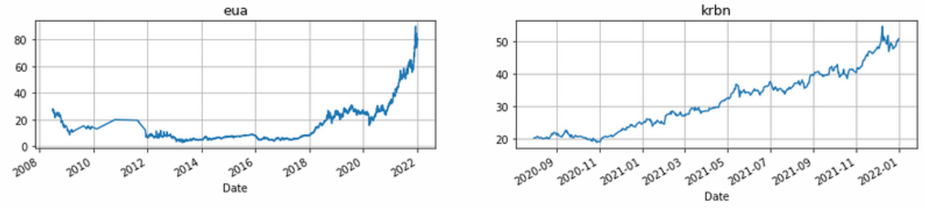
- **KCC(inception date: 201.10.5)**
  - **Provides direct exposure to the California Carbon Allowances that trade under California's cap-and-trade program. As a result, this ETF will closely follow the price performance of California's CCA carbon credits, providing good exposure to the growth of the carbon markets, though with greater risk and volatility**

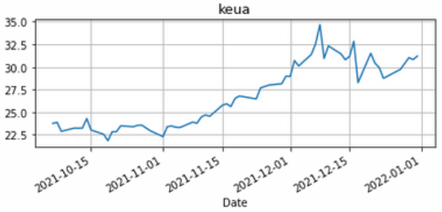


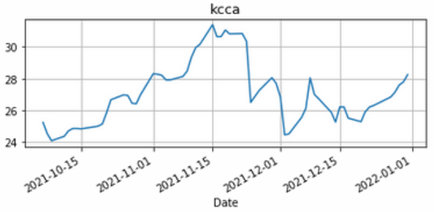
Carbon Credit in United States Financial Market

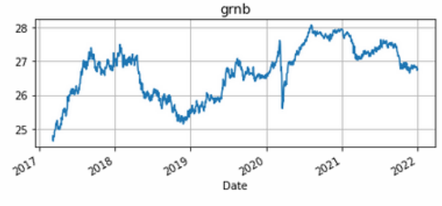
Compared to other indirect carbon related product: the price patterns

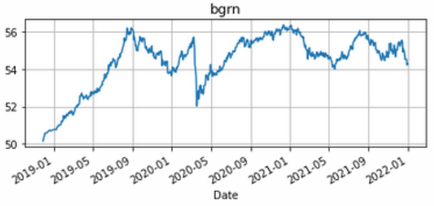
- The price patterns of Carbon related ETFs
- ETFs listed in https://carboncredits.com/how-to-invest-in-carbon-credits-carbon-etfs-and-carbon-stocks/

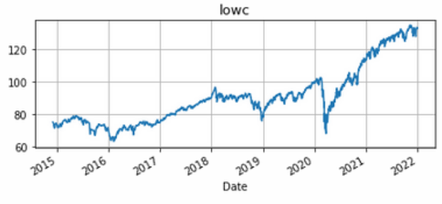


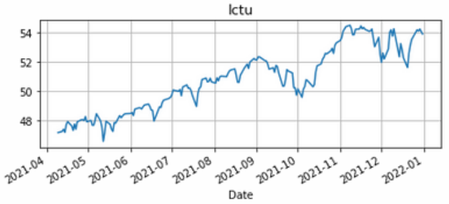






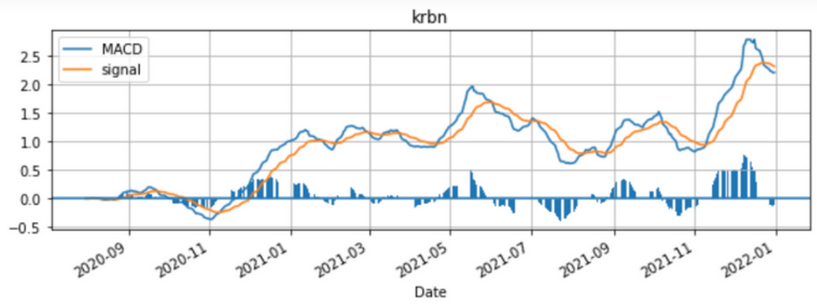


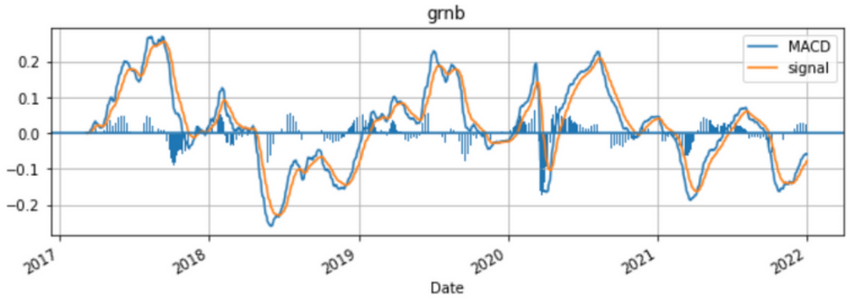


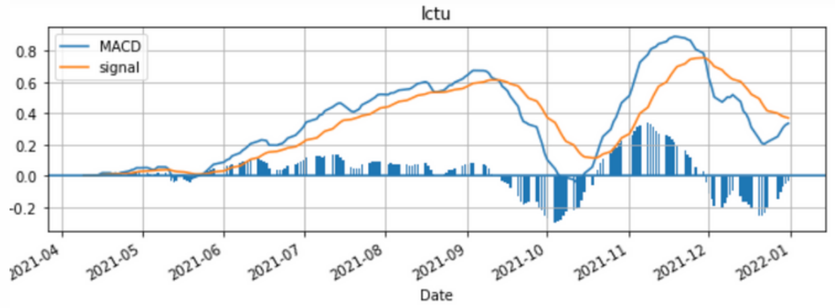


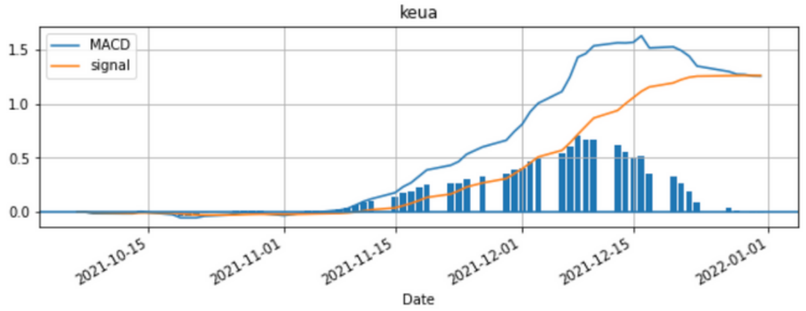
Carbon Credit in United States Financial Market

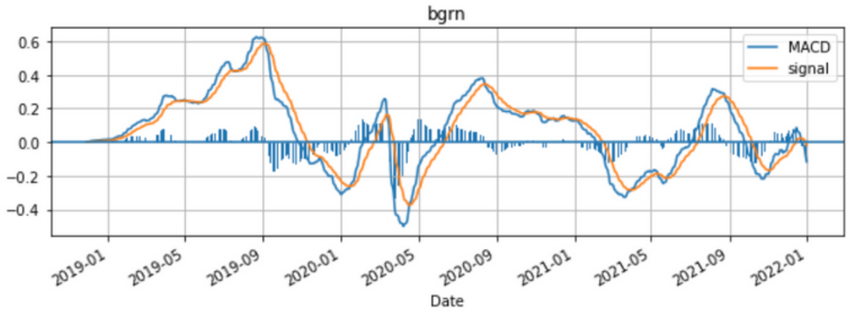
Compared to other indirect carbon related product: MACD & signal

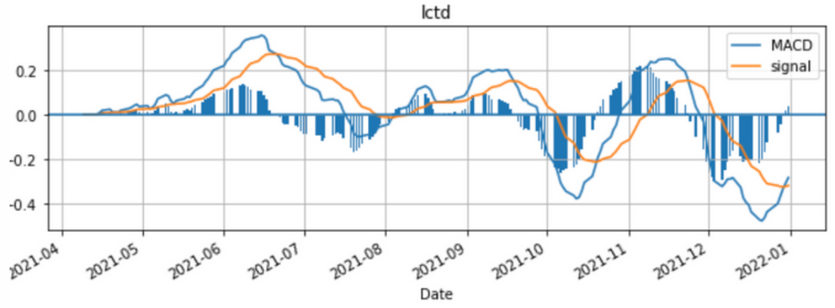


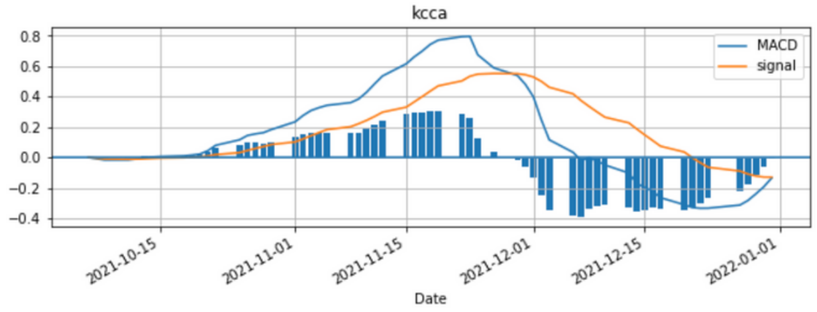


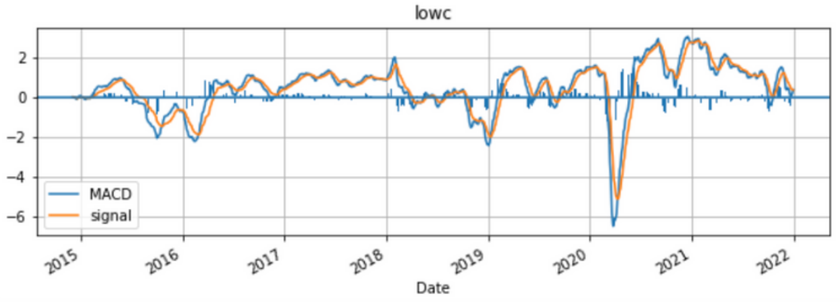


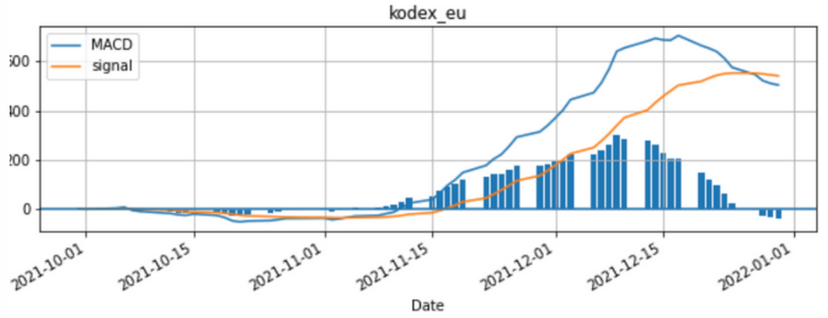






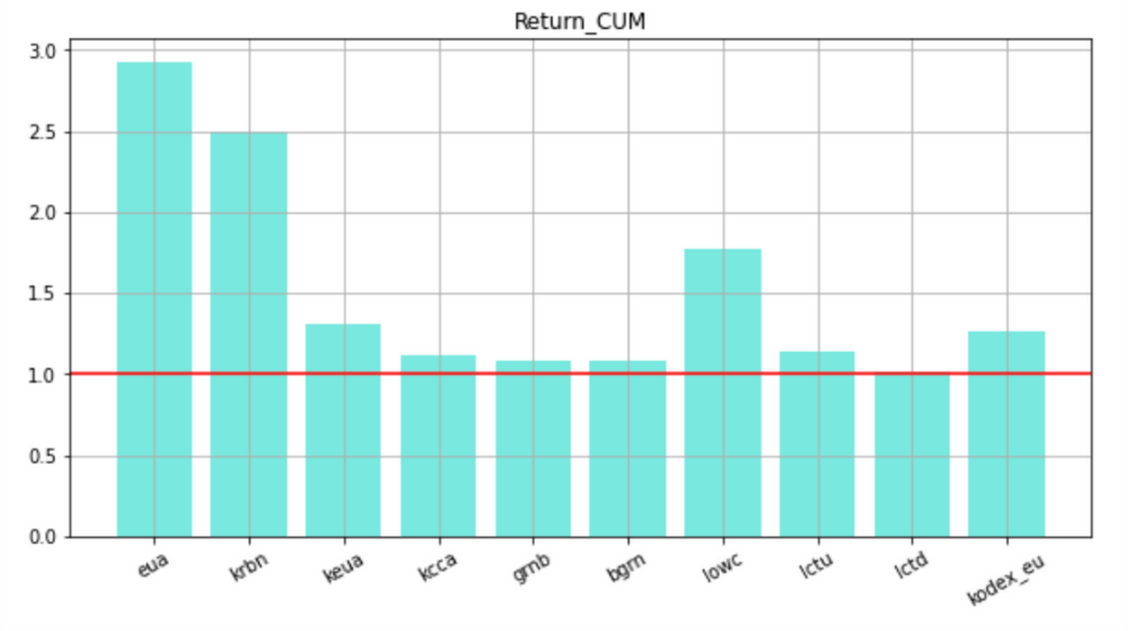




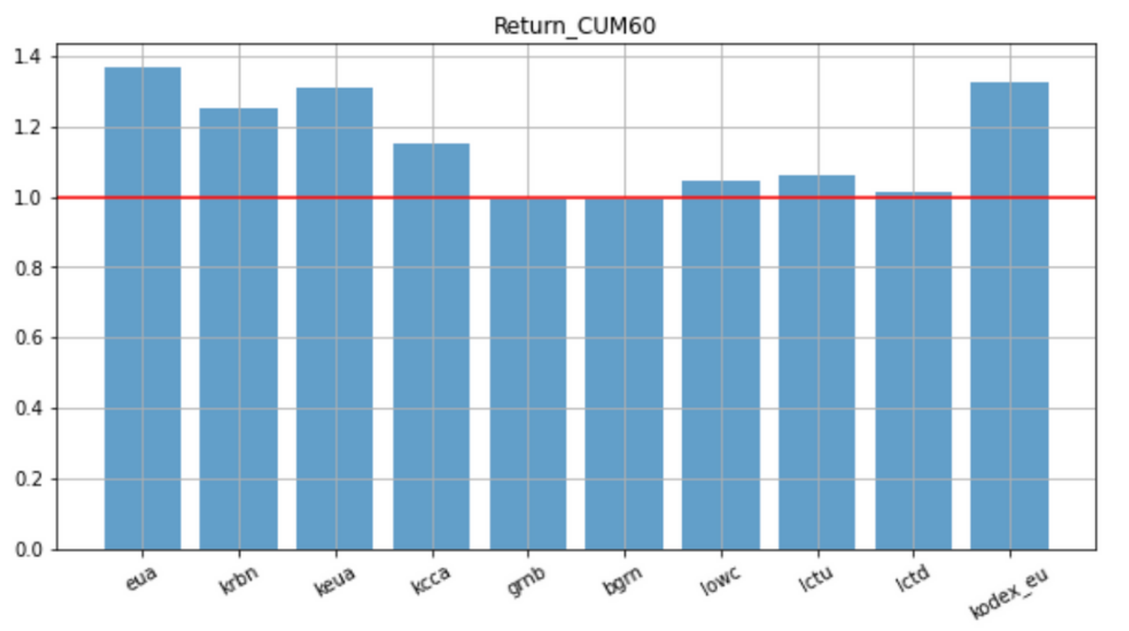


Carbon Credit in United States Financial Market

Compared to other indirect carbon related product: cumulative return rate



as from inception date

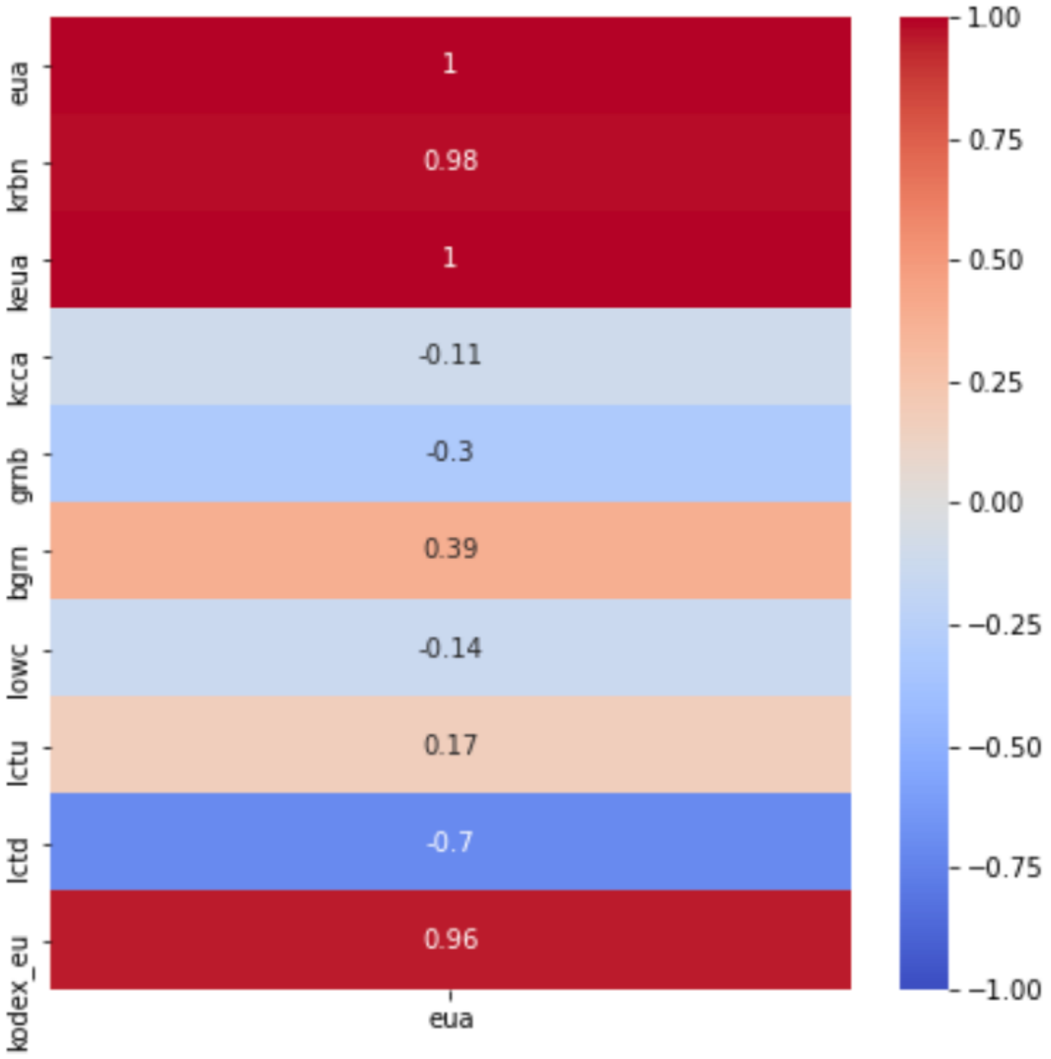


For recent 60 days

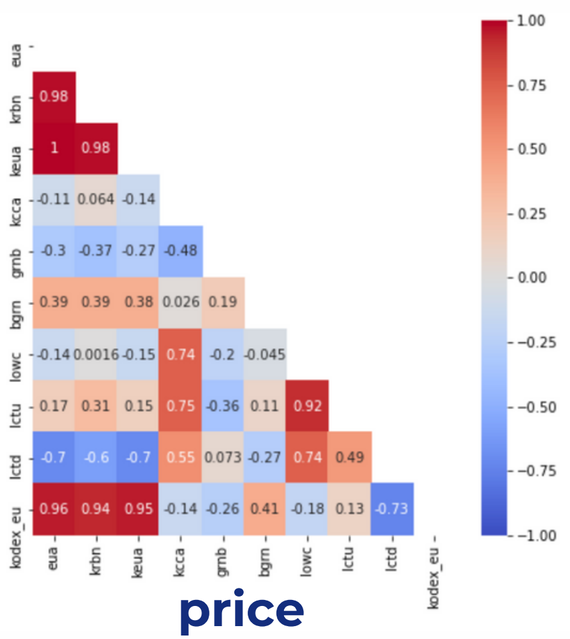
Compared to other indirect carbon related product, the products exposures to direct carbon credits(KRBN, KEUA, KCCA, KODEX_EU) show a good earning rate

Carbon Credit in United States Financial Market

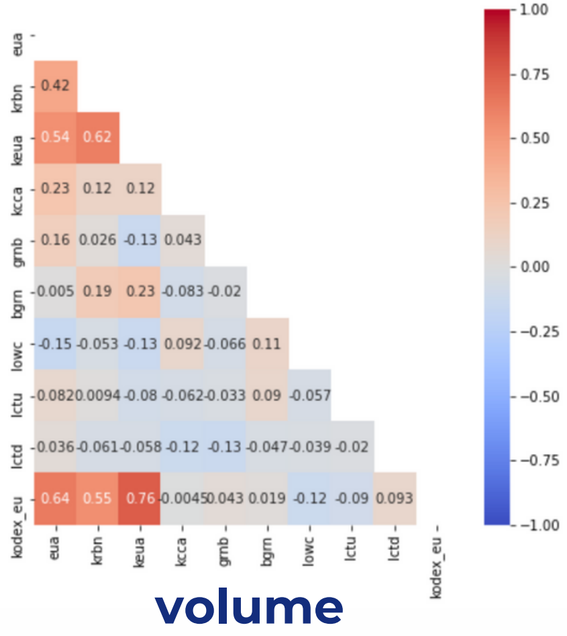
Compared to other indirect carbon related product: Correlation



correlation of EUA price with other products



price



volume

- **Price of products exposures to direct carbon credits(KRBN, KEUA, KCCA, KODEX_EU) are highly correlated**
- **Volume shows a bit lower correlation**

Explorative Data Analysis (EDA)

WHAT DETERMINES CARBON CREDIT PRICE?

