In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
from tqdm import tqdm
from sklearn.metrics import mean_squared_error, r2_score
```

In [ ]:
```
# hide warning messages
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
# better plots
sns.set(rc={'figure.figsize':(12,8)});
```

In [ ]:
```
directory = os.path.dirname(os.getcwd())
directory
```

Out[ ]:
```
'd:\\github\\AssignmentEconometricsIV'
```

Question

The third question consists of an inflation forecasting exercise using a large set of monthly macroeconomic variables and nonlinear models. As in the previous question, the forecasts are based on a rolling-window framework of fixed length of 492 observations, starting in January 1959. Therefore, the forecasts start on January 1990. The last forecasts are for November 2021. More specifically, the rolling window forecasting scheme can be described as follows:

1. Run all in-sample analysis and estimation using data from observation a to observation $a + 492 - 1$.
2. Compute the forecast for observation at position $a + 492$.
3. Set $a = a + 1$ and repeat the two steps above.

In [ ]:
```
# read the data
input_path = f'{directory}\\data\\stacionarized_cpi.csv'
df = pd.read_csv(input_path)
df['date'] = pd.to_datetime(df['date'])
df = df.set_index('date')
```

In [ ]:
```
df.head()
```

Out[ ]:

	RPI	W875RX1	DPCERA3M086SBEA	RETAILx	INDPRO	IPFPNSS	IPFINAL	IPCONGD	IF
date									
1959-03-01	0.643011	0.735934	0.941009	0.832120	1.430253	0.603609	0.489927	0.000000	

date	RPI	W875RX1	DPCERA3M086SBEA	RETAILx	INDPRO	IPFPNSS	IPFINAL	IPCONGD	IF
1959-04-01	0.649412	0.704864	-0.363947	0.061571	2.107741	1.433803	1.454234	1.565338	
1959-05-01	0.576311	0.661646	1.200535	0.780340	1.495024	0.826920	0.958304	0.476849	
1959-06-01	0.310244	0.297379	0.370829	0.906434	0.114438	0.703445	0.712642	-0.476849	
1959-07-01	-0.058921	-0.076384	-0.342687	-0.033018	-2.423797	0.116693	0.824692	1.305596	

5 rows × 104 columns

In [ ]:

```
# set the amount of lags
lags = 4
```

In [ ]:

```
# create lagged variables
for col in df.columns:
    for i in range(lags):
        name_col = col + f"(-{i+1})"
        df[name_col] = df[col].shift(i+1)
```

In [ ]:

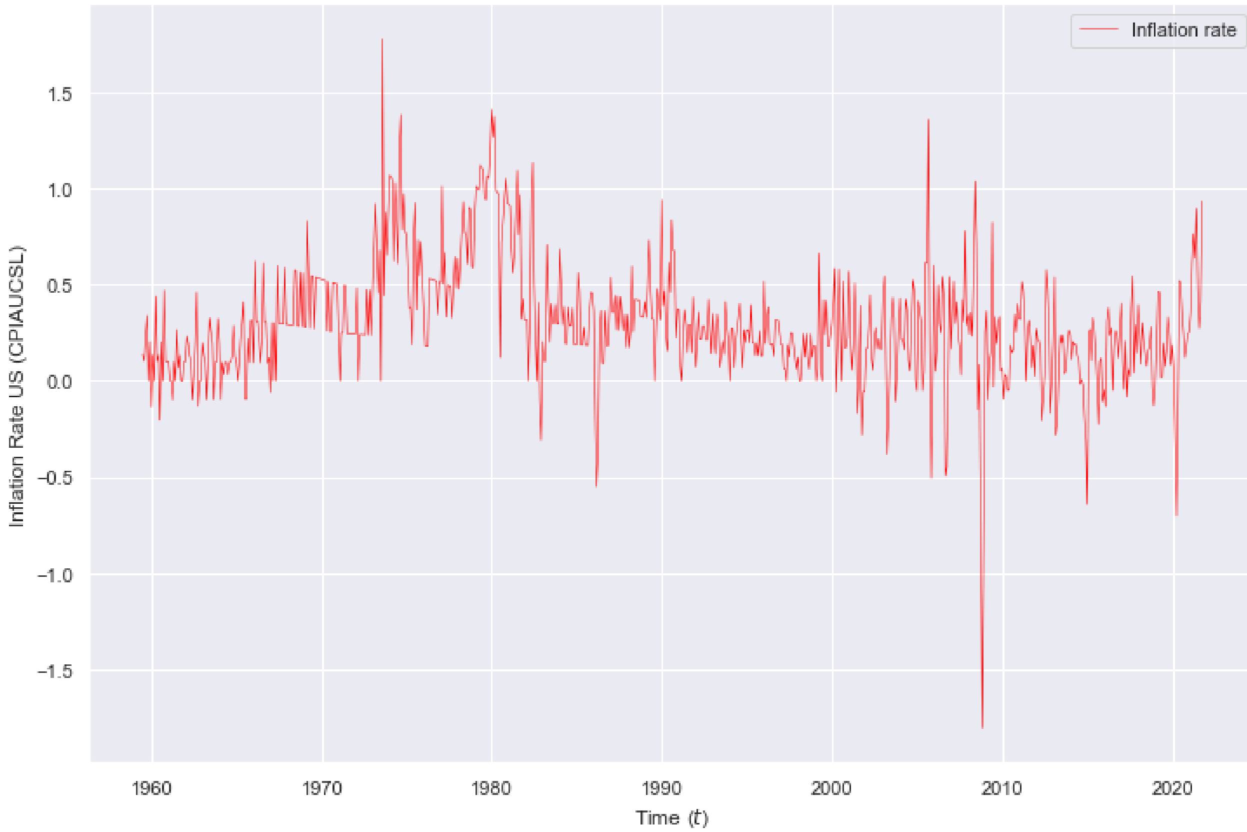
```
# inflation rate ahead
df['CPIAUCSL(+1)'] = df['CPIAUCSL'].shift(-1)
```

In [ ]:

```
# drop nan rows
df = df.dropna()
```

In [ ]:

```
# plot Inflation rate of US
plt.plot(df['CPIAUCSL'], color='red', label='Inflation rate', linewidth = 0.5)

plt.xlabel(f'Time ($t$)')
plt.ylabel('Inflation Rate US (CPIAUCSL)')
plt.legend()
plt.show()
```



In [ ]:

```
df
```

Out[ ]:

date	RPI	W875RX1	DPCERA3M086SBEA	RETAILx	INDPRO	IPFPNSS	IPFINAL	IPCONGD
1959-07-01	-0.058921	-0.076384	-0.342687	-0.033018	-2.423797	0.116693	0.824692	1.305596
1959-08-01	-0.563656	-0.574751	0.600333	0.636421	-3.446532	-0.702622	-0.234750	0.117877
1959-09-01	0.072136	0.000000	1.001845	-1.315700	-0.120914	-0.470898	-0.353804	-0.353743
1959-10-01	0.128006	0.115215	-0.683507	0.728802	-0.729019	-0.236695	-0.473264	-0.474023
1959-11-01	0.759485	0.671571	-0.044618	-2.670393	0.607511	-1.070659	-1.674509	-2.161044
...	...	...	...	...	...	...	...	...
2021-06-01	-0.268168	0.246839	0.588917	0.848804	0.546727	0.002528	0.114008	-0.202956
2021-07-01	0.803329	0.376098	-0.313442	-1.637075	0.767226	1.461057	1.668025	0.970731
2021-08-01	-0.041013	-0.094940	0.727968	1.152697	-0.136496	-0.232743	-0.485054	-0.407293
2021-09-01	-1.333562	0.169571	0.264718	0.738929	-1.018960	-0.235388	-0.539534	-0.555028

	RPI	W875RX1	DPCERA3M086SBEA	RETAILx	INDPRO	IPFPNSS	IPFINAL	IPCONGD
date								
2021-10-01	-0.226827	0.002789	0.740685	1.769110	1.664607	1.015773	1.119566	1.035674

748 rows × 521 columns

In [ ]:
```
from functions.nonlinear_models import NN_forecast, LSTM_forecast
```

In [ ]:
```
# set variables
y = df["CPIAUCSL(+1)"]
X = df.drop("CPIAUCSL(+1)", axis=1)
```

In [ ]:
```
# add forecast columns
df["CPIAUCSL_estimated_NN"] = np.nan
df["CPIAUCSL_estimated_LSTM"] = np.nan
```

(a) (60 points)

Estimate a model based on a neural network specification and compute one step ahead forecasts. You are free to choose between shallow, deep, convolution or LSTM networks. However, you have to motivate your particular choice;

feedforward Neural Network model

Cunha Medeiros, Marcelo and Schütte, Erik Christian Montes and Soussi, Tobias Skipper, Global Inflation: Implications for forecasting and monetary policy (June 24, 2022). Available at SSRN: https://ssrn.com/abstract=4145665 or http://dx.doi.org/10.2139/ssrn.4145665

Architecture:

- Optimizer: Adam (https://arxiv.org/abs/1412.6980)
- Layers: 3 (like the previous paper: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4145665)
- Activation Function: ReLu $f(x) = \max(x, 0)$
- Neurons: 100, 60 and 30 in the first, second and third layers respectively (based on https://www.sciencedirect.com/science/article/abs/pii/B978008051433850015X)

LSTM Neural Network model works as the same way, with LSTM acting like a neuron.

In [ ]:
```
# set some parameters
rolling_window = 492 - lags
T = len(y) - rolling_window
```

In [ ]:
```
# List of forecast squared errors
#errors_NN = []
```

