Importing packages

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.graphics.tsaplots as sgt
import statsmodels.tsa.stattools as sts
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from arch import arch_model
from pmdarima.arima import auto_arima
from scipy.stats.distributions import chi2
import statsmodels.api as sm
from pylab import rcParams
from math import sqrt
import seaborn as sns
from datetime import datetime, timedelta
from arch import arch_model
import warnings
warnings.filterwarnings("ignore")
sns.set()
```

Importing Data

In [ ]:

```
raw_csv_data = pd.read_csv("data.csv", sep=';')
delu = raw_csv_data.copy()
delu.ddmmyy = pd.to_datetime(delu.ddmmyy, dayfirst = True)
delu.set_index("ddmmyy", inplace = True)
```

In [ ]:

```
raw_csv_prices = pd.read_csv("exogenuous.csv", sep=';')
exp = raw_csv_prices.copy()
exp.ddmmyy = pd.to_datetime(exp.ddmmyy, dayfirst = True)
exp.set_index("ddmmyy", inplace = True)
```

Converting data

In [ ]:

```
delu["DAP_ret"] = delu.DAP.pct_change(1)*100
delu["DAP_delta"] = delu.DAP.diff(1)
delu = delu.iloc[1:]
```

Data analysis

In [ ]:

```
delu.DAP.describe()
```

In [ ]:

```
sts.adfuller(delu.DAP_delta)
```

In [ ]:

```
sgt.plot_acf(delu.DAP_delta, lags = 50, zero = False)
plt.title ("ACF of Electricity Price Differences in DE-LU BZN in October 2019- March 2022", size = 10)
plt.show()
```

In [ ]:

```
sgt.plot_pacf(delu.DAP_delta, lags = 50, zero = False, method = "ols")
plt.title ("PACF of Electricity Price Differences in DE-LU BZN in October 2019- March 2022", size = 10)
plt.show()
```

In [ ]:

```
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(delu.DAP_delta, model='additive', period = 30)
fig = decomposition.plot()
plt.show()
```

Splitting the data

In [ ]:

```
size = int(len(delu)*0.9)
df, df_test = delu.iloc [:size], delu.iloc [size:]
```

In [ ]:

```
start_date = "2021-24-11"
end_date = "2022-03-31"
```

SARIMAX model construction

In [ ]:

```
model_auto = auto_arima (delu.DAP, exogenous = df[['ttf', 'api2', 'CO2','Load', 'Coal', 'Gas', 'Oil',
                                                    'Hydro', 'Pumped', 'Wind', 'Solar', 'Biofuels', 'Nuclear',
                                                   'AT_net', 'BE_net', 'CH_net', 'CZ_net', 'DK1_net', 'DK2_net',
                                                   'FR_net', 'NL_net', 'NO2_net','SE4_net']],
                          m = 7, max_order = 7,max_p = 2, max_q = 7, max_d = 2,
                           max_P = 7, max_Q = 7, max_D = 2,
                           maxiter = 50, alpha = 0.05, n_jobs = -1)
model_auto.summary()
```

Testing SARIMAX model

In [ ]:

```
df_model_auto_pred = pd.DataFrame (model_auto.predict(n_periods = len(df_test[start_date:end_date]),
                            exogenous = df_test[['ttf', 'api2', 'CO2','Load', 'Coal', 'Gas', 'Oil',
                                                    'Hydro', 'Pumped', 'Wind', 'Solar', 'Biofuels', 'Nuclear',
                                                   'AT_net', 'BE_net', 'CH_net', 'CZ_net', 'DK1_net', 'DK2_net',
                                                 'FR_net', 'NL_net', 'NO2_net','SE4_net']][start_date:end_date]),
                             index = df_test[start_date:end_date].index)
ax = df_model_auto_pred.plot(figsize=(10,5), color = "red", label = "predicted")
df_test.DAP[start_date:end_date].plot(figsize=(10,5), color = "blue", label = "observed")
ax.set_ylabel('Electricity price , Euro / MWh')
ax.set_xlabel('Date')
plt.title("Testing electricity prices by SARIMAX model with exogenous prices, internal and external flows")
plt.show()
```

Forecasting electricity prices by SARIMAX model

In [ ]:

```
mod =sm.tsa.statespace.SARIMAX(df.DAP,
                                order=(1, 1, 2),
                                seasonal_order=(3, 0, 0, 7),
                                exog = df[['ttf', 'api2', 'CO2','Load', 'Coal', 'Gas', 'Oil',
                                                    'Hydro', 'Pumped', 'Wind', 'Solar', 'Biofuels', 'Nuclear',
                                                   'AT_net', 'BE_net', 'CH_net', 'CZ_net', 'DK1_net', 'DK2_net',
                                                'FR_net', 'NL_net', 'NO2_net','SE4_net']],
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
```

In [ ]:

```
results.summary()
```

In [ ]:

```
pred_uc_1 = results.get_forecast(steps=31, exog = exp[['ttf', 'api2', 'CO2','Load', 'Coal', 'Gas', 'Oil',
                                                     'Hydro', 'Pumped', 'Wind', 'Solar', 'Biofuels', 'Nuclear',
                                                    'AT_net', 'BE_net', 'CH_net', 'CZ_net', 'DK1_net', 'DK2_net',
                                                     'FR_net', 'NL_net', 'NO2_net','SE4_net']])
pred_ci_1 = pred_uc_1.conf_int()
ax = df_test.DAP.plot(label='observed', figsize=(10, 5))
pred_uc_1.predicted_mean.plot(ax=ax, label='forecasted')
ax.fill_between(pred_ci_1.index, pred_ci_1.iloc[:, 0],
                pred_ci_1.iloc[:, 1], color='k', alpha=.25)
ax.set_ylabel('Electricity price , Euro / MWh')
ax.set_ylabel('Date')
plt.title ('Forecasting electricity prices by SARIMAX model with exogenous prices, internal and external flows')
plt.legend()
plt.show()
```

In [ ]:

```
pred_uc_1.predicted_mean.describe()
```

Extracting residuals

In [ ]:

```
sarimax_res = results.resid
sarimax_res.describe()
```

Calculating residual returns

In [ ]:

```
sarimax_res_ret=abs(sarimax_res/df.DAP)
sarimax_res_ret = sarimax_res_ret.iloc[1:]
sarimax_res_ret.describe()
```

Plotting ACF and PACF for residual returns

In [ ]:

```
sgt.plot_acf(sarimax_res_ret, lags = 50, zero = False)
plt.title ("ACF of TTF Prices in October 2019- March 2022", size = 12)
plt.show()
```

In [ ]:

```
sgt.plot_pacf(sarimax_res_ret, lags = 50, zero = False, method = "ols")
plt.title ("PACF of electricity price residual returns by SARIMAX with exogenous prices, internal and exogenous flows"
           size = 10)
plt.show()
```

Assessing ADF-test for residual returns

In [ ]:

```
sts.adfuller(sarimax_res_ret)
```

Fitting GARCH model

In [ ]:

```
model = arch_model(sarimax_res_ret, p=7, q=0)
model_fit = model.fit()
model_fit.summary()
```

Testing GARCH model

In [ ]:

```
rolling_predictions = []
test_size = 1275

for i in range(test_size):
    train = sarimax_res_ret[:-(test_size-i)]
    model = arch_model(train, p=7, q=0)
    model_fit = model.fit(disp='off')
    pred = model_fit.forecast(horizon=1)
    rolling_predictions.append(np.sqrt(pred.variance.values[-1,:][0]))
```

In [ ]:

```
rolling_predictions = pd.Series(rolling_predictions, index=sarimax_res_ret.index[-1275:])
```

In [ ]:

```
plt.figure(figsize=(8,5))
true, = plt.plot(sarimax_res_ret[-1274:])
preds, = plt.plot(rolling_predictions)
plt.title('Testing residual returns by GARCH (7,0)', fontsize=10)
plt.legend(['true residual returns', 'predicted volatility'], fontsize=10)
```

Forecasting residual returns by GARCH model

In [ ]:

```
train = sarimax_res_ret
model = arch_model(train, p=7, q=0)
model_fit = model.fit(disp='off')
```

In [ ]:

```
pred = model_fit.forecast(horizon=30)
future_dates = [sarimax_res_5_ret.index[-1] + timedelta(days=i) for i in range(1,31)]
pred = pd.Series(np.sqrt(pred.variance.values[-1,:]), index=future_dates)
```

In [ ]:

```
plt.figure(figsize=(8,4))
plt.plot(pred)
plt.title('Forecasting residual return volatility by GARCH (7, 0) for the upcoming 30 trading days', fontsize=10)
```