- ENERGY PRICE

  A carbon compounds in the air basically generated during the incineration of fossil fuel

- CLIMATE

  Climate determines the energy demand for heating

- ECONOMIC ELEMENTS

  Industrial and household activity affect the carbon emission

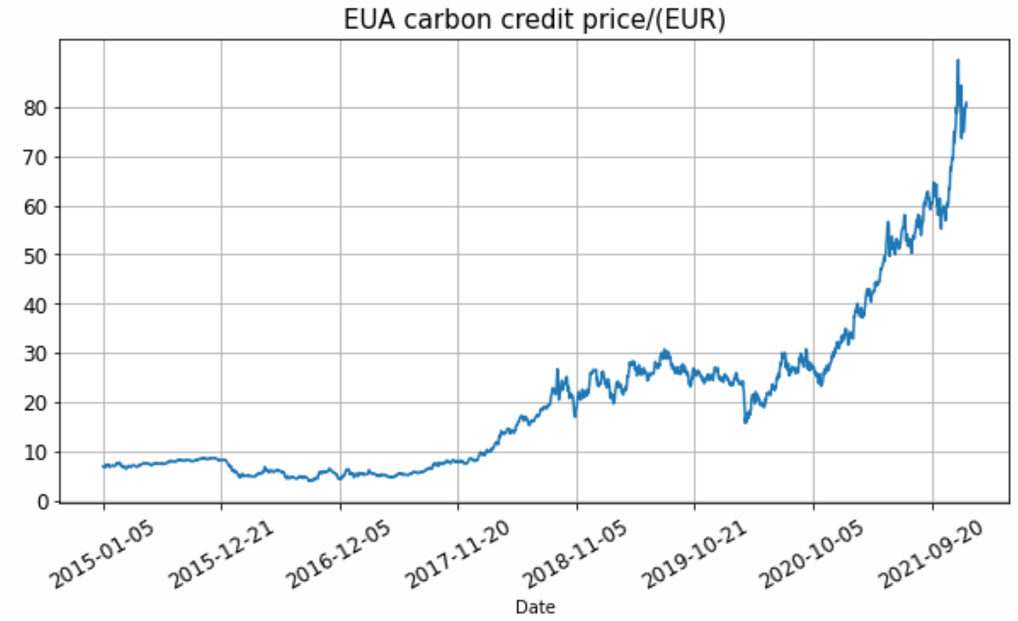
the features in this project

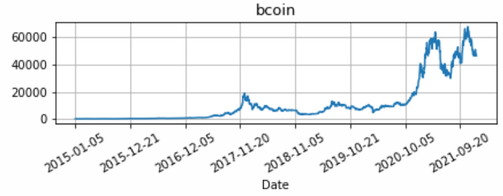
1. commodities : energy, grain, metals etc.

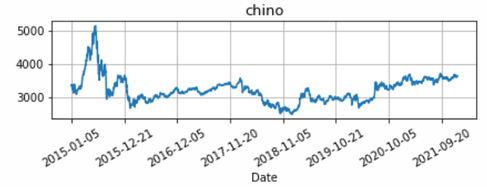
2. stock market indexes of related countries

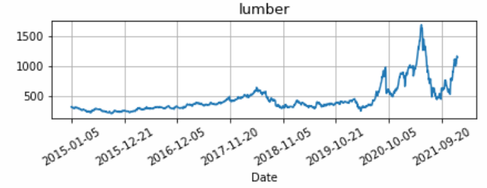
3. currency: euro and bitcoin

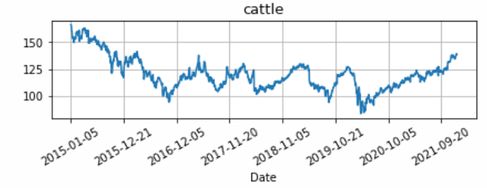
Explorative Data Analysis(EDA): Price fluctuation

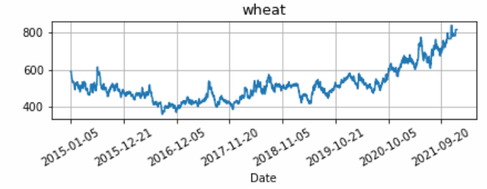


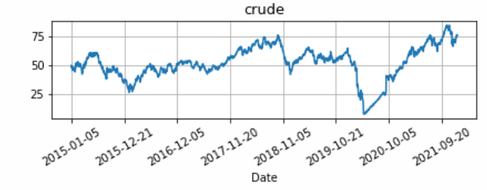


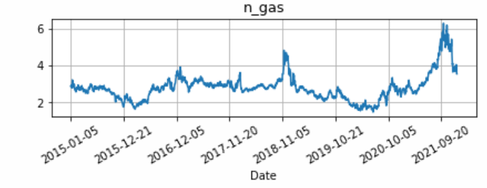




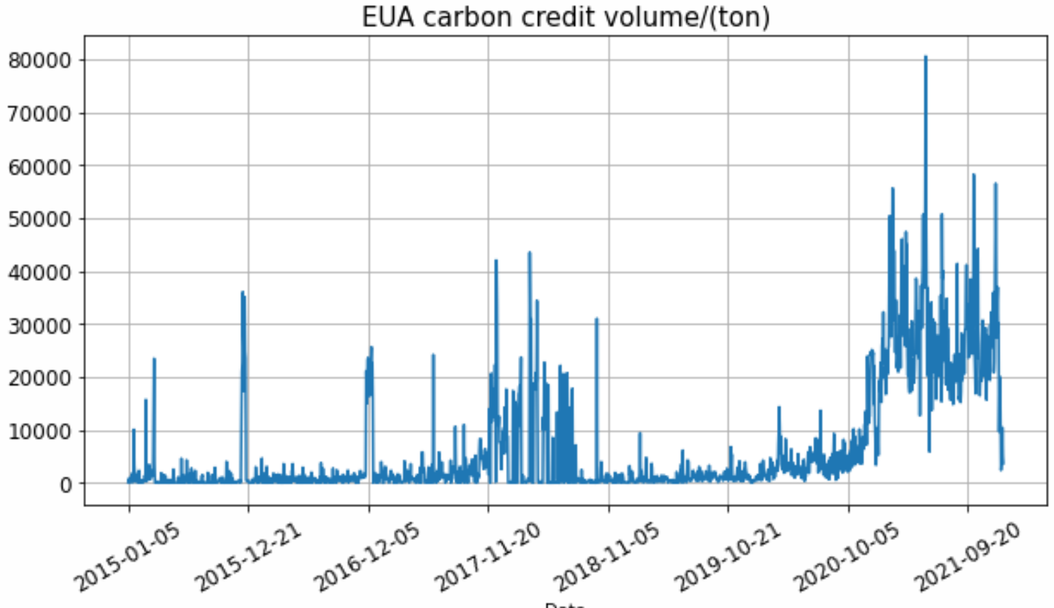


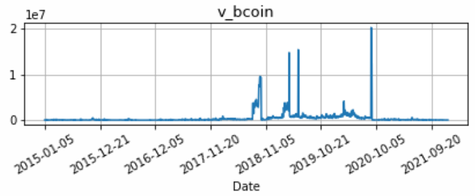


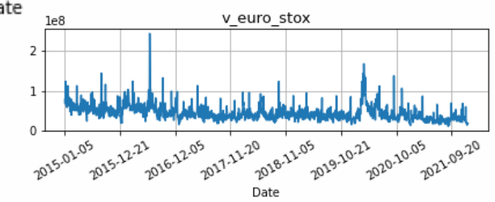


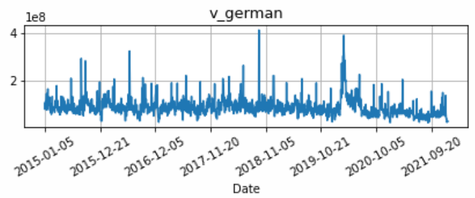


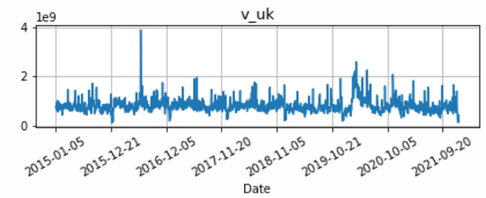
Explorative Data Analysis(EDA): Volume

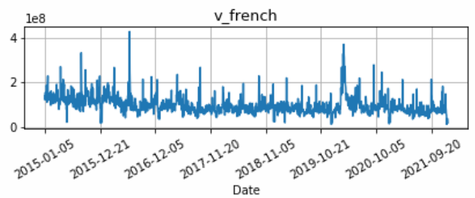


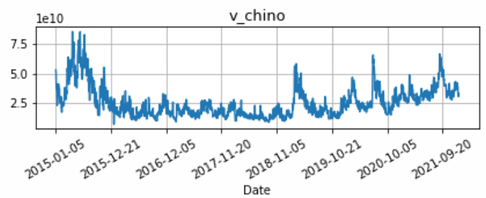


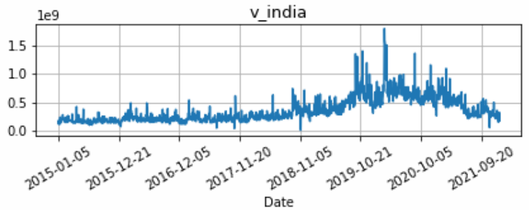


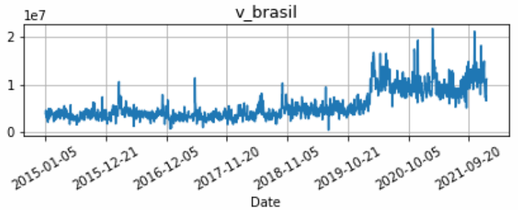


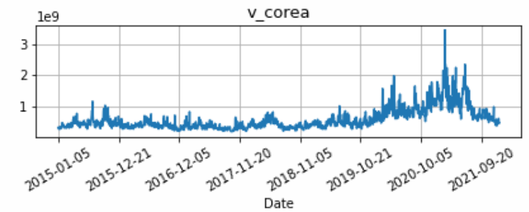


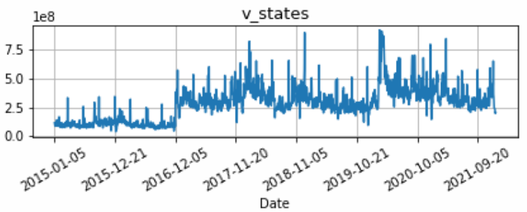


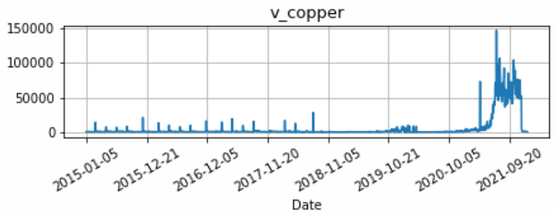


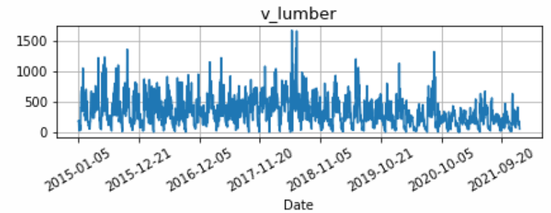


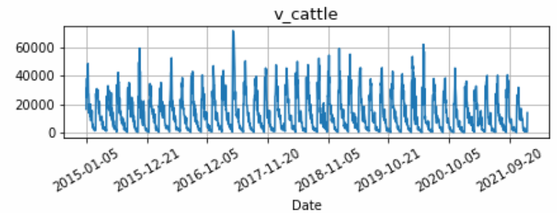


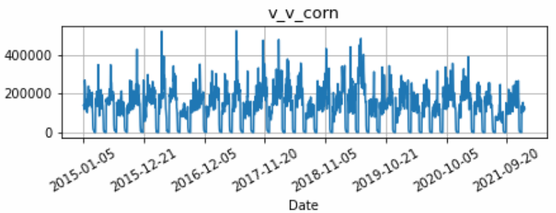


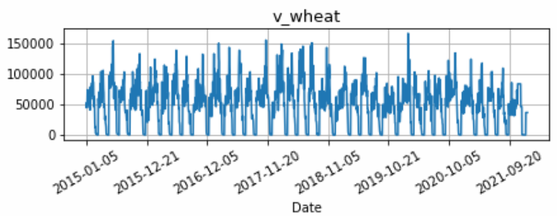


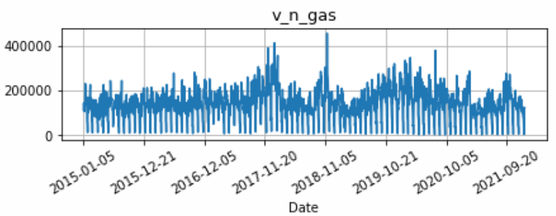




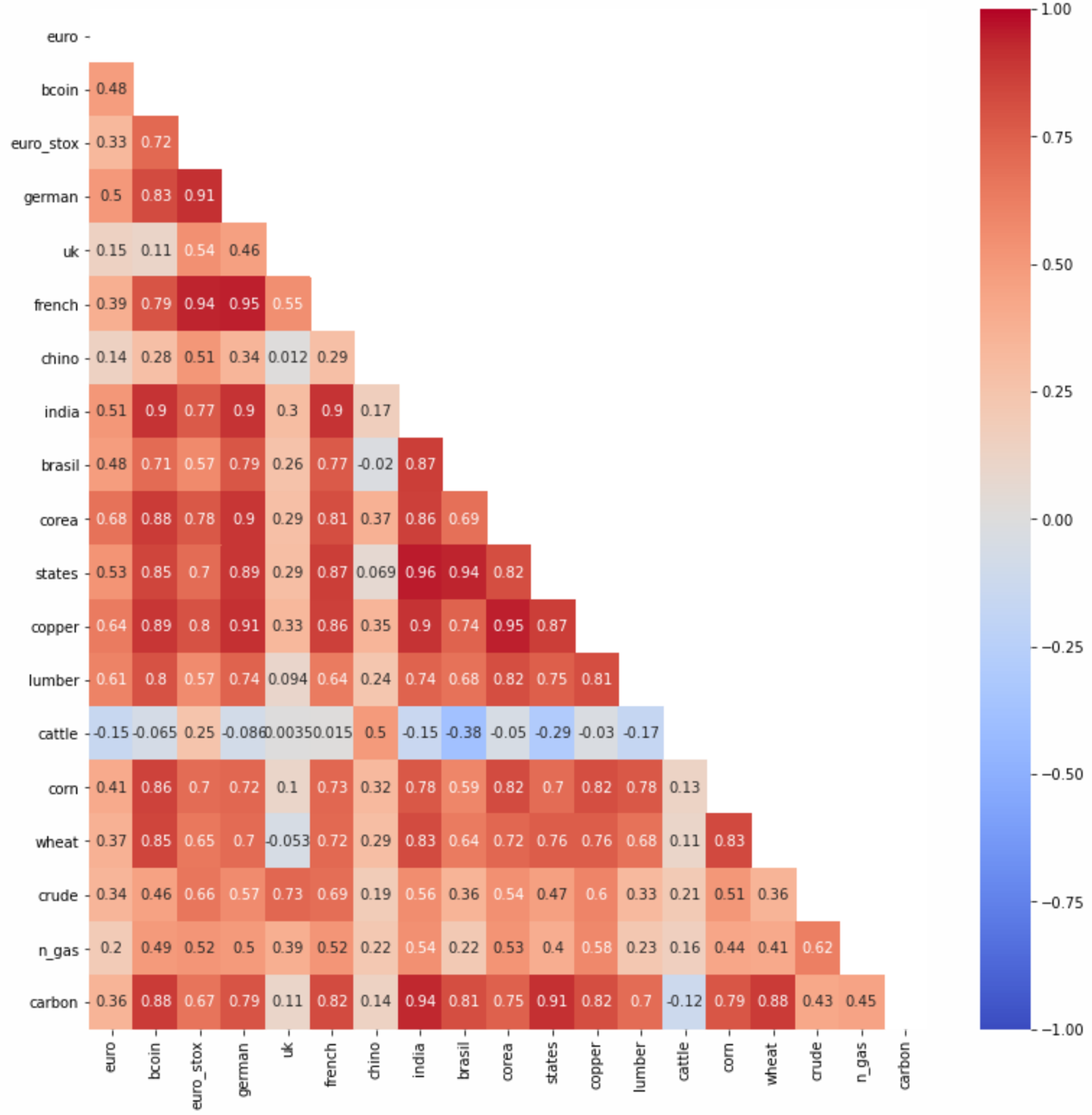


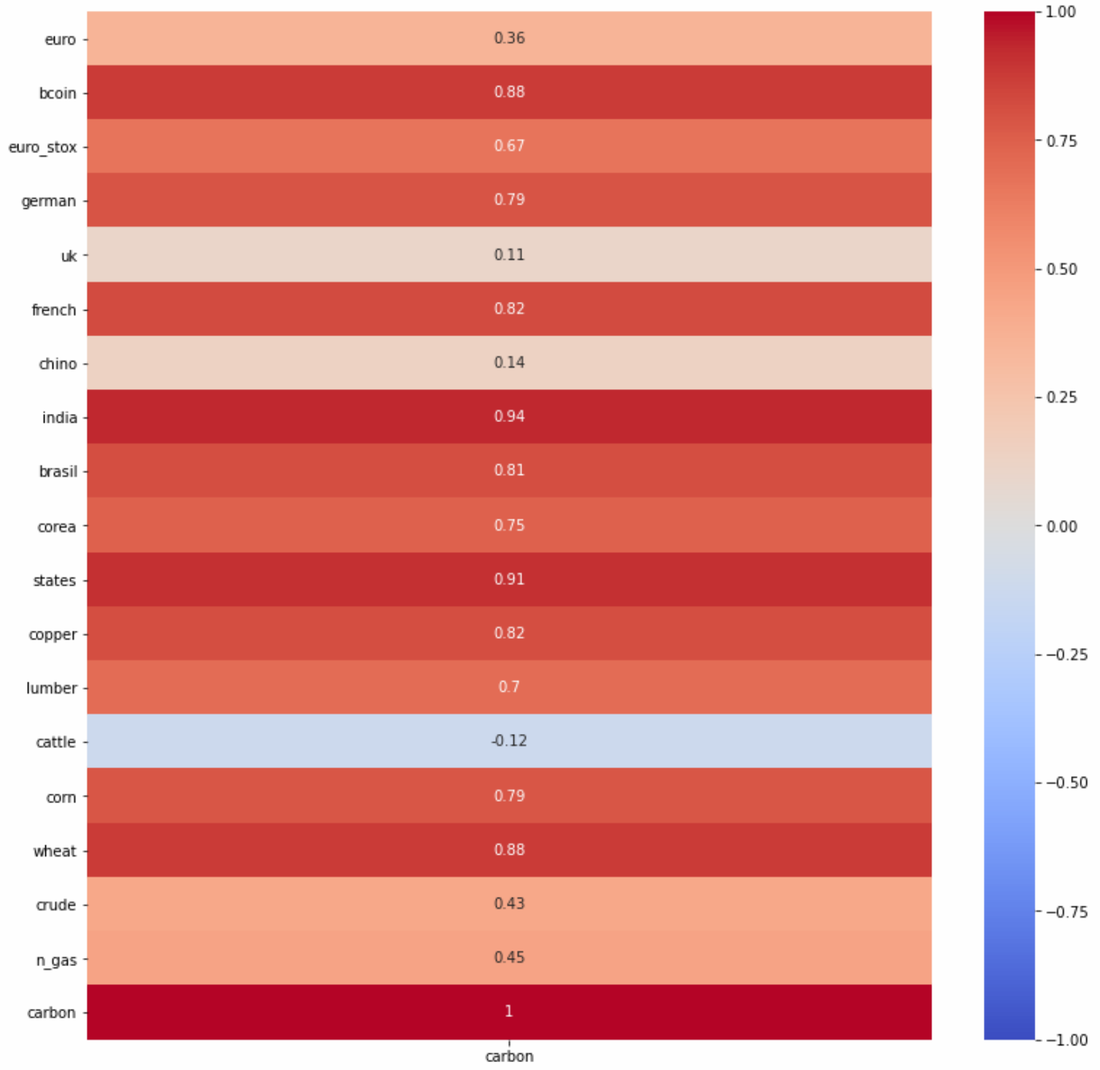






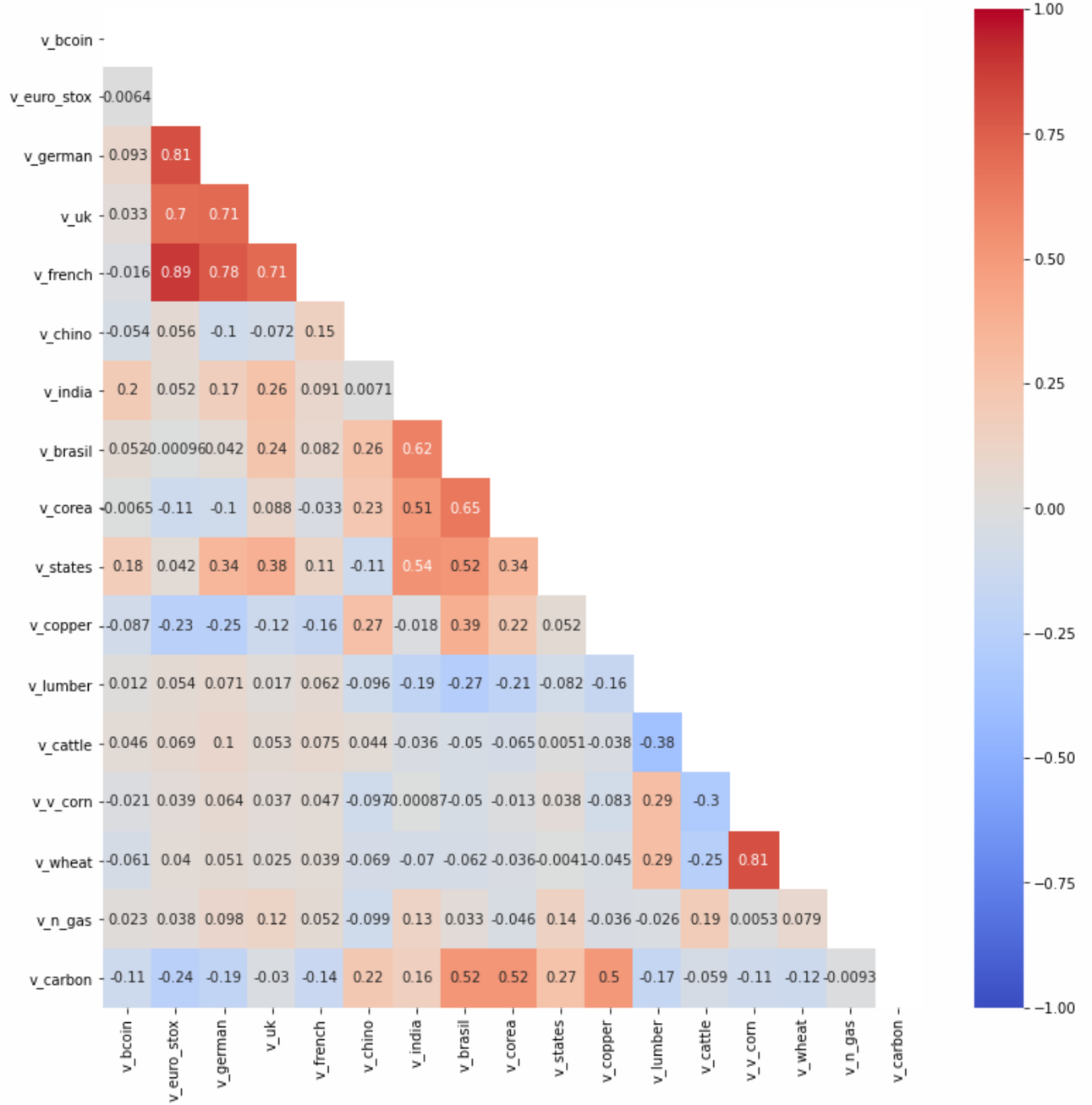
Explorative Data Analysis(EDA): Correlation of Price

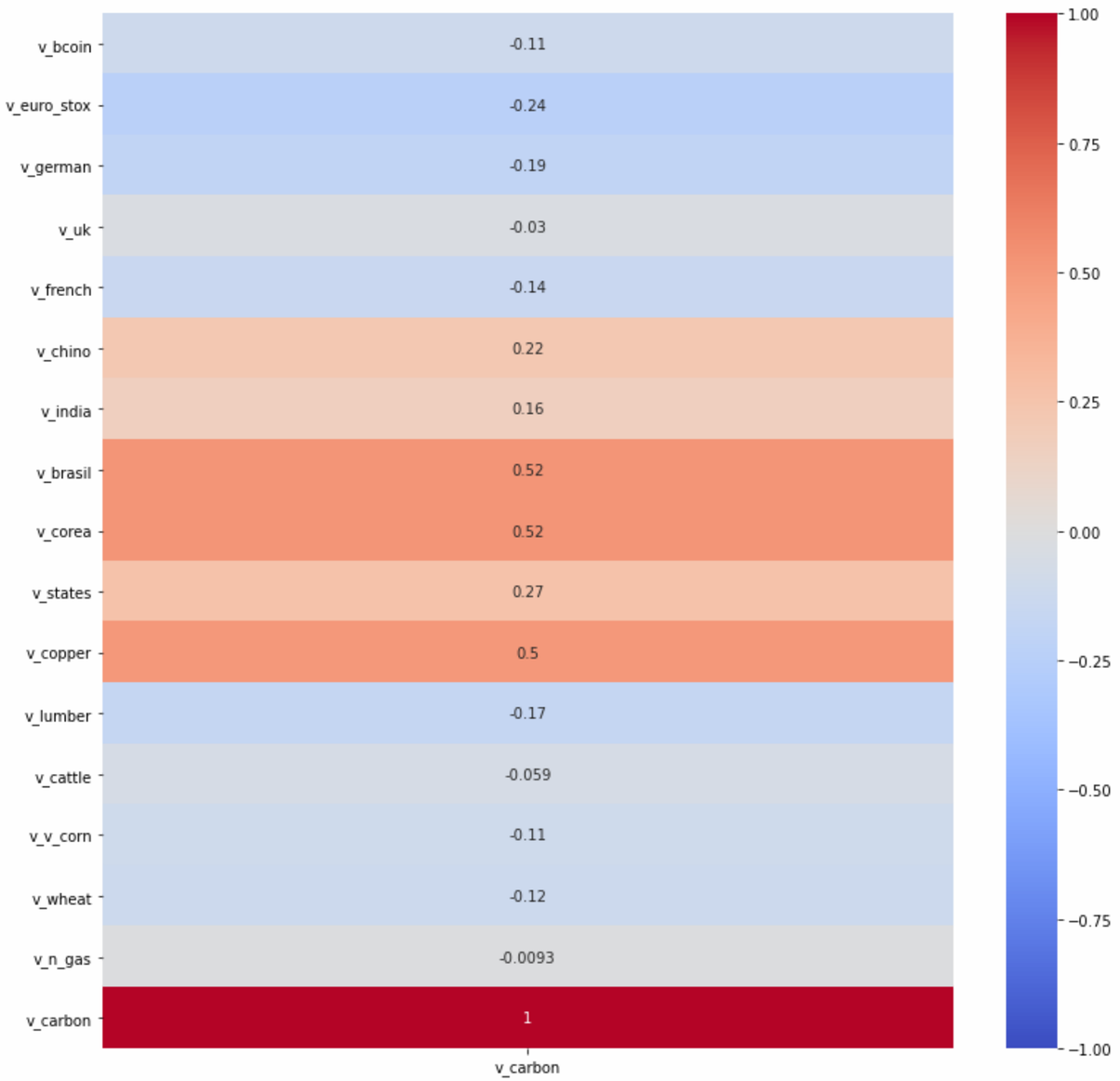




correlation of EUA price with other features

Explorative Data Analysis(EDA): Correlation of Volume





correlation of EUA volume with other features

Machine Learning Models

Model outline

- Features
  - close price and trade volume of 17 indexes
    - Currencies : 'euro', 'bitcoin'
    - Stock market indexes: 'German', 'UK', 'France', 'China', 'India', 'Brazil', 'Korea', 'United States', 'Euro Stoxx
    - Commodities: 'copper', 'lumber', 'cattle', 'corn', 'wheat', 'crude oil', 'natural gas'
    - imported using FinanceDataReader
    - date from 2015.1.4 to 2021.12.31

- Window size : 5 / 10 / 30 days

- Models
  - Linear Regression, Ridge, Lasso, XGBRegressor, LGBMRegressor, RandomForestRegressor

- Accuracy metrics: r2score, MSE

- Train_test_split: the latest 20% data is separated as test data

Basic Code for Machine learning

data split/scale

0.1 window size =10

```
window_size =10
x = []
y = []

for i in range(len(df) - window_size):
    x.append([df.iloc[i+j,:] for j in range(window_size)])
    y.append(df.iloc[window_size+i, 18])
```

```
X = np.asarray(x)
y = np.asarray(y)
print(X.shape, y.shape)
```

```
(1815, 10, 38) (1815,)
```

```
tts = int(X.shape[0]*0.8)

x_train = X[:tts]
x_test = X[tts:]

y_train =y[:tts]
y_test = y[tts:]

print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(1452, 10, 38) (363, 10, 38) (1452,) (363,)
```

```
X_train = np.reshape(x_train, (x_train.shape[0], -1))
X_test = np.reshape(x_test, (x_test.shape[0], -1))
print(X_train.shape, X_test.shape)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

x_train_scaled= scaler.transform(X_train)
x_test_scaled = scaler.transform(X_test)
```

```
(1452, 380) (363, 380)
```

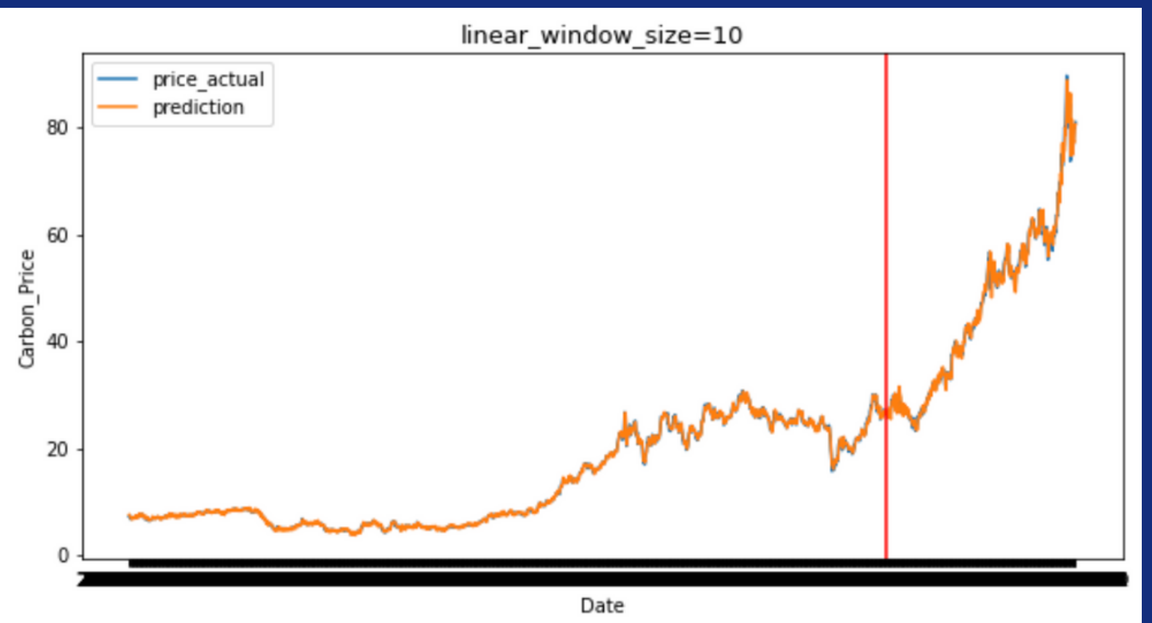
model fit/ visualization

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

lr = LinearRegression()
rid = Ridge()
las = Lasso()
xgb = XGBRegressor()
lgbm = LGBMRegressor()
rfr = RandomForestRegressor()
```

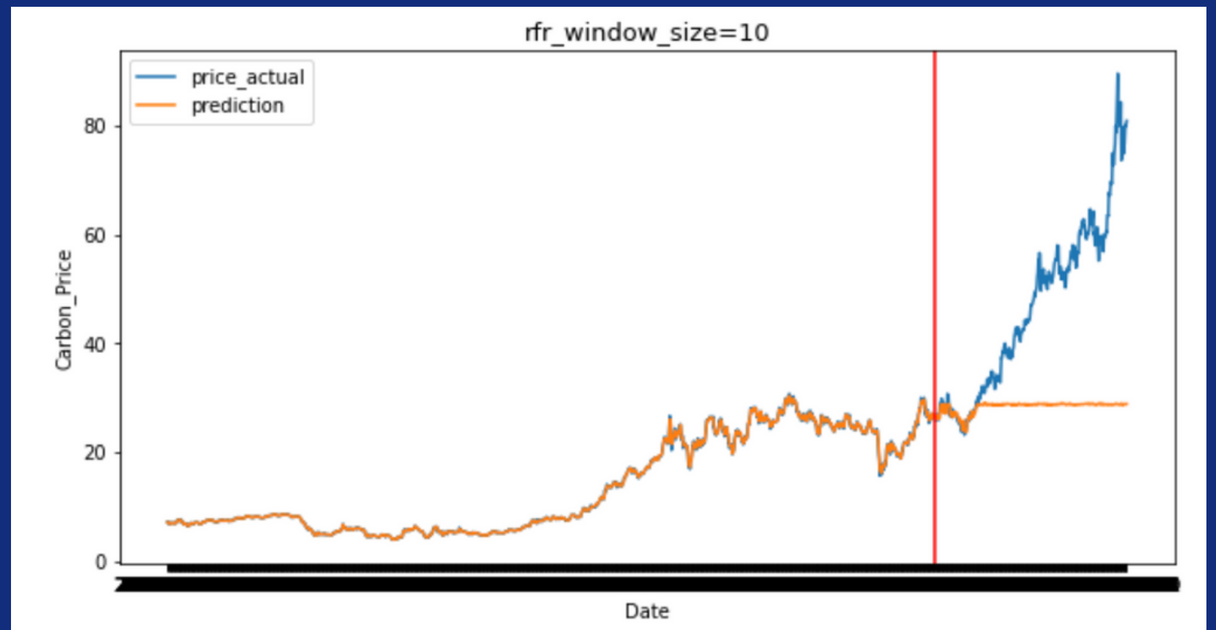
```
mds = [lr, rid,las,xgb,lgbm,rfr]
n_mds = ['linear', 'ridge','lasso','xgb','lgbm','rfr']
```

```
for i, m in enumerate(mds):
    m.fit(x_train_scaled, y_train)
    trpreds = m.predict(x_train_scaled)
    preds = m.predict(x_test_scaled)

    x = np.concatenate((trpreds, preds))
    x = pd.DataFrame(x)
    x.index = df.iloc[10:].index

    plt.figure(figsize=(10,5))
    plt.plot(df['carbon'].iloc[10:], label ='price_actual')
    plt.plot(x, label ='prediction')
    plt.axvline(df.index[x_train.shape[0]+10], color='red', ls ='-')
    plt.title('{}_window_size=10'.format(n_mds[i]), fontsize =13)
    plt.xlabel('Date')
    plt.ylabel('Carbon_Price')
    plt.legend()
    plt.show()


    print('----------------------')
    print('model:', n_mds[i])
    print('train_r2score:', np.round(r2_score(y_train, trpreds),2))
    print('test_r2score:', np.round(r2_score(y_test, preds),2))
    print('train_mse:', np.round(mean_squared_error(y_train, trpreds),2))
    print('test_mse:', np.round(mean_squared_error(y_test, preds),2))
```

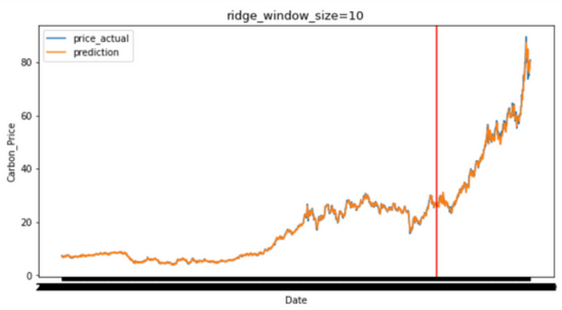
Results: window size=10



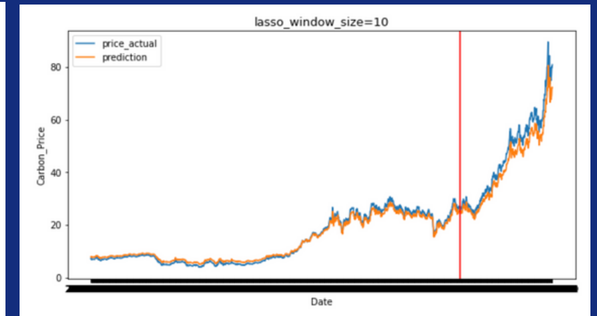

```
----------------------
model: linear
train_r2score: 1.0
test_r2score: 0.99
train_mse: 0.14
test_mse: 2.34
```



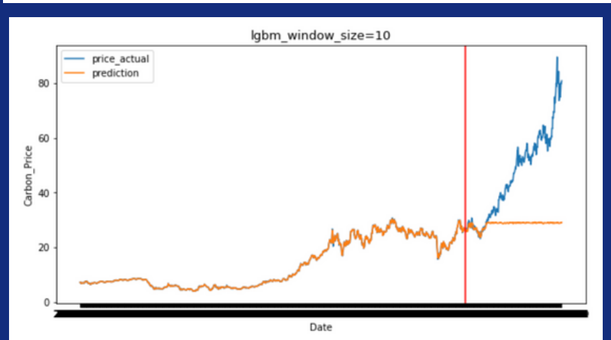

```
----------------------
model: rfr
train_r2score: 1.0
test_r2score: -1.21
train_mse: 0.03
test_mse: 555.9
```



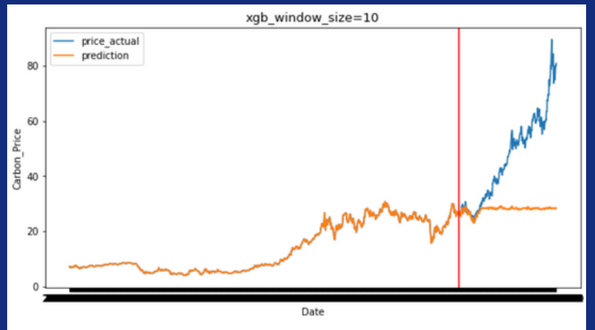

```
----------------------
model: ridge
train_r2score: 1.0
test_r2score: 0.99
train_mse: 0.15
test_mse: 2.26
```




```
----------------------
model: lasso
train_r2score: 0.98
test_r2score: 0.92
train_mse: 1.21
test_mse: 21.17
```

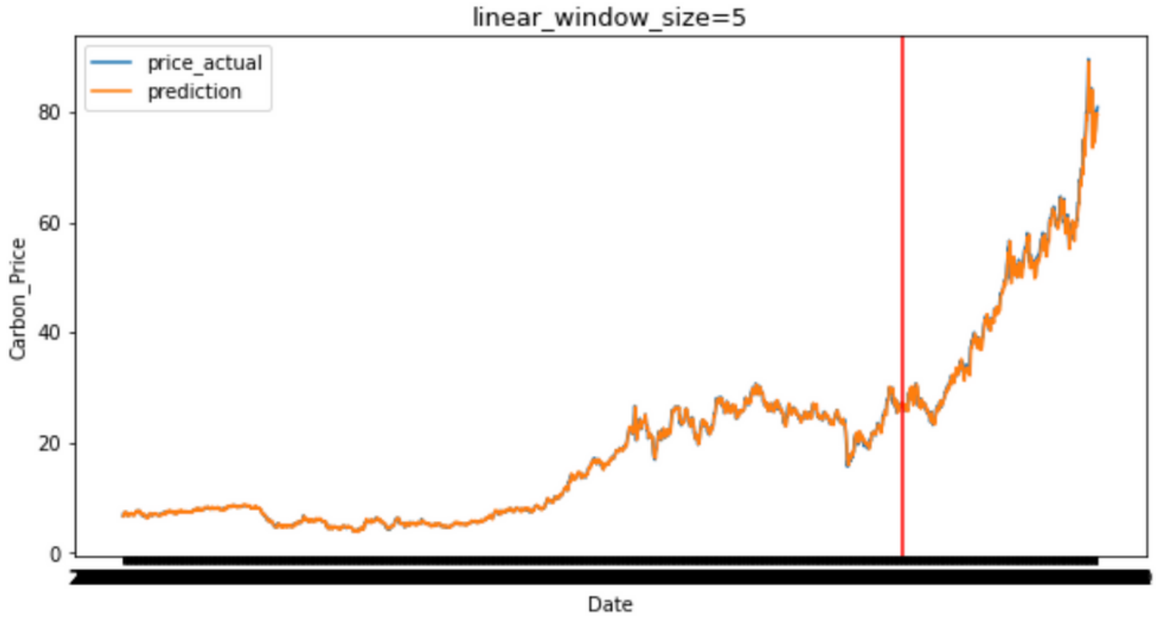



```
----------------------
model: lgbm
train_r2score: 1.0
test_r2score: -1.19
train_mse: 0.01
test_mse: 550.83
```

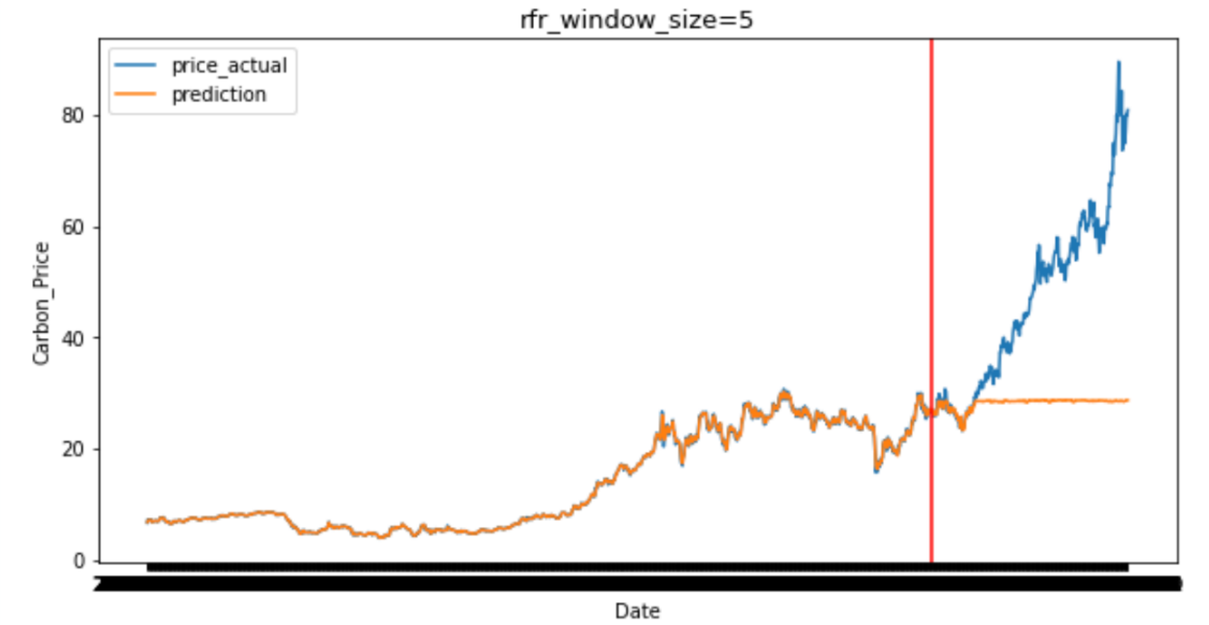



```
----------------------
model: xgb
train_r2score: 1.0
test_r2score: -1.29
train_mse: 0.0
test_mse: 575.29
```

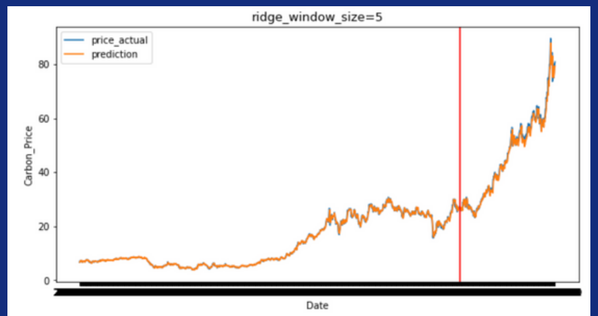
Results: window size=5



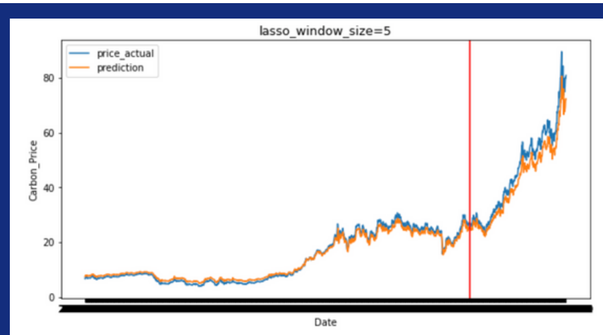

```
----------------------
model: linear
train_r2score: 1.0
test_r2score: 0.99
train_mse: 0.17
test_mse: 2.27
```



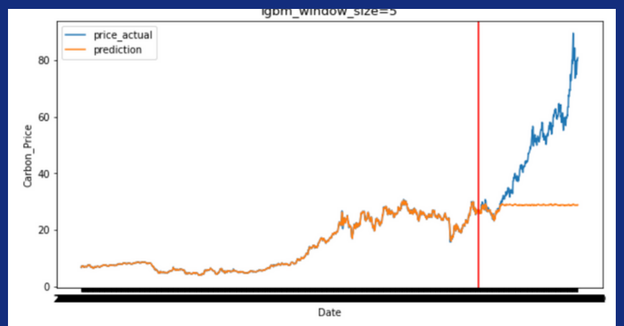

```
----------------------
model: rfr
train_r2score: 1.0
test_r2score: -1.23
train_mse: 0.03
test_mse: 561.83
```



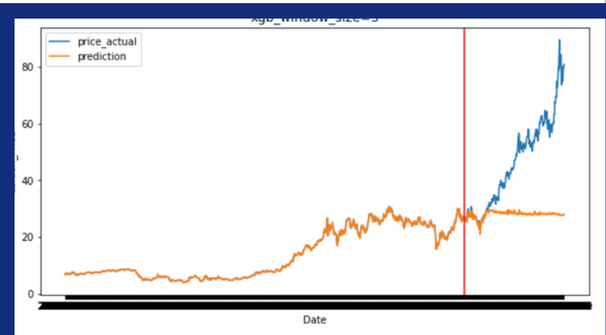

```
----------------------
model: ridge
train_r2score: 1.0
test_r2score: 0.99
train_mse: 0.18
test_mse: 2.4
```




```
----------------------
model: lasso
train_r2score: 0.98
test_r2score: 0.92
train_mse: 1.21
test_mse: 21.17
```

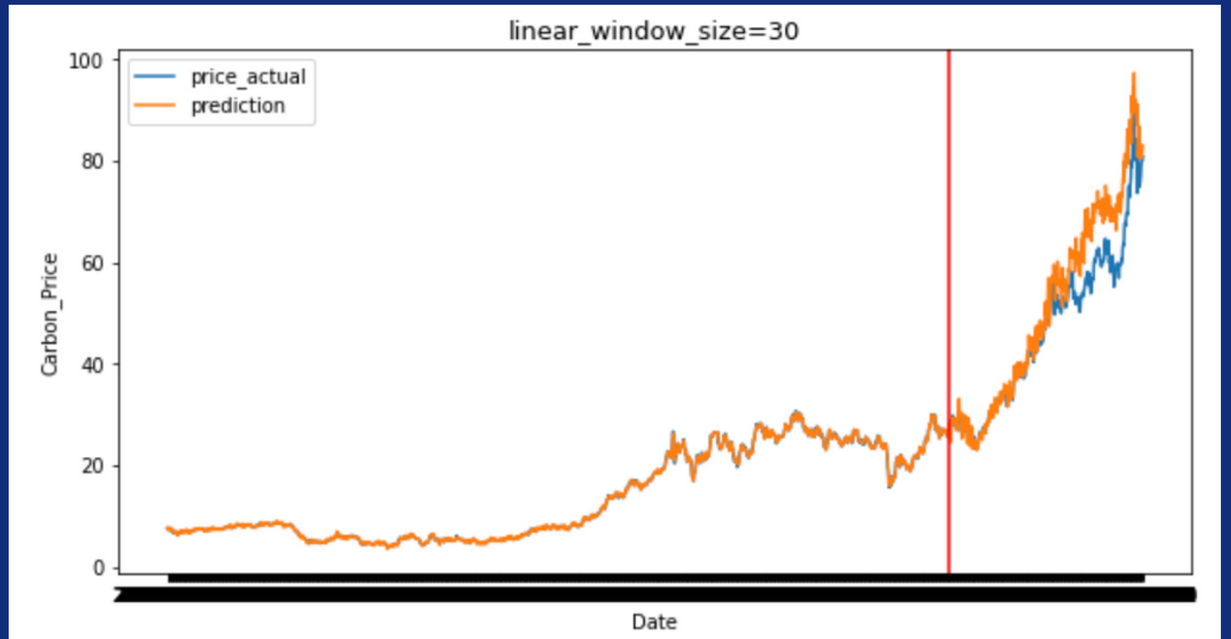



```
----------------------
model: lgbm
train_r2score: 1.0
test_r2score: -1.2
train_mse: 0.01
test_mse: 554.3
```

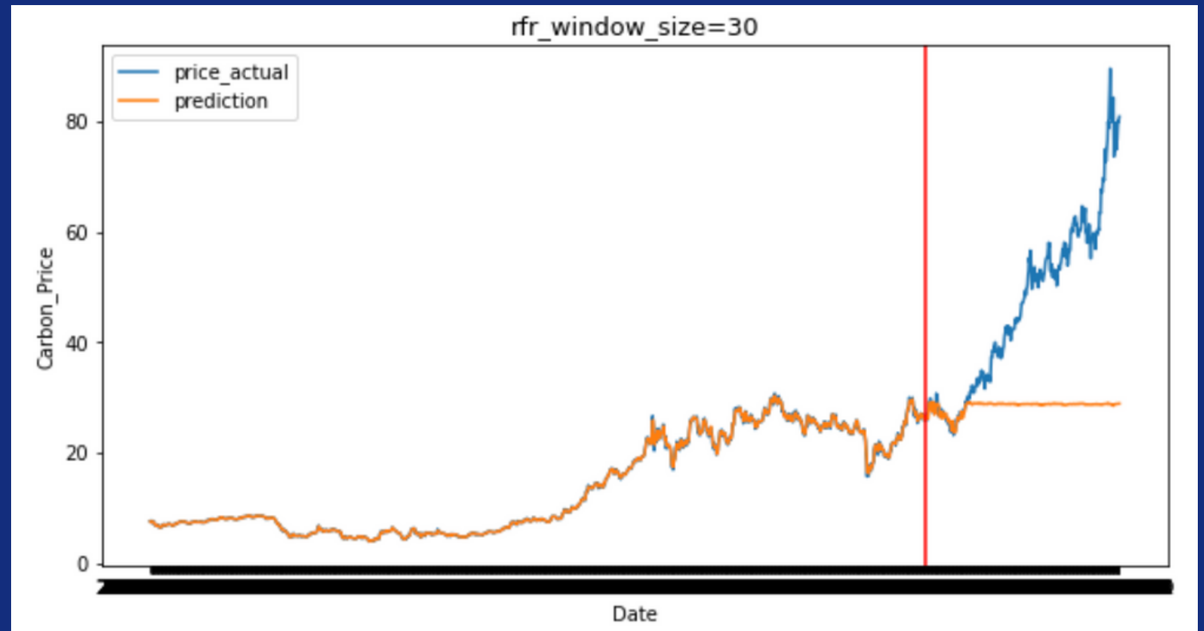



```
----------------------
odel: xgb
ain_r2score: 1.0
st_r2score: -1.29
ain_mse: 0.0
st_mse: 577.15
```

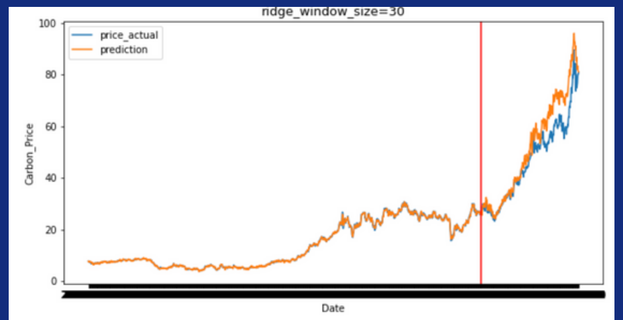
Results: window size=30



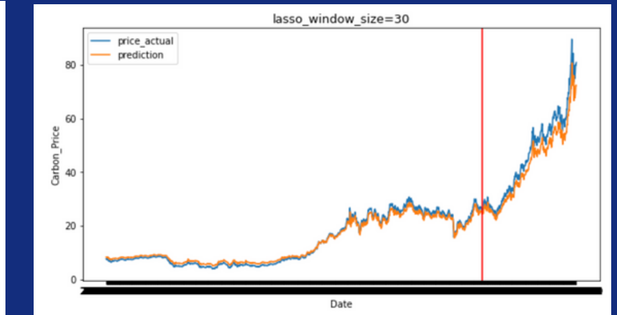

```
----------------------
model: linear
train_r2score: 1.0
test_r2score: 0.83
train_mse: 0.04
test_mse: 42.51
```



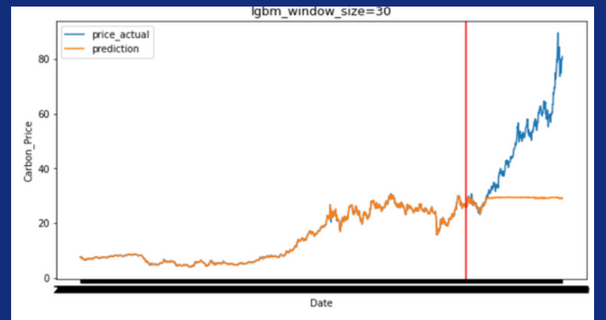

```
----------------------
model: rfr
train_r2score: 1.0
test_r2score: -1.25
train_mse: 0.03
test_mse: 562.18
```



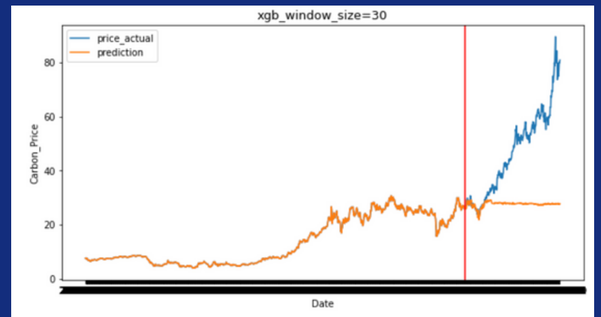

```
----------------------
model: ridge
train_r2score: 1.0
test_r2score: 0.81
train_mse: 0.06
test_mse: 47.7
```




```
----------------------
model: lasso
train_r2score: 0.98
test_r2score: 0.92
train_mse: 1.21
test_mse: 21.1
```

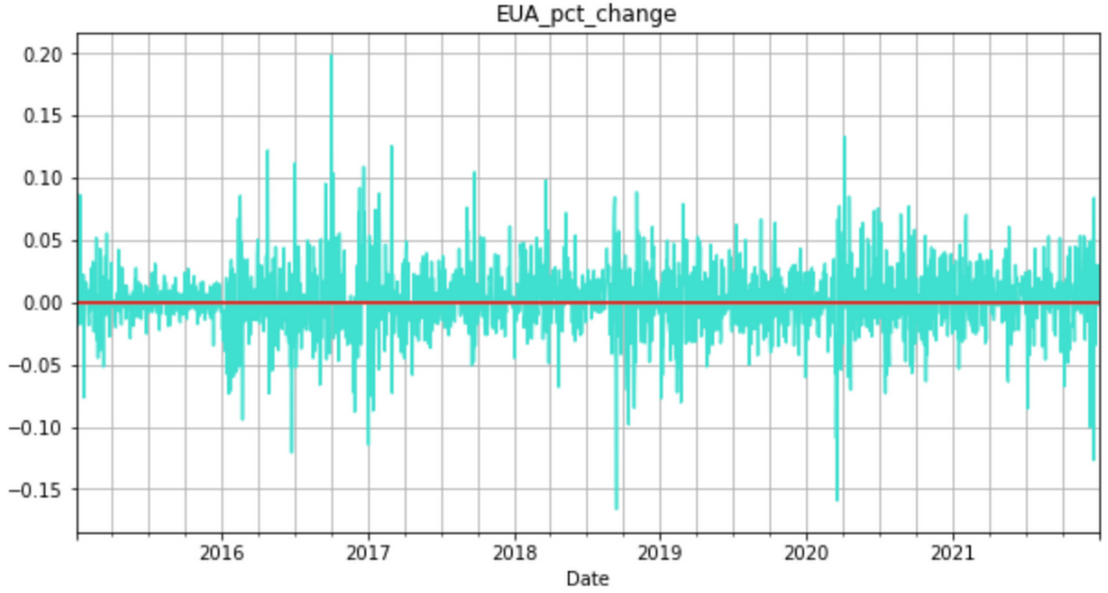


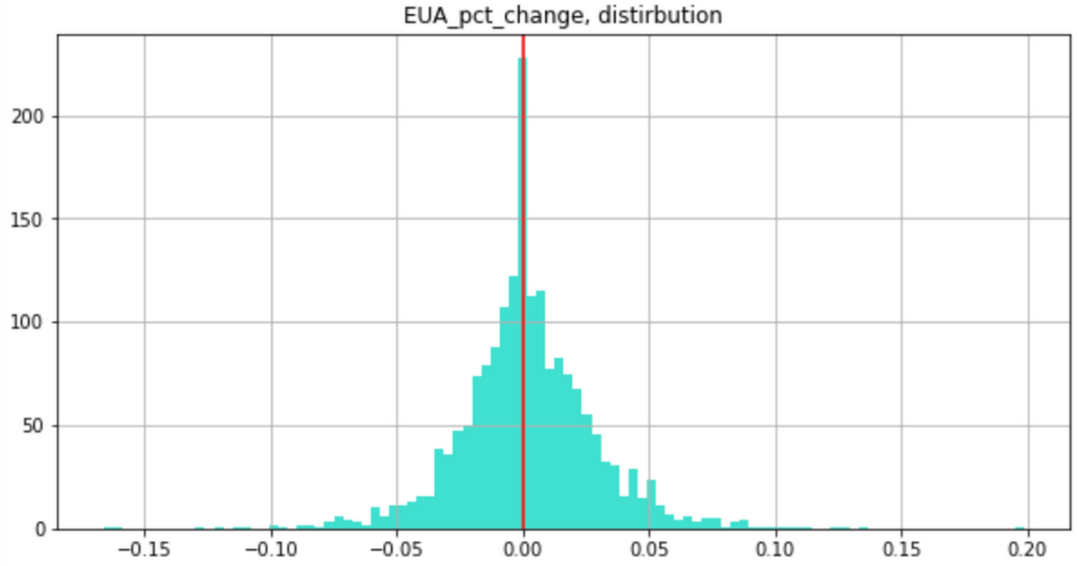

```
----------------------
model: lgbm
train_r2score: 1.0
test_r2score: -1.18
train_mse: 0.0
test_mse: 544.94
```

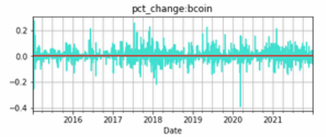


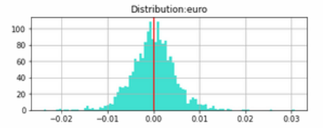

```
----------------------
model: xgb
train_r2score: 1.0
test_r2score: -1.41
train_mse: 0.0
test_mse: 602.32
```

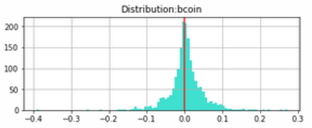
Target changed to pct_change

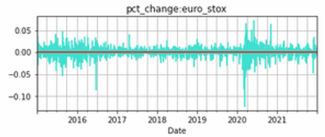


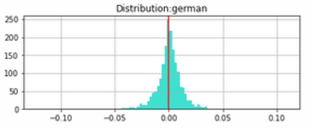


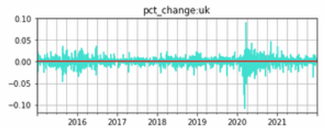


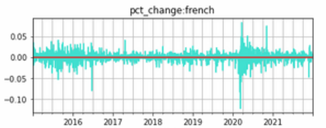


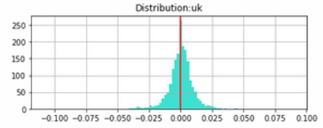


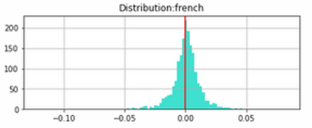


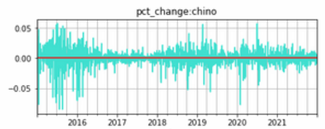


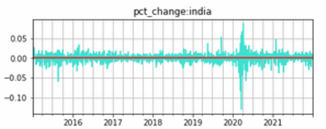


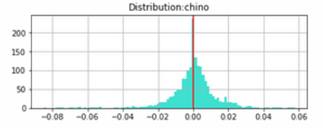


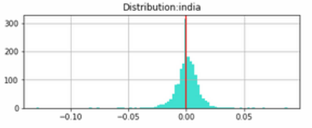


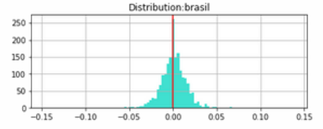


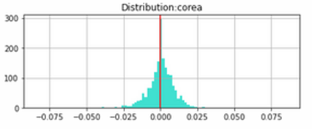


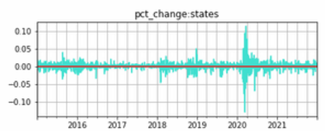


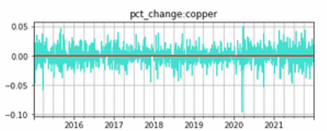


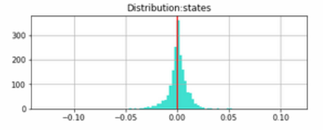


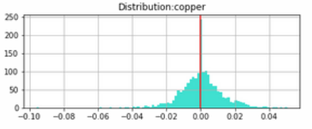


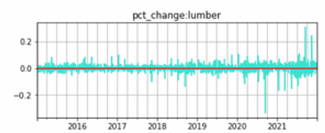


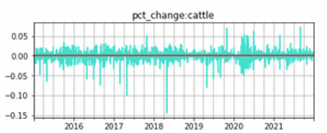


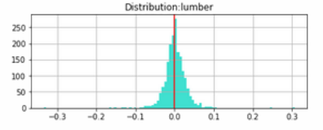


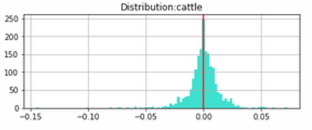


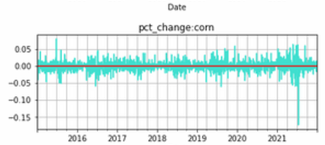


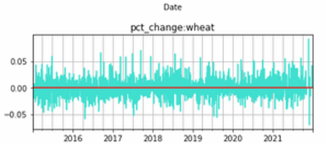


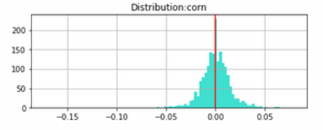


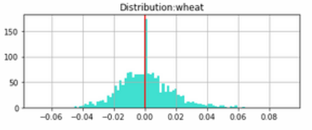


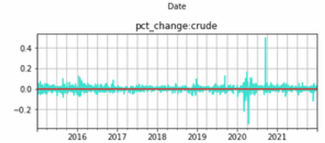


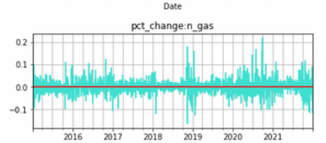


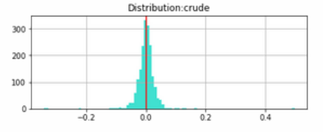


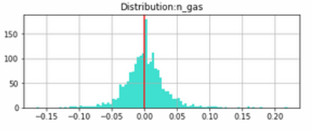




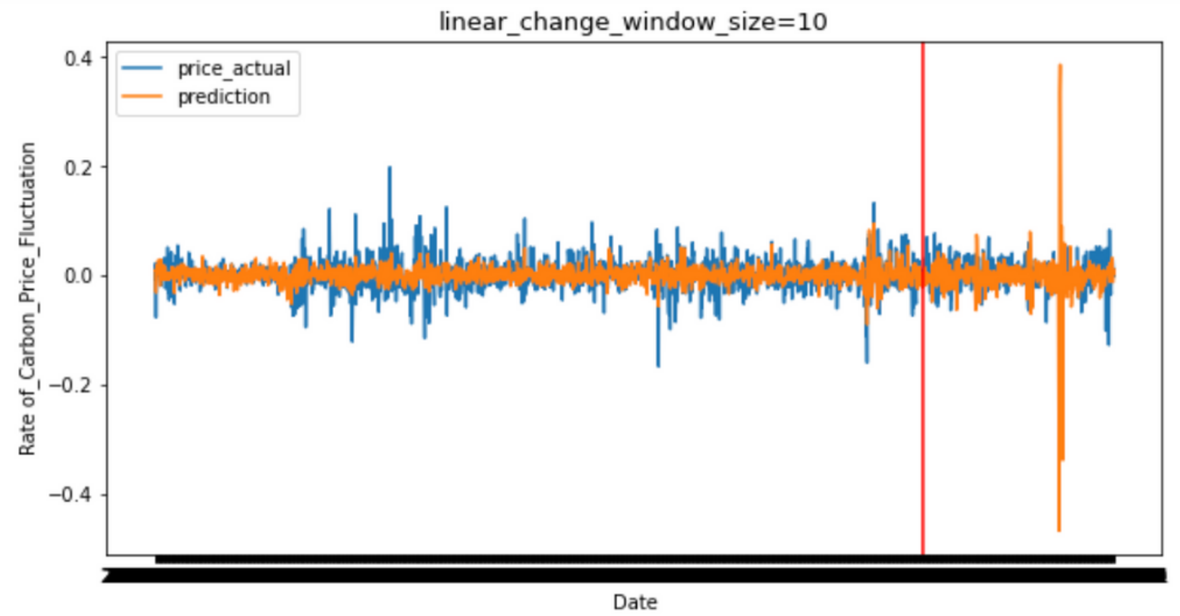


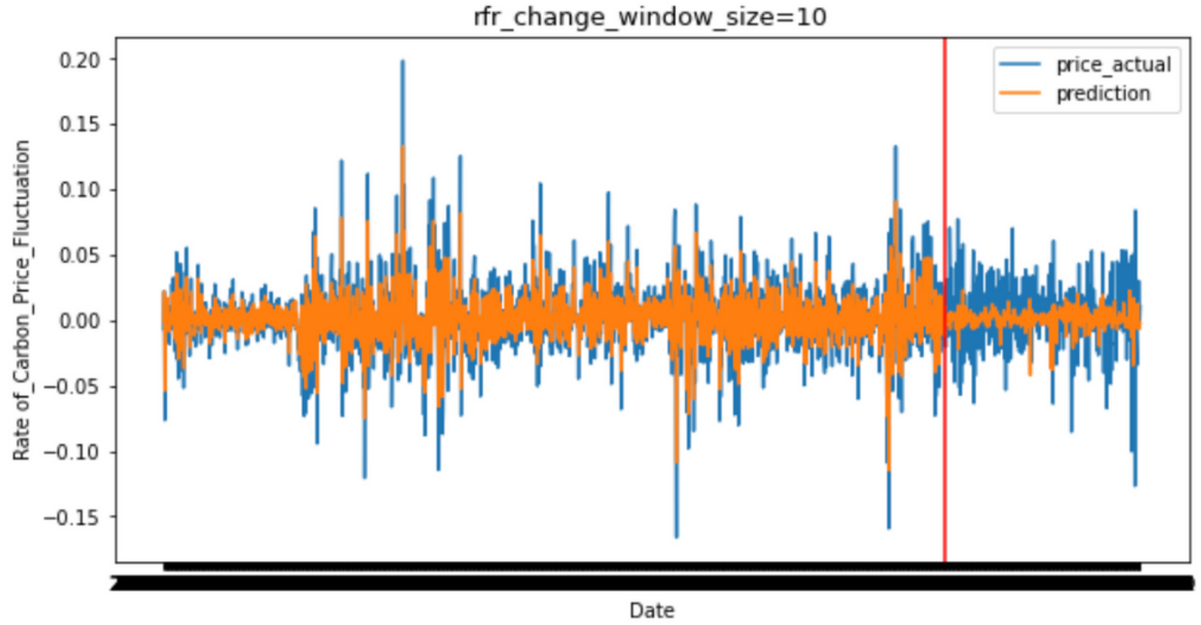






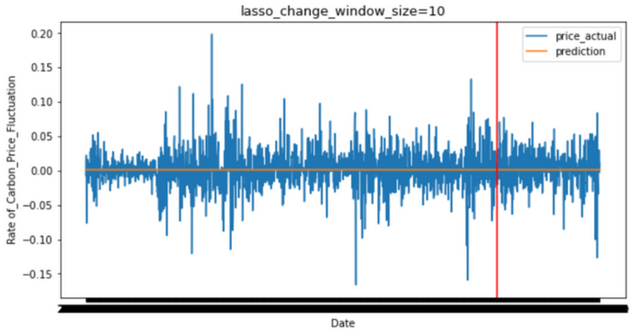
Results: window size=10

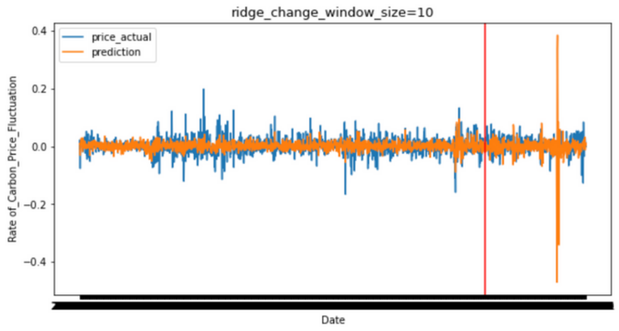


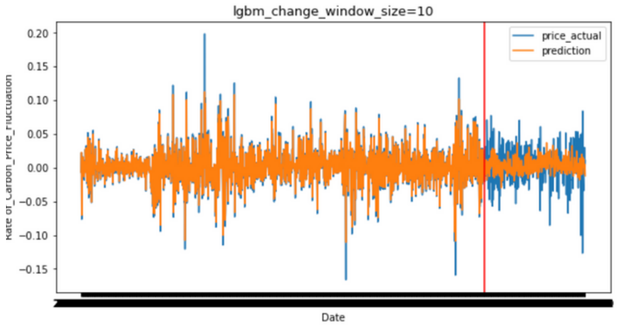


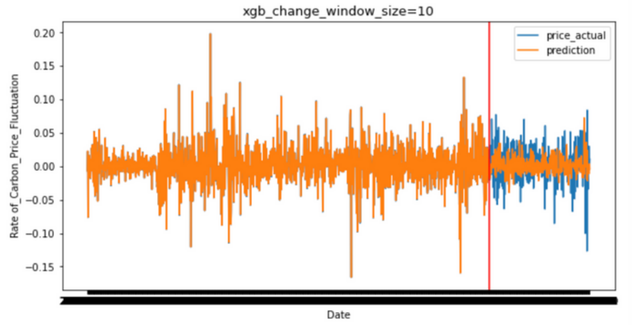

```
----------------------
model: linear
train_r2score: 0.27
test_r2score: -3.23
train_mse: 0.0
test_mse: 0.0
```

```
----------------------
model: rfr
train_r2score: 0.86
test_r2score: -0.08
train_mse: 0.0
test_mse: 0.0
```









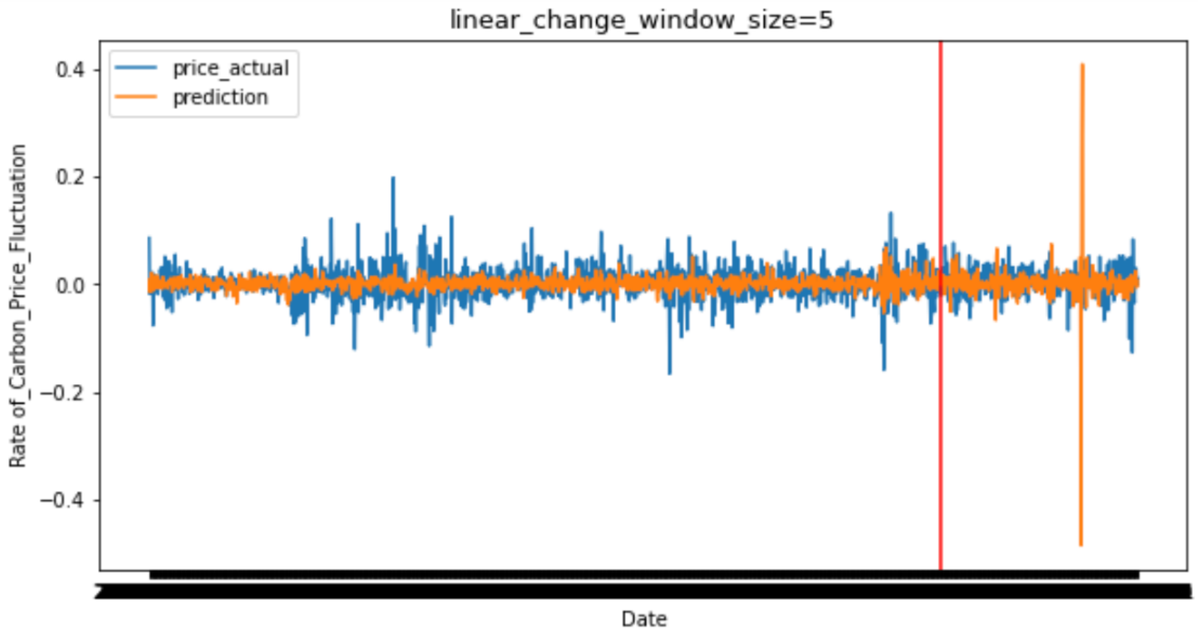

```
----------------------
model: lasso
train_r2score: 0.0
test_r2score: -0.01
train_mse: 0.0
test_mse: 0.0
```

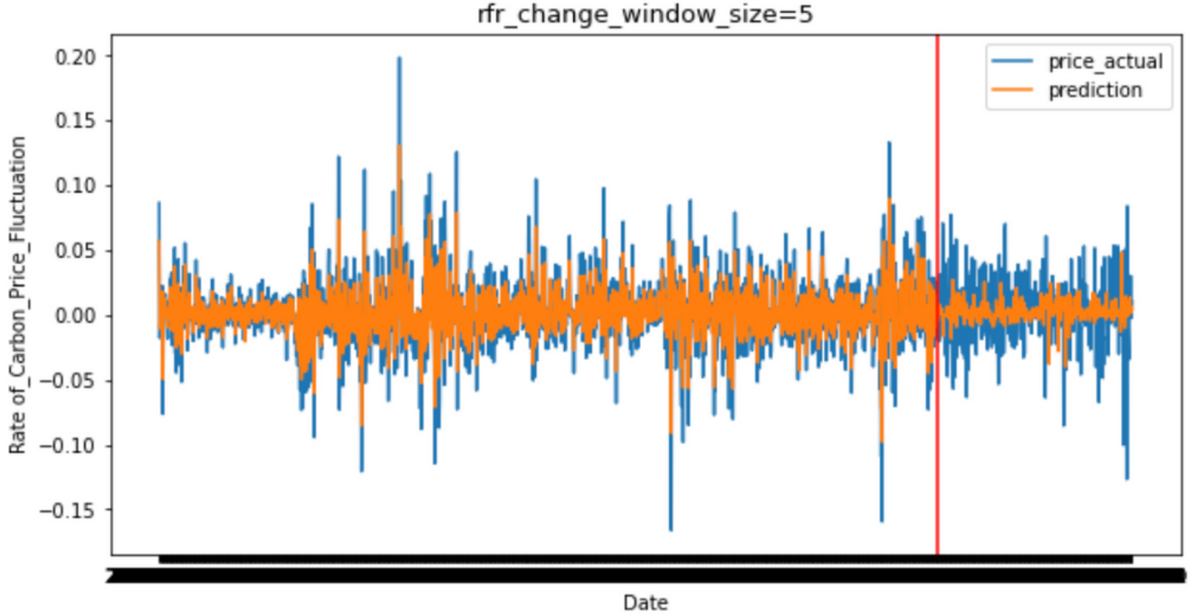
```
----------------------
model: ridge
train_r2score: 0.27
test_r2score: -3.23
train_mse: 0.0
test_mse: 0.0
```

```
----------------------
odel: lgbm
rain_r2score: 0.98
est_r2score: -0.26
rain_mse: 0.0
est_mse: 0.0
```

```
----------------------
model: xgb
train_r2score: 1.0
test_r2score: -0.26
train_mse: 0.0
test_mse: 0.0
```

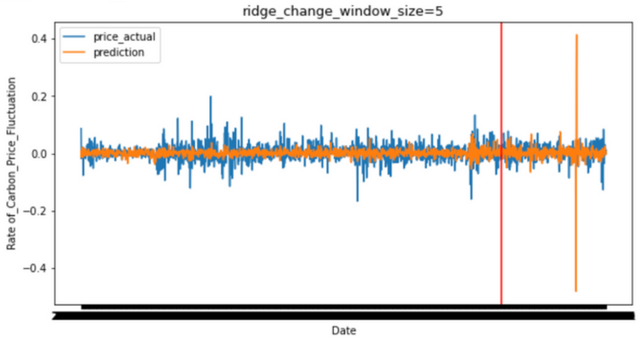
Results: window size=5

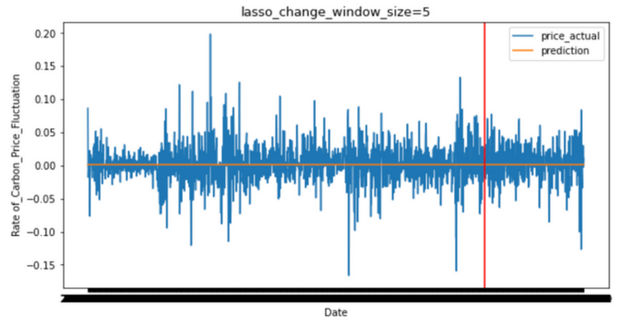


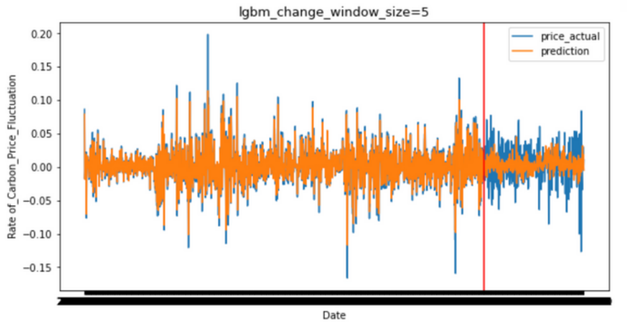


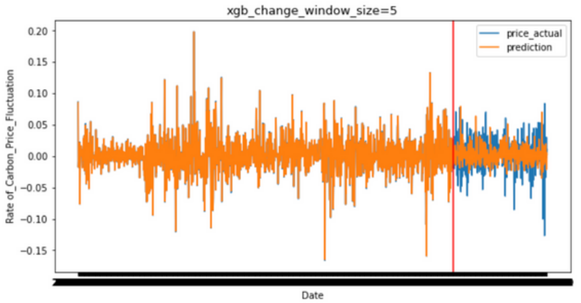

```
----------------------
model: linear
train_r2score: 0.15
test_r2score: -2.38
train_mse: 0.0
test_mse: 0.0
```

```
----------------------
model: rfr
train_r2score: 0.86
test_r2score: -0.13
train_mse: 0.0
test_mse: 0.0
```









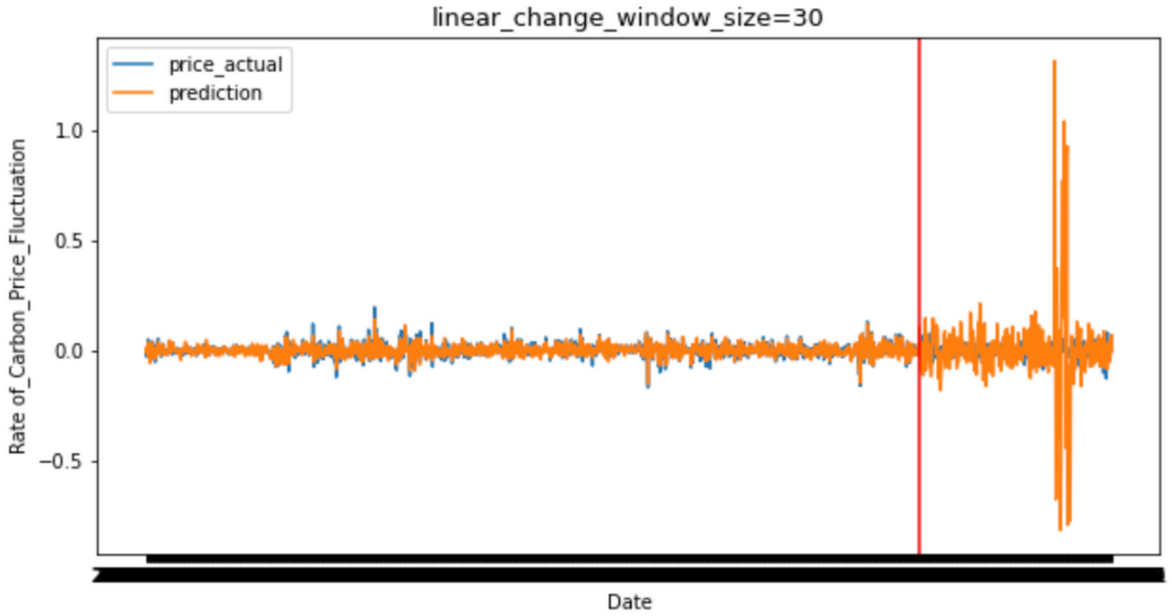

```
----------------------
model: ridge
train_r2score: 0.15
test_r2score: -2.34
train_mse: 0.0
test_mse: 0.0
```

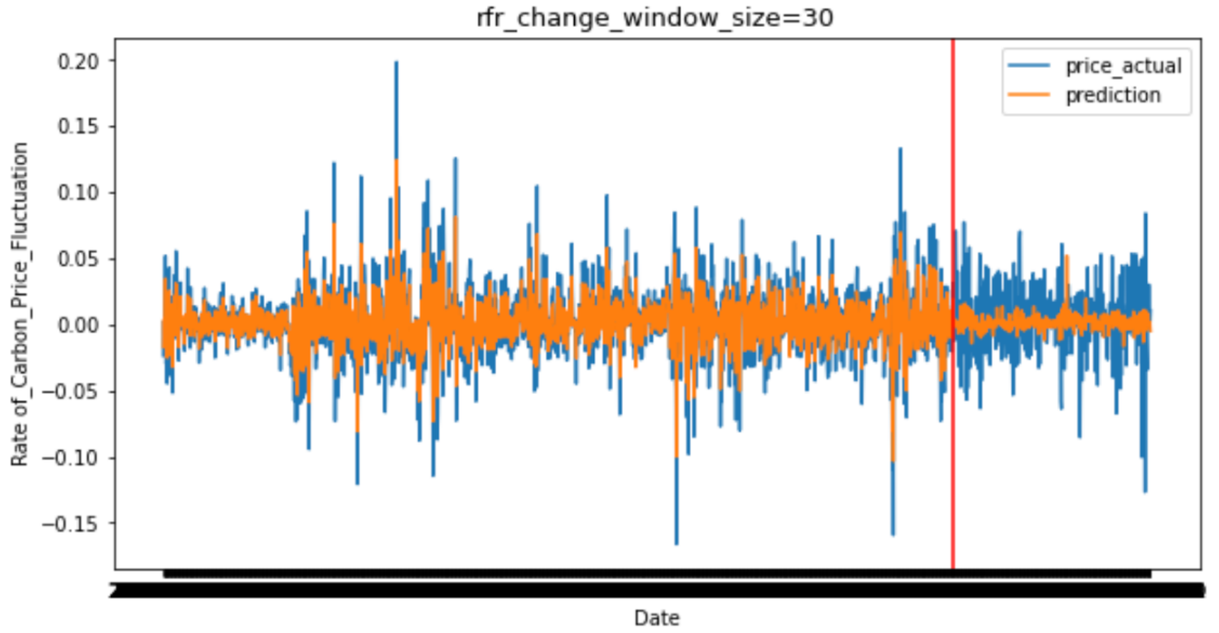
```
----------------------
model: lasso
train_r2score: 0.0
test_r2score: -0.01
train_mse: 0.0
test_mse: 0.0
```

```
----------------------
model: lgbm
train_r2score: 0.97
test_r2score: -0.24
train_mse: 0.0
test_mse: 0.0
```

```
----------------------
model: xgb
train_r2score: 1.0
test_r2score: -0.36
train_mse: 0.0
test_mse: 0.0
```

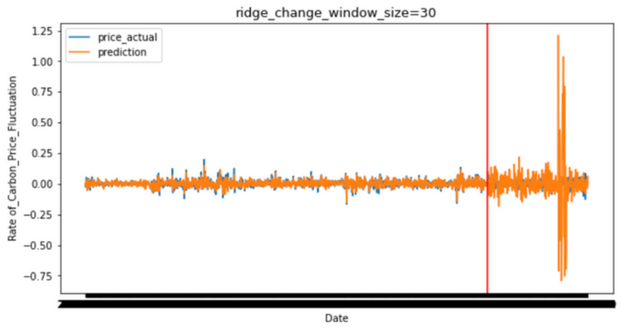
Results: window size=30

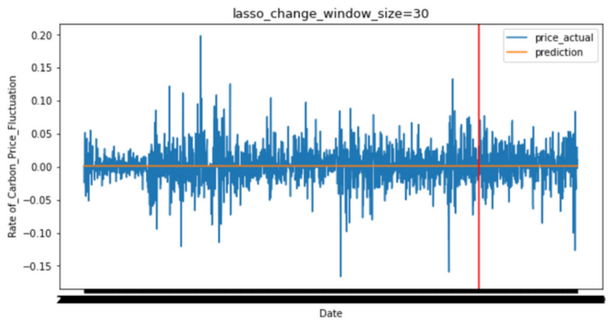


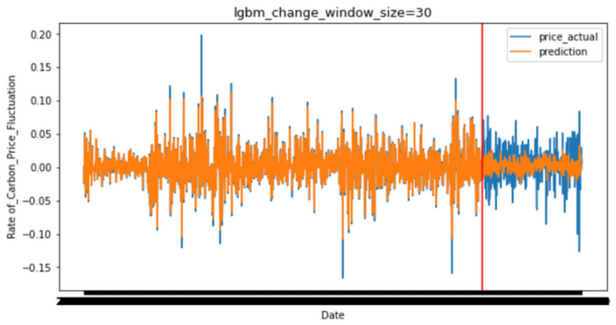


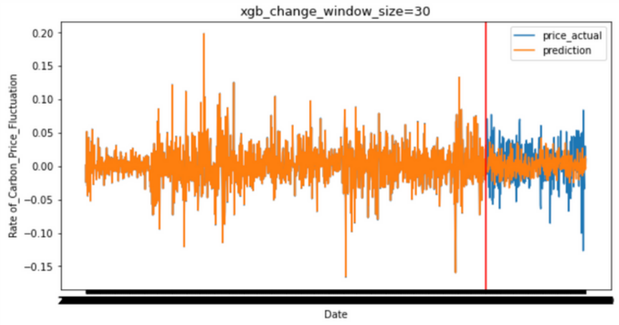

```
----------------------
model: linear
train_r2score: 0.78
test_r2score: -44.11
train_mse: 0.0
test_mse: 0.03
```

```
----------------------
model: rfr
train_r2score: 0.85
test_r2score: -0.04
train_mse: 0.0
test_mse: 0.0
```










```
----------------------
model: ridge
train_r2score: 0.79
test_r2score: -39.87
train_mse: 0.0
test_mse: 0.03
```

```
----------------------
model: lasso
train_r2score: 0.0
test_r2score: -0.01
train_mse: 0.0
test_mse: 0.0
```

```
----------------------
model: lgbm
train_r2score: 0.98
test_r2score: -0.14
train_mse: 0.0
test_mse: 0.0
```

```
----------------------
model: xgb
train_r2score: 1.0
test_r2score: -0.23
train_mse: 0.0
test_mse: 0.0
```

Deep Learning Models

Model outline

- Features
  - close price and trade volume of 17 indexes
    - Currencies: 'euro', 'bitcoin'
    - Stock market indexes: 'German', 'UK', 'France', 'China', 'India', 'Brazil', 'Korea', 'United States', 'Euro Stoxx
    - Commodities: 'copper', 'lumber', 'cattle', 'corn', 'wheat', 'crude oil', 'natural gas'
    - imported using FinanceDataReader
    - date from 2015.1.4 to 2021.12.31

- window size : 5 / 7 / 10 / 30 Days
- Epochs/ Batch size : 100/20, 100/10
- Layers: LSTM, Dense, Dropout
- Accuracy metrics: validation loss and validation accuracy
- Train, test, split: the latest 10% data is separated as test data

Basic Code for Deep Learning Model

data split / scale

```
def split_xy5(dataset, time_steps):
    x, y = list(), list()
    for i in range(len(dataset)):
        x_end_number = i + time_steps

        if x_end_number > len(dataset)-1:
            break
        tmp_x = dataset.iloc[i:x_end_number, :]
        tmp_y = dataset.iloc[x_end_number, 18]
        x.append(tmp_x)
        y.append(tmp_y)
    return np.array(x), np.array(y)

x, y = split_xy5(df, 30)

print(x[0,:], y[0])
print(x.shape, y.shape)
```

...

```
tt_split = int(len(x)*0.9)
x_train= x[:tt_split]
y_train= y[:tt_split]

x_test = x[tt_split:]
y_test = y[tt_split:]
print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)
```

...

```
x_train2d = np.reshape(x_train, (-1, x.shape[1]*x.shape[2]))
x_test2d = np.reshape(x_test, (-1, x.shape[1]*x.shape[2]))
x_2d = np.reshape(x,(-1, x.shape[1]*x.shape[2]))

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(x_train2d)
x_train2d_scaled = scaler.transform(x_train2d)
x_test2d_scaled = scaler.transform(x_test2d)
x_2d_scaled = scaler.transform(x_2d)

x_train3d = x_train2d_scaled.reshape(-1, x.shape[1], x.shape[2])
x_test3d = x_test2d_scaled.reshape(-1, x.shape[1], x.shape[2])
x_3d = x_2d_scaled.reshape(-1, x.shape[1], x.shape[2])
```

model layer/fit/compile

```
from keras.layers import Dense
from keras.callbacks import EarlyStopping
import keras

model = Sequential()

model.add(LSTM(128, activation='relu', return_sequences=True,
               input_shape=(x_3d.shape[1], x_3d.shape[2])))
model.add(LSTM(128, activation='relu',return_sequences=True))
model.add(LSTM(128, activation='relu',return_sequences=True))
model.add(LSTM(128, activation='relu',return_sequences=True))
model.add(LSTM(128, activation='relu',return_sequences=True))
model.add(LSTM(128, activation='relu'))
model.add(Dense(32, activation ='relu'))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mse', metrics =['accuracy'])
model.summary()
```

...

```
epoch = 100
callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss', patience=20),\
             keras.callbacks.ModelCheckpoint(filepath='best_model.h5',
                                             monitor='val_loss',
                                             save_best_only=True)]

history = model.fit(x_train3d, y_train, verbose=1,
         batch_size = 20, epochs = epoch, validation_split=0.11, callbacks= ca

model = keras.models.load_model('best_model.h5')
score_train = model.evaluate(x_train3d, y_train, verbose=0)
score_test = model.evaluate(x_test3d, y_test, verbose=0)
print('train_score')
print("%.2f%%" % (score_train[1]*100))
print('test_score')
print("%.2f%%" % (score_test[1]*100))

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
print('min_loss_values:', np.min(loss_values))
print('min_loss_values_val:', np.min(val_loss_values))
```

result visualization

```
epochs = range(len(loss_values))
plt.plot(epochs, loss_values, 'b-', label = 'Training Loss')
plt.plot(epochs, val_loss_values, 'g-', label = 'Validation Loss')
plt.plot(epochs, acc_values, 'b+', label = 'Training Acc')
plt.plot(epochs, val_acc_values, 'g+', label = 'Validation Acc')
plt.title('Training and Validation Loss(simple/window_size=30)')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

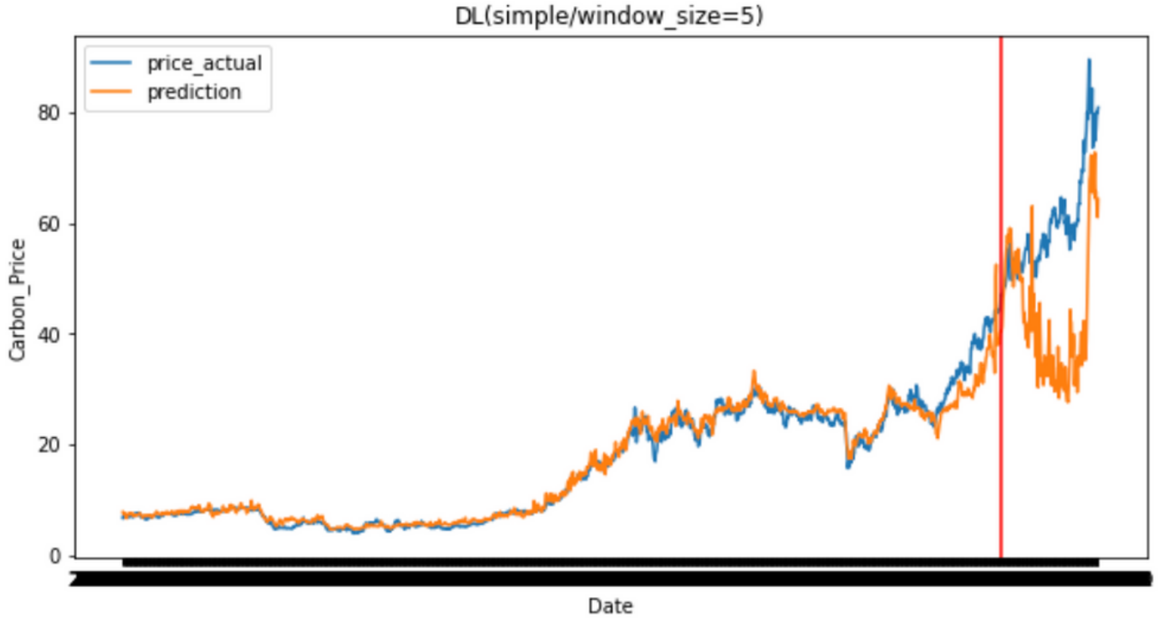
...

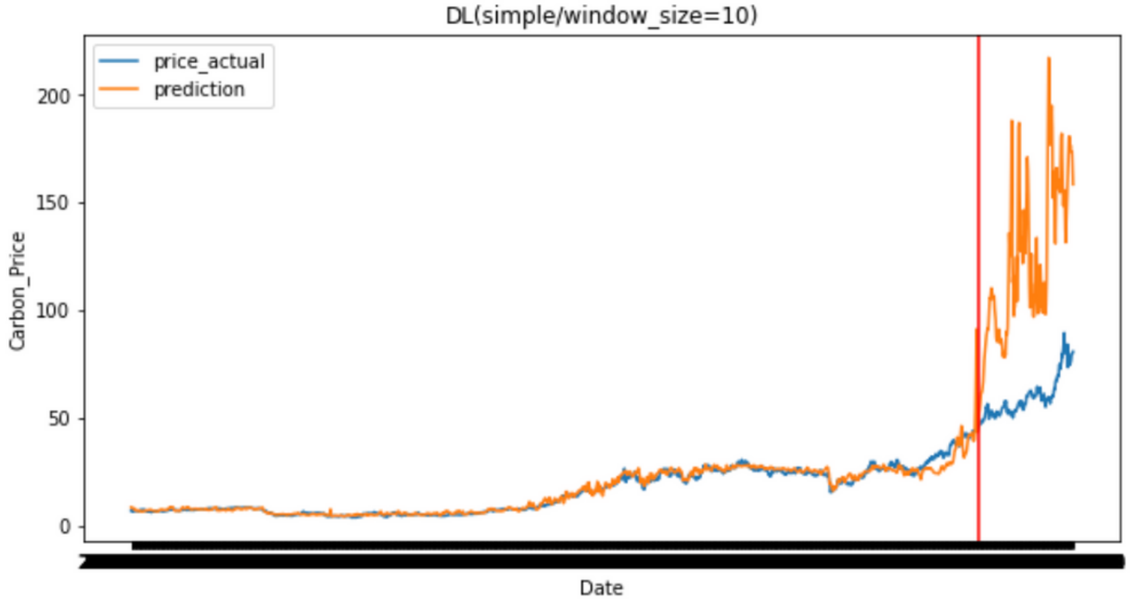
```
epochs = range(len(loss_values))
#plt.plot(epochs, loss_values, 'b-', label = 'Training Loss')
plt.plot(epochs, val_loss_values, 'g-', label = 'Validation Loss')
plt.plot(epochs, acc_values, 'b+', label = 'Training Acc')
plt.plot(epochs, val_acc_values, 'g+', label = 'Validation Acc')
plt.title('Training and Validation Loss(simple/window_size=30)')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

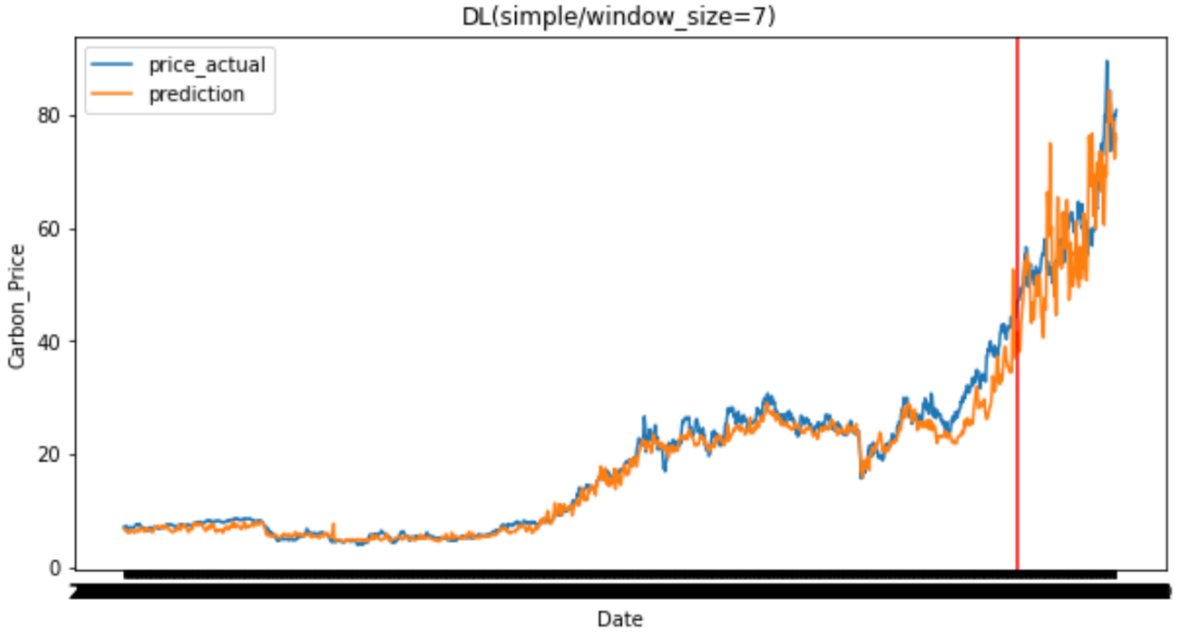
...

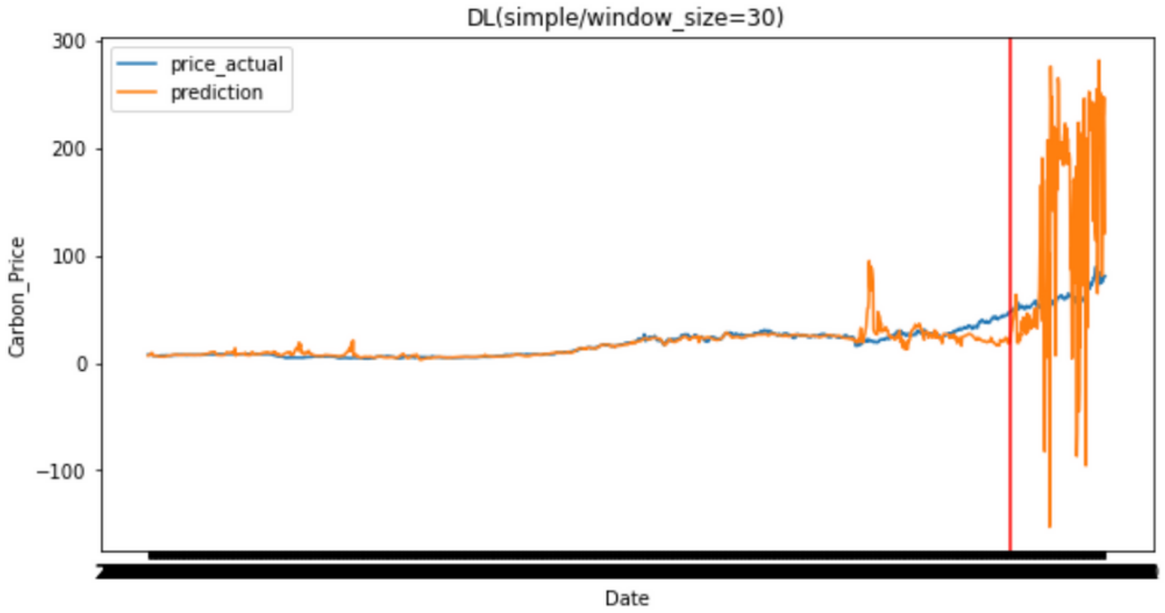
```
x_pred = model.predict(x_3d)
df_x_pred = pd.DataFrame(x_pred, index = df.index[30:])
df_y = pd.DataFrame(y, index=df.index[30:])
plt.figure(figsize = (10,5))

plt.plot(df_y, label = 'price_actual')
plt.plot(df_x_pred, label = 'prediction')
plt.axvline(df_x_pred.index[x_train.shape[0]], color = 'red', ls='-')

plt.title('DL(simple/window_size=30)')

plt.xlabel('Date')
plt.ylabel('Carbon_Price')

plt.legend()
plt.show()
```

Results: basic models by window size

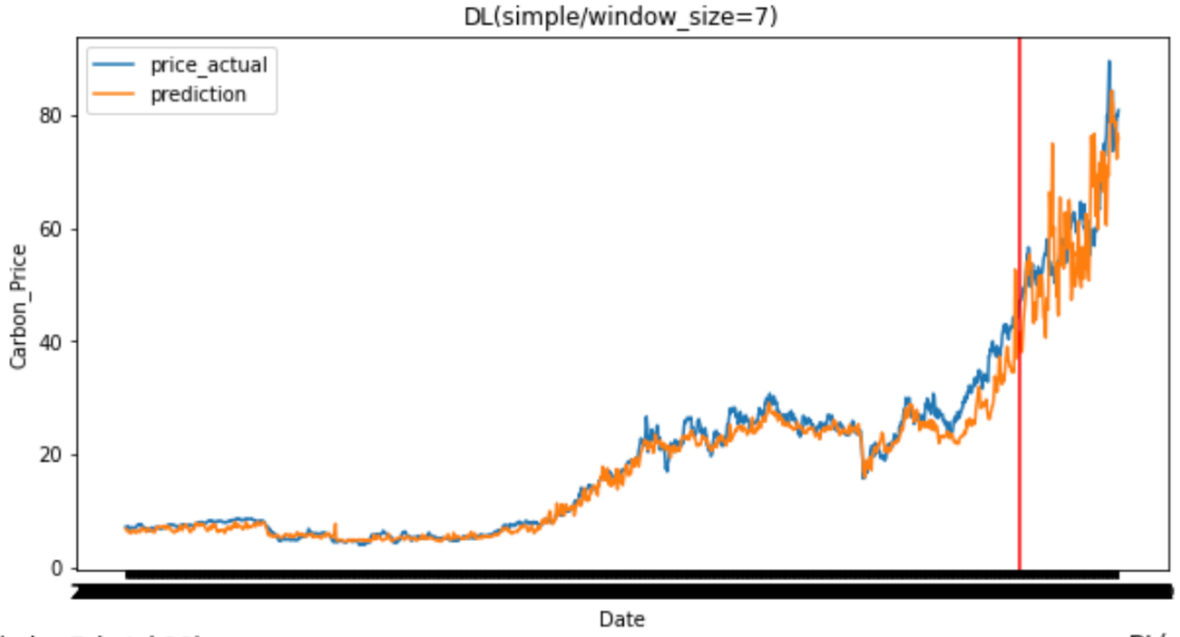






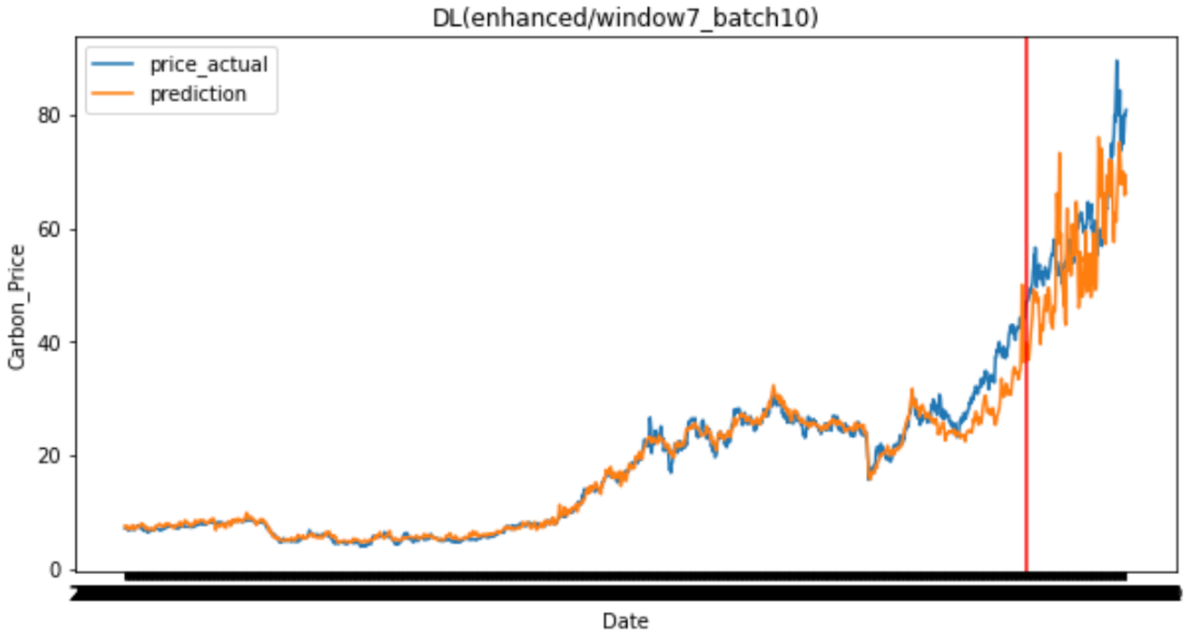


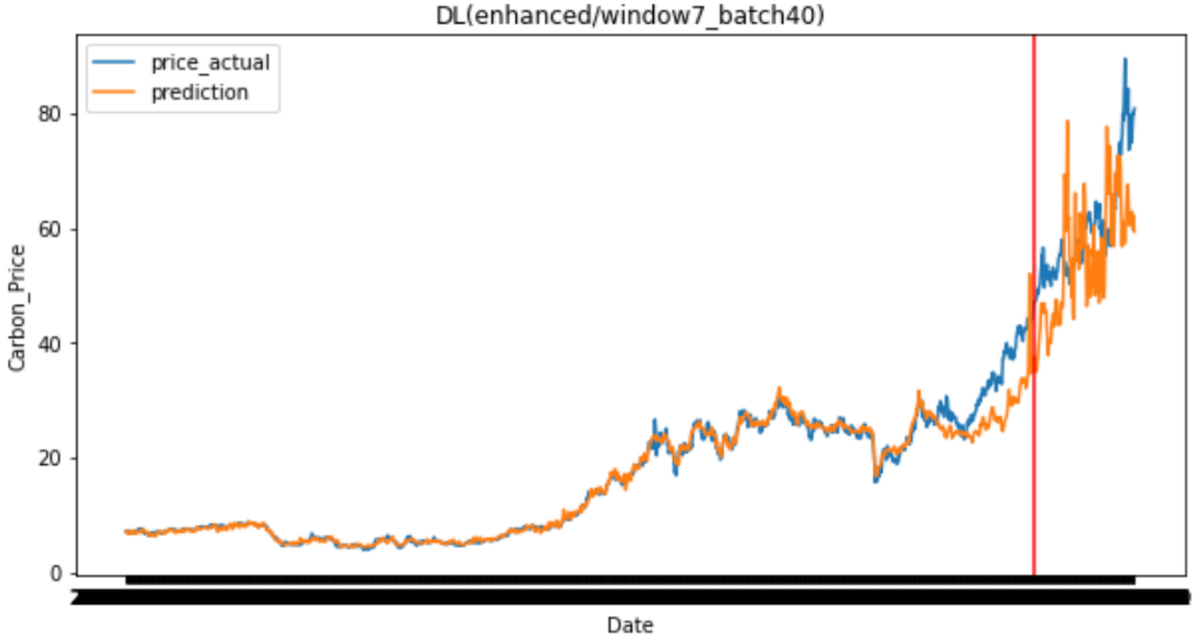
Results: batch size adjusting(20 -> 10/40)



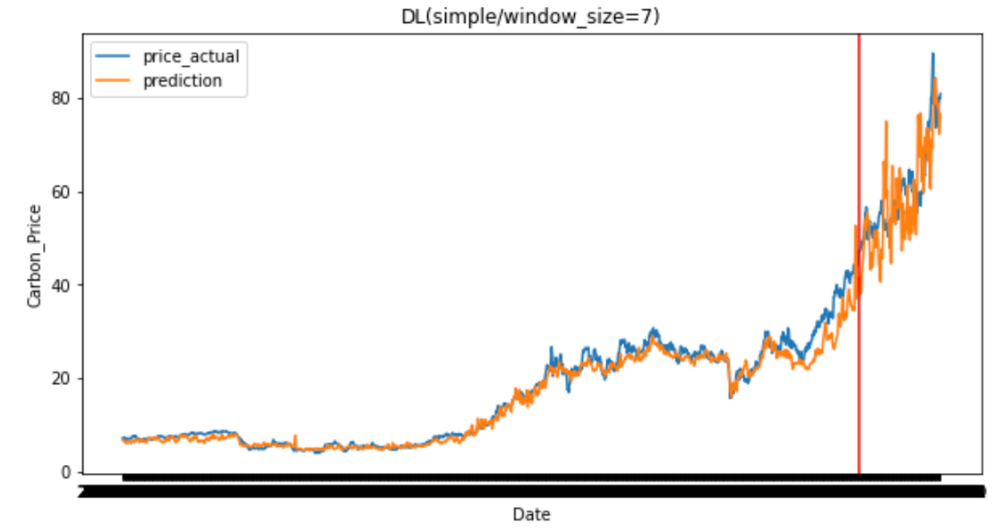
validation loss

size10 = 33.020
size20 = 28.677
size40 = 42.159





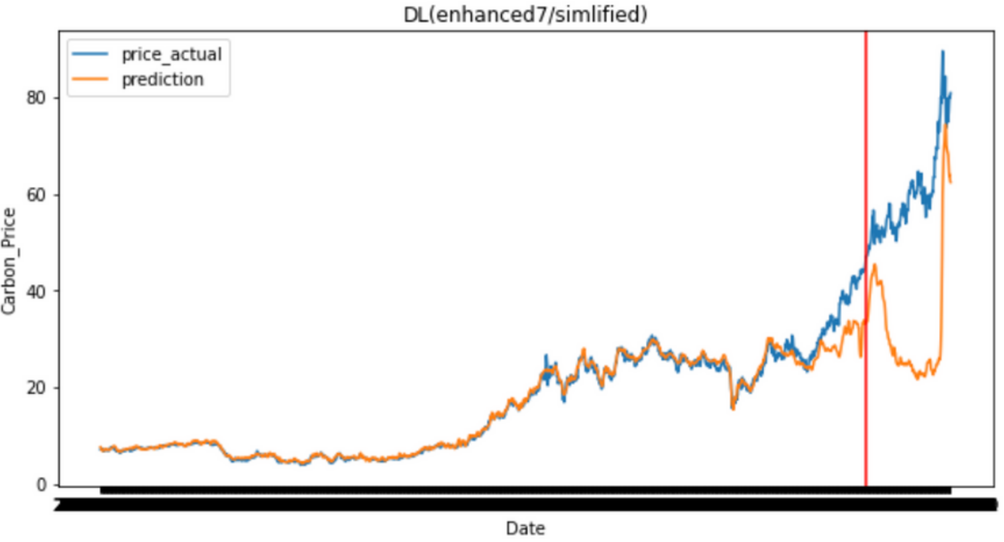
Results: simple model(window=7, batch=20)



Model: "sequential_11"

Layer (type)	Output Shape	Param #
lstm_63 (LSTM)	(None, 7, 128)	85504
lstm_64 (LSTM)	(None, 7, 128)	131584
lstm_65 (LSTM)	(None, 7, 128)	131584
lstm_66 (LSTM)	(None, 7, 128)	131584
lstm_67 (LSTM)	(None, 7, 128)	131584
lstm_68 (LSTM)	(None, 128)	131584
dense_22 (Dense)	(None, 32)	4128
dense_23 (Dense)	(None, 1)	33

Total params: 747,585
Trainable params: 747,585
Non-trainable params: 0



Model: "sequential_13"

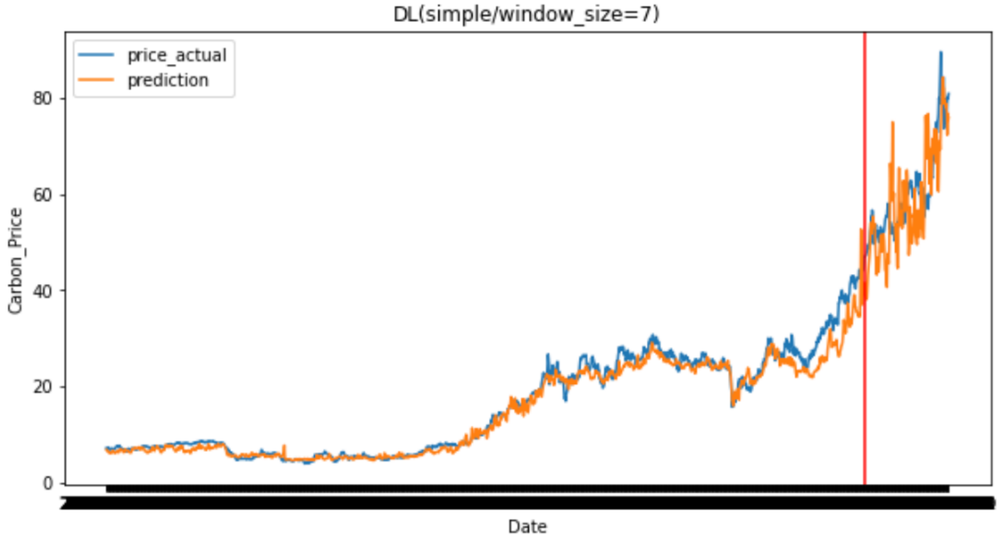
Layer (type)	Output Shape	Param #
lstm_75 (LSTM)	(None, 7, 128)	85504
lstm_76 (LSTM)	(None, 7, 128)	131584
lstm_77 (LSTM)	(None, 128)	131584
dense_26 (Dense)	(None, 32)	4128
dense_27 (Dense)	(None, 1)	33

Total params: 352,833
Trainable params: 352,833
Non-trainable params: 0

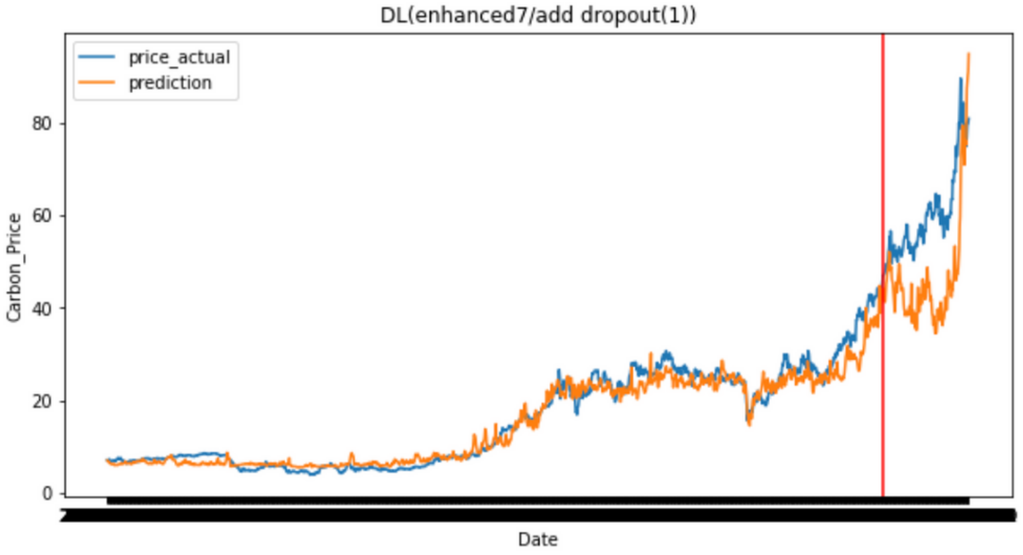
validation loss

basic = 28.677
simple = 38.038

Results: 1 drop-out layer(window=7, batch=20)




```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_63 (LSTM)               (None, 7, 128)            85504
_________________________________________________________________
lstm_64 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_65 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_66 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_67 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_68 (LSTM)               (None, 128)               131584
_________________________________________________________________
dense_22 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 33
=================================================================
Total params: 747,585
Trainable params: 747,585
Non-trainable params: 0
_________________________________________________________________
```

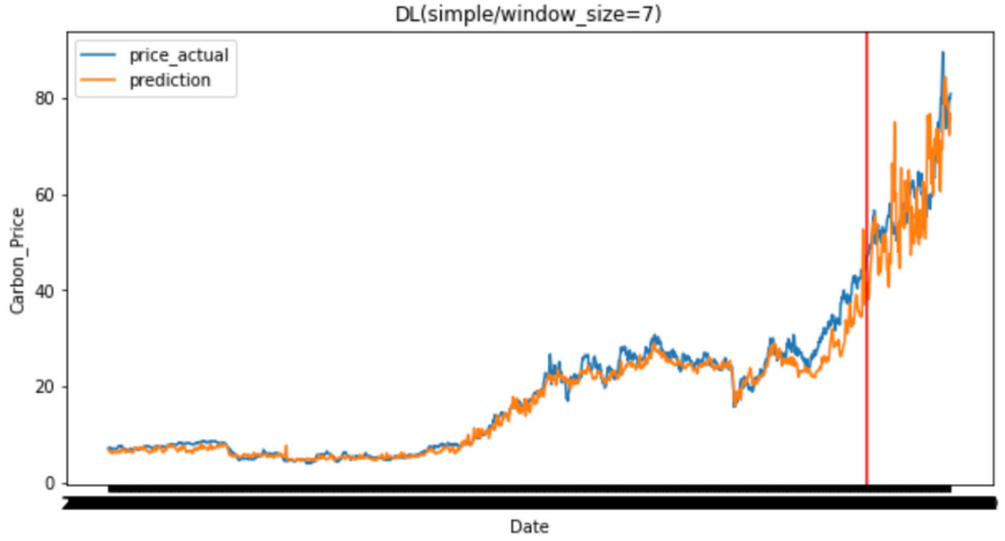



```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_78 (LSTM)               (None, 7, 128)            85504
_________________________________________________________________
lstm_79 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_80 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_81 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 128)            0
_________________________________________________________________
lstm_82 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_83 (LSTM)               (None, 128)               131584
_________________________________________________________________
dense_28 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_29 (Dense)             (None, 1)                 33
=================================================================
Total params: 747,585
Trainable params: 747,585
Non-trainable params: 0
_________________________________________________________________
```

validation loss

basic = 28.677
1D-O = 24.005

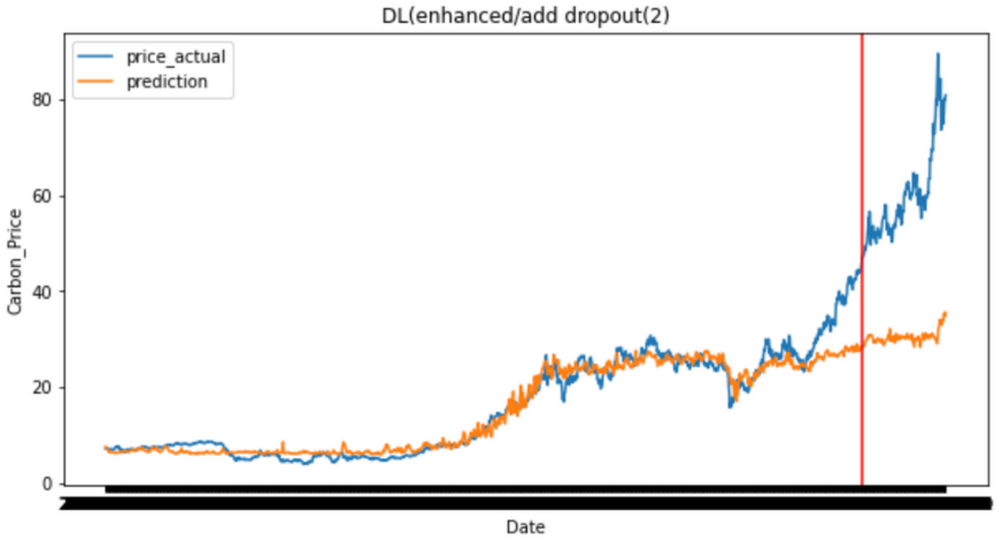
Results: 2drop-out layers(window=7, batch=20)



Model: "sequential_11"

Layer (type)	Output Shape	Param #
lstm_63 (LSTM)	(None, 7, 128)	85504
lstm_64 (LSTM)	(None, 7, 128)	131584
lstm_65 (LSTM)	(None, 7, 128)	131584
lstm_66 (LSTM)	(None, 7, 128)	131584
lstm_67 (LSTM)	(None, 7, 128)	131584
lstm_68 (LSTM)	(None, 128)	131584
dense_22 (Dense)	(None, 32)	4128
dense_23 (Dense)	(None, 1)	33

Total params: 747,585
Trainable params: 747,585
Non-trainable params: 0



Model: "sequential_15"

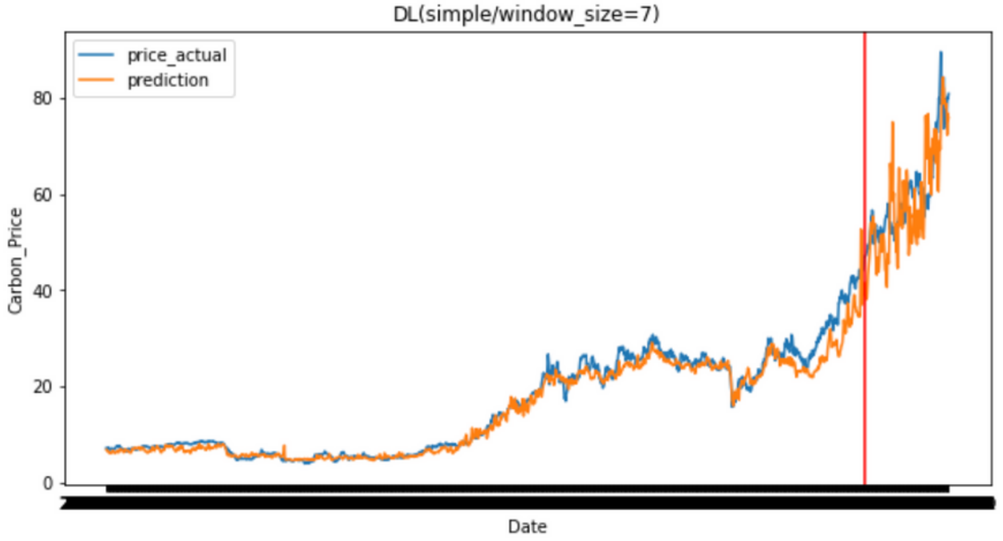
Layer (type)	Output Shape	Param #
lstm_84 (LSTM)	(None, 7, 128)	85504
lstm_85 (LSTM)	(None, 7, 128)	131584
dropout_2 (Dropout)	(None, 7, 128)	0
lstm_86 (LSTM)	(None, 7, 128)	131584
lstm_87 (LSTM)	(None, 7, 128)	131584
lstm_88 (LSTM)	(None, 7, 128)	131584
dropout_3 (Dropout)	(None, 7, 128)	0
lstm_89 (LSTM)	(None, 128)	131584
dense_30 (Dense)	(None, 32)	4128
dense_31 (Dense)	(None, 1)	33

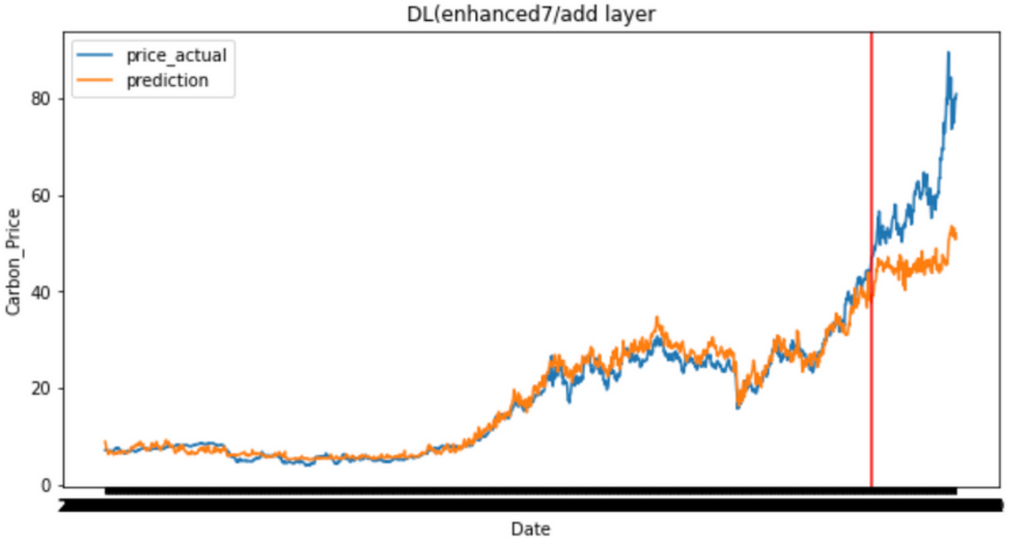
Total params: 747,585
Trainable params: 747,585
Non-trainable params: 0

validation loss

basic = 28.677
2D-O = 68.324

Results: Add LSTM layers(window=7, batch=20)





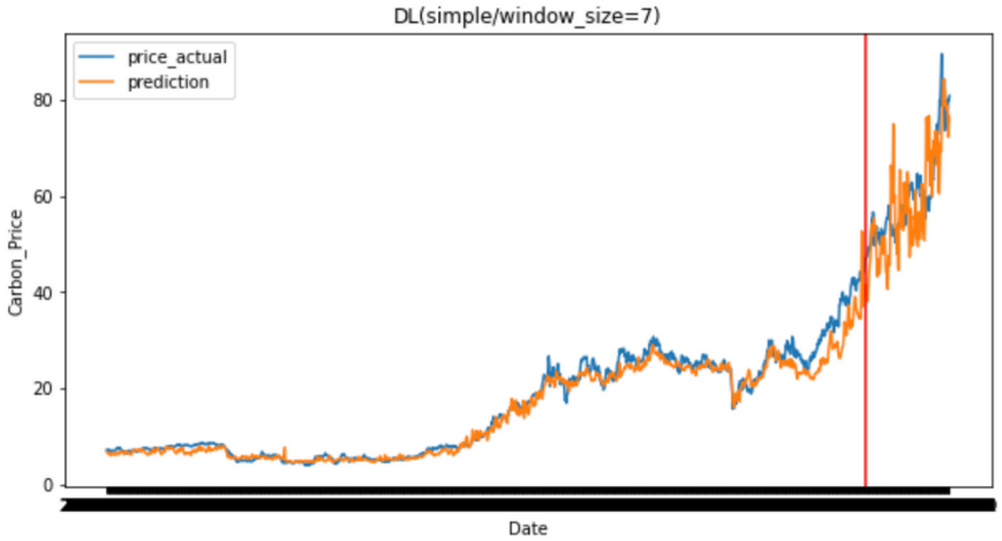

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_63 (LSTM)               (None, 7, 128)            85504
_________________________________________________________________
lstm_64 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_65 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_66 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_67 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_68 (LSTM)               (None, 128)               131584
_________________________________________________________________
dense_22 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 33
=================================================================
Total params: 747,585
Trainable params: 747,585
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_96 (LSTM)               (None, 7, 128)            85504
_________________________________________________________________
lstm_97 (LSTM)               (None, 7, 128)            131584
_________________________________________________________________
lstm_98 (LSTM)               (None, 7, 64)             49408
_________________________________________________________________
lstm_99 (LSTM)               (None, 7, 128)            98816
_________________________________________________________________
lstm_100 (LSTM)              (None, 7, 128)            131584
_________________________________________________________________
lstm_101 (LSTM)              (None, 7, 32)             20608
_________________________________________________________________
lstm_102 (LSTM)              (None, 7, 128)            82432
_________________________________________________________________
lstm_103 (LSTM)              (None, 7, 16)             9280
_________________________________________________________________
lstm_104 (LSTM)              (None, 7, 128)            74240
_________________________________________________________________
lstm_105 (LSTM)              (None, 128)               131584
_________________________________________________________________
dense_32 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_33 (Dense)             (None, 1)                 33
=================================================================
Total params: 819,201
Trainable params: 819,201
Non-trainable params: 0
_________________________________________________________________
```

validation loss

basic = 28.677
add LSTM= 7.844

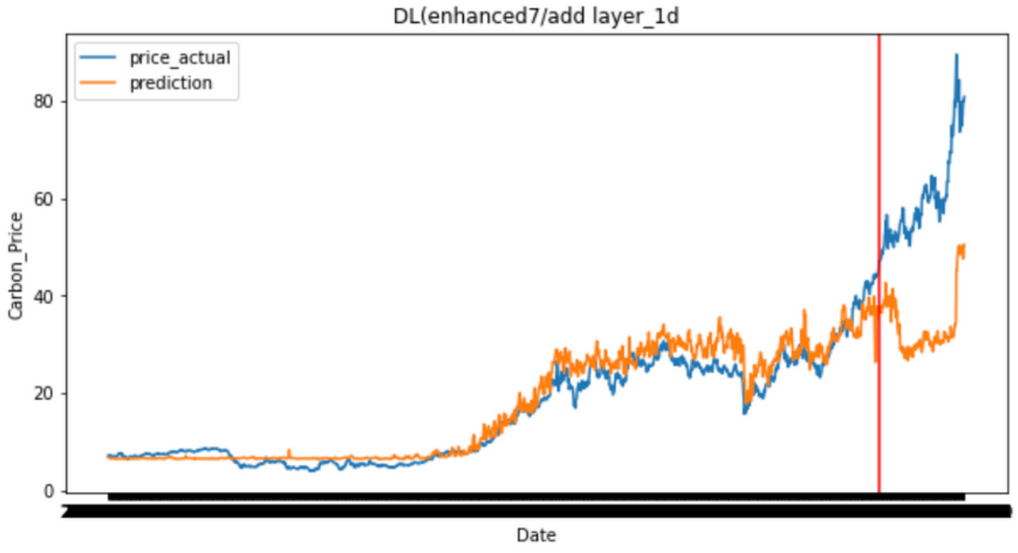
Results: Add layers(window=7, batch=20)



Model: "sequential_11"

Layer (type)	Output Shape	Param #
lstm_63 (LSTM)	(None, 7, 128)	85504
lstm_64 (LSTM)	(None, 7, 128)	131584
lstm_65 (LSTM)	(None, 7, 128)	131584
lstm_66 (LSTM)	(None, 7, 128)	131584
lstm_67 (LSTM)	(None, 7, 128)	131584
lstm_68 (LSTM)	(None, 128)	131584
dense_22 (Dense)	(None, 32)	4128
dense_23 (Dense)	(None, 1)	33

Total params: 747,585
Trainable params: 747,585
Non-trainable params: 0



Model: "sequential_20"

Layer (type)	Output Shape	Param #
lstm_116 (LSTM)	(None, 7, 128)	85504
lstm_117 (LSTM)	(None, 7, 128)	131584
lstm_118 (LSTM)	(None, 7, 64)	49408
lstm_119 (LSTM)	(None, 7, 32)	12416
lstm_120 (LSTM)	(None, 7, 32)	8320
lstm_121 (LSTM)	(None, 7, 128)	82432
lstm_122 (LSTM)	(None, 7, 128)	131584
lstm_123 (LSTM)	(None, 7, 32)	20608
lstm_124 (LSTM)	(None, 7, 128)	82432
lstm_125 (LSTM)	(None, 7, 16)	9280
lstm_126 (LSTM)	(None, 7, 128)	74240
lstm_127 (LSTM)	(None, 7, 128)	131584
lstm_128 (LSTM)	(None, 128)	131584
dense_36 (Dense)	(None, 32)	4128
dense_37 (Dense)	(None, 1)	33

Total params: 955,137
Trainable params: 955,137
Non-trainable params: 0

validation loss

basic = 28.677
add layers= 18.088

Implications & Discussion

Implications and Discussion

- **Linear Regression(window size=5) showed the best accuracy in predicting carbon prices (r2score:0.99, MSE:2.27)**
- **When the rate of change put into the model, the model performance was poor**
- **Although there is room to be finely adjusted the deep learning models, the linear model showed higher accuracy score, the start line of the refinement can be the relatively simpler model.**

Park Sinae

Thank you!

sinae515@gmail.com