```
errors_LSTM = []

for t in tqdm(range(T), desc='Processing for time'):
    # predict date
    date = y[[rolling_window+t]].index

    # estimation sets
    X_train = X[t:(rolling_window+t)]
    y_train = y[t:(rolling_window+t)]

    # forecast sets
    X_test = X.iloc[[rolling_window+t]]
    y_test = y[rolling_window+t]

    # estimations
    #y_pred_NN, error_pred_NN = NN_forecast(X_train, y_train, X_test, y_test)
    y_pred_LSTM, error_pred_LSTM = LSTM_forecast(X_train, y_train, X_test, y_test)

    # fill forecast columns
    #df["CPIAUCSL_estimated_NN"][date] = y_pred_NN
    df["CPIAUCSL_estimated_LSTM"][date] = y_pred_LSTM

    # append forecast squared errors
    #errors_NN.append(error_pred_NN)
    errors_LSTM.append(error_pred_LSTM)
```

```
Processing for time: 100%|██████████| 260/260 [14:31:51<00:00, 201.20s/it]
```

In [ ]:

```
df[["CPIAUCSL(+1)", "CPIAUCSL_estimated_NN", "CPIAUCSL_estimated_LSTM"]][rolling_window
```

Out[ ]:

date	CPIAUCSL(+1)	CPIAUCSL_estimated_NN	CPIAUCSL_estimated_LSTM
2000-03-01	-0.058514	0.242504	0.321759
2000-04-01	0.175234	0.124897	0.129800
2000-05-01	0.580720	0.217302	0.177507
2000-06-01	0.289519	0.527120	0.401178
2000-07-01	0.000000	0.295989	0.225920
...	...	...	...
2021-06-01	0.471599	0.368065	0.353107
2021-07-01	0.273614	-0.373261	0.011361
2021-08-01	0.410742	0.060732	0.363657
2021-09-01	0.934505	0.172199	0.187941
2021-10-01	0.773092	0.320927	0.294376

260 rows × 3 columns

In [ ]:

```
output = f'{directory}\\output\\NN_predictions.csv'
df[["CPIAUCSL", "CPIAUCSL(+1)", "CPIAUCSL_estimated_NN", "CPIAUCSL_estimated_LSTM"]].to
```

In [ ]:
```
path = f'{directory}\\output\\NN_predictions.csv'
df = pd.read_csv(path, index_col=0)
df.index = pd.to_datetime(df.index)
```

In [ ]:
```
df[rolling_window:]
```

Out[ ]:

date	CPIAUCSL	CPIAUCSL(+1)	CPIAUCSL_estimated_NN	CPIAUCSL_estimated_LSTM
2000-03-01	0.584795	-0.058514	0.242504	0.321759
2000-04-01	-0.058514	0.175234	0.124897	0.129800
2000-05-01	0.175234	0.580720	0.217302	0.177507
2000-06-01	0.580720	0.289519	0.527120	0.401178
2000-07-01	0.289519	0.000000	0.295989	0.225920
...	...	...	...	...
2021-06-01	0.896742	0.471599	0.368065	0.353107
2021-07-01	0.471599	0.273614	-0.373261	0.011361
2021-08-01	0.273614	0.410742	0.060732	0.363657
2021-09-01	0.410742	0.934505	0.172199	0.187941
2021-10-01	0.934505	0.773092	0.320927	0.294376

260 rows × 4 columns

In [ ]:
```
# prediction
prediction_dates = list(X.iloc[(rolling_window):].index)
```

In [ ]:
```
# treat the error list
errors_NN_ = [item[0] for item in errors_NN]
errors_LSTM_ = [item[0] for item in errors_LSTM]

# compute cumulated mse
cum_errors_NN = list(np.cumsum(errors_NN_))
cum_errors_LSTM = list(np.cumsum(errors_LSTM_))
```

In [ ]:
```
errors = {'Cumulated_MSE_NN': cum_errors_NN, 'Cumulated_MSE_LSTM': cum_errors_LSTM}
mse = pd.DataFrame(errors, index=prediction_dates)
```

In [ ]:
```
mse
```

Out[ ]:

	Cumulated_MSE_NN	Cumulated_MSE_LSTM
2000-03-01	0.090612	0.144607

	Cumulated_MSE_NN	Cumulated_MSE_LSTM
2000-04-01	0.093146	0.146672
2000-05-01	0.225218	0.309253
2000-06-01	0.281672	0.321720
2000-07-01	0.369282	0.372760
...	...	...
2021-06-01	25.398293	20.114305
2021-07-01	25.816740	20.183083
2021-08-01	25.939247	20.185299
2021-09-01	26.520357	20.742657
2021-10-01	26.724810	20.971827

260 rows × 2 columns

In [ ]:

```
output = f'{directory}\\output\\NN_MSE.csv'
mse.to_csv(output, sep=',', encoding='utf-8')
```

In [ ]:

```
path = f'{directory}\\output\\NN_MSE.csv'
mse = pd.read_csv(path, index_col=0)
mse.index = pd.to_datetime(mse.index)
```

In [ ]:

```
mse
```

Out[ ]:

	Cumulated_MSE_NN	Cumulated_MSE_LSTM
2000-03-01	0.090612	0.144607
2000-04-01	0.093146	0.146672
2000-05-01	0.225218	0.309252
2000-06-01	0.281672	0.321720
2000-07-01	0.369282	0.372760
...	...	...
2021-06-01	25.398293	20.114305
2021-07-01	25.816740	20.183083
2021-08-01	25.939247	20.185299
2021-09-01	26.520357	20.742657
2021-10-01	26.724810	20.971827

260 rows × 2 columns

(b) (20 points)

Plot you forecasts against the four linear benchmarks from Question 2. Comment on you results.

In [ ]:

```
path = '../output/forecasts.csv'
forecast_ln = pd.read_csv(path, index_col=0)
prediction_dates.append(pd.to_datetime('2021-11-01'))
forecast_ln.index = prediction_dates
```

In [ ]:

```
forecast_ln
```

Out[ ]:

	forecasts.forecasts_Q2.AR	forecasts.forecasts_Q2.AR_PC	forecasts.forecasts_Q2.Ridge_4lags	forecast
2000-03-01	0.347137	0.306083	0.358069	
2000-04-01	0.424690	0.401117	0.359949	
2000-05-01	0.184807	0.128173	0.357127	
2000-06-01	0.250046	0.286717	0.358928	
2000-07-01	0.416175	0.423749	0.361864	
...	...	...	...	
2021-07-01	0.404299	0.519179	0.239185	
2021-08-01	0.190942	0.086930	0.233436	
2021-09-01	0.205723	0.207572	0.232049	
2021-10-01	0.365649	0.260807	0.233505	
2021-11-01	0.602073	0.553538	0.237903	

261 rows × 7 columns

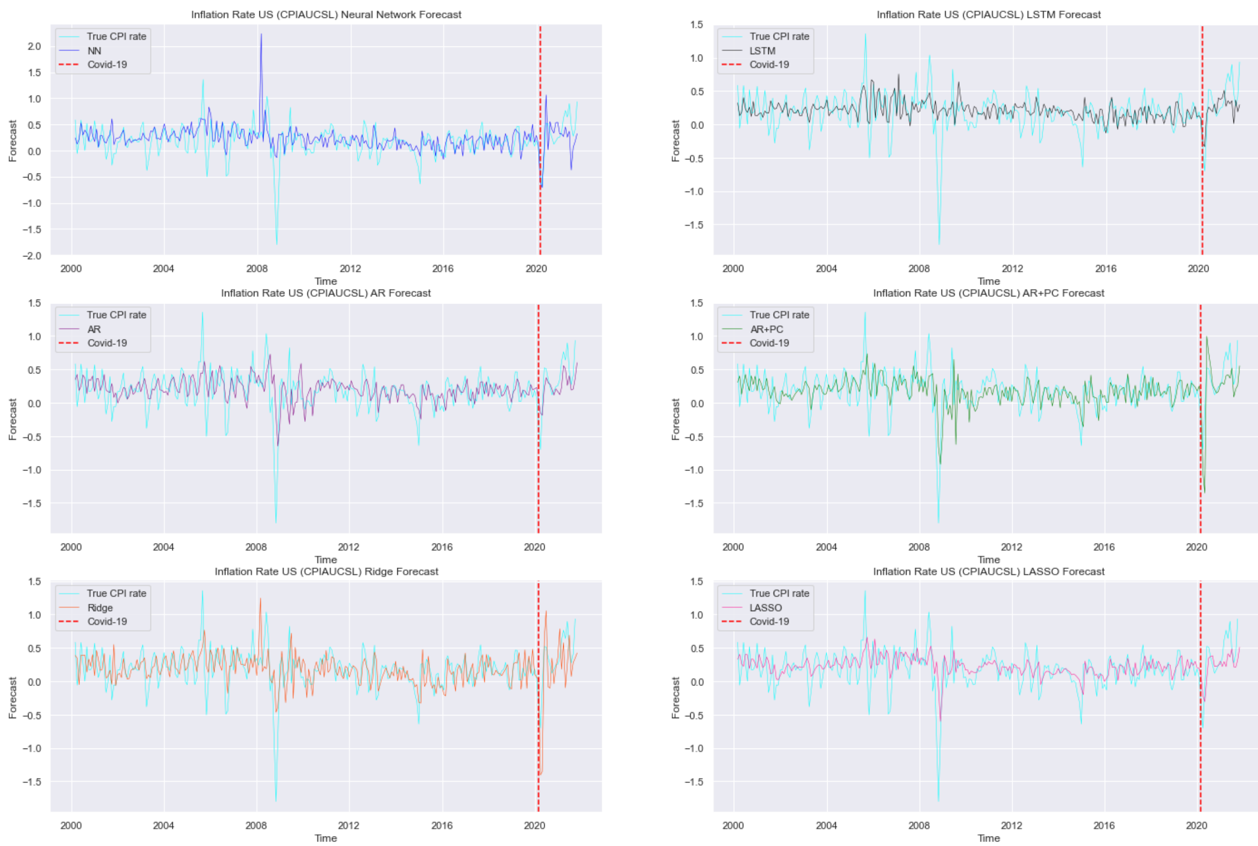
In [ ]:

```
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(24, 16))

# True CPI
axes[0,0].plot(df['CPIAUCSL'][rolling_window:], color='cyan', label='True CPI rate', li
# NN forecasts
axes[0,0].plot(df["CPIAUCSL_estimated_NN"], color='blue', label='NN', linewidth=0.5)
# adding a vertical line at 2020, January
axes[0,0].axvline(x=mse.index[240], color='red', linestyle='--', label='Covid-19')
# labels
```

```
axes[0,0].set_xlabel('Time')
axes[0,0].set_ylabel('Forecast')
axes[0,0].set_title('Inflation Rate US (CPIAUCSL) Neural Network Forecast')
axes[0,0].legend()

# True CPI
axes[0,1].plot(df['CPIAUCSL'][rolling_window:], color='cyan', label='True CPI rate', li
# LSTM forecasts
axes[0,1].plot(df["CPIAUCSL_estimated_LSTM"], color='black', label='LSTM', linewidth=0.
# adding a vertical line at 2020, January
axes[0,1].axvline(x=mse.index[240], color='red', linestyle='--', label='Covid-19')
# labels
axes[0,1].set_xlabel('Time')
axes[0,1].set_ylabel('Forecast')
axes[0,1].set_title('Inflation Rate US (CPIAUCSL) LSTM Forecast')
axes[0,1].legend()

# True CPI
axes[1,0].plot(df['CPIAUCSL'][rolling_window:], color='cyan', label='True CPI rate', li
# AR
axes[1,0].plot(forecast_ln["forecasts.forecasts_Q2.AR"], color='purple', label='AR', li
# adding a vertical line at 2020, January
axes[1,0].axvline(x=mse.index[240], color='red', linestyle='--', label='Covid-19')
# labels
axes[1,0].set_xlabel('Time')
axes[1,0].set_ylabel('Forecast')
axes[1,0].set_title('Inflation Rate US (CPIAUCSL) AR Forecast')
axes[1,0].legend()

# True CPI
axes[1,1].plot(df['CPIAUCSL'][rolling_window:], color='cyan', label='True CPI rate', li
# AR+PC
axes[1,1].plot(forecast_ln["forecasts.forecasts_Q2.AR_PC"], color='green', label='AR+PC
# adding a vertical line at 2020, January
axes[1,1].axvline(x=mse.index[240], color='red', linestyle='--', label='Covid-19')
# labels
axes[1,1].set_xlabel('Time')
axes[1,1].set_ylabel('Forecast')
axes[1,1].set_title('Inflation Rate US (CPIAUCSL) AR+PC Forecast')
axes[1,1].legend()

# True CPI
axes[2,0].plot(df['CPIAUCSL'][rolling_window:], color='cyan', label='True CPI rate', li
# Ridge
axes[2,0].plot(forecast_ln["forecasts.forecasts_Q2.Ridge"], color='orangered', label='R
# adding a vertical line at 2020, January
axes[2,0].axvline(x=mse.index[240], color='red', linestyle='--', label='Covid-19')
# labels
axes[2,0].set_xlabel('Time')
axes[2,0].set_ylabel('Forecast')
axes[2,0].set_title('Inflation Rate US (CPIAUCSL) Ridge Forecast')
axes[2,0].legend()

# True CPI
axes[2,1].plot(df['CPIAUCSL'][rolling_window:], color='cyan', label='True CPI rate', li
# LASSO
axes[2,1].plot(forecast_ln["forecasts.forecasts_Q2.LASSO"], color='deeppink', label='LA
# adding a vertical line at 2020, January
axes[2,1].axvline(x=mse.index[240], color='red', linestyle='--', label='Covid-19')
# labels
```

```
axes[2,1].set_xlabel('Time')
axes[2,1].set_ylabel('Forecast')
axes[2,1].set_title('Inflation Rate US (CPIAUCSL) LASSO Forecast')
axes[2,1].legend()
```

Out[ ]: <matplotlib.legend.Legend at 0x274f8d7bb50>



In [ ]:

```
plt.plot(df['CPIAUCSL'][rolling_window:], color='cyan', label='True CPI rate', linewidt

plt.plot(df["CPIAUCSL_estimated_NN"], color='blue', label='NN', linewidth = 0.5)

plt.plot(df["CPIAUCSL_estimated_LSTM"], color='black', label='LSTM', linewidth = 0.5)

# adding a vertical line at 2020, January
plt.axvline(x=mse.index[240], color='red', linestyle='--', label='Covid-19')

plt.xlabel('Time')
plt.ylabel('Inflation Rate US (CPIAUCSL) Forecasts')
plt.legend()
plt.show()
```

(c) (20 points)

Compute the mean squared error of the NN-based model and the benchmarks. Does the NN model outperform the linear alternative?

In [ ]:

```
path = '../output/cum_error.csv'
cum_mse = pd.read_csv(path, index_col=0)
cum_mse["date"] = pd.to_datetime(cum_mse["date"])
cum_mse.set_index("date", inplace=True)
```

In [ ]:

```
cum_mse
```

Out[ ]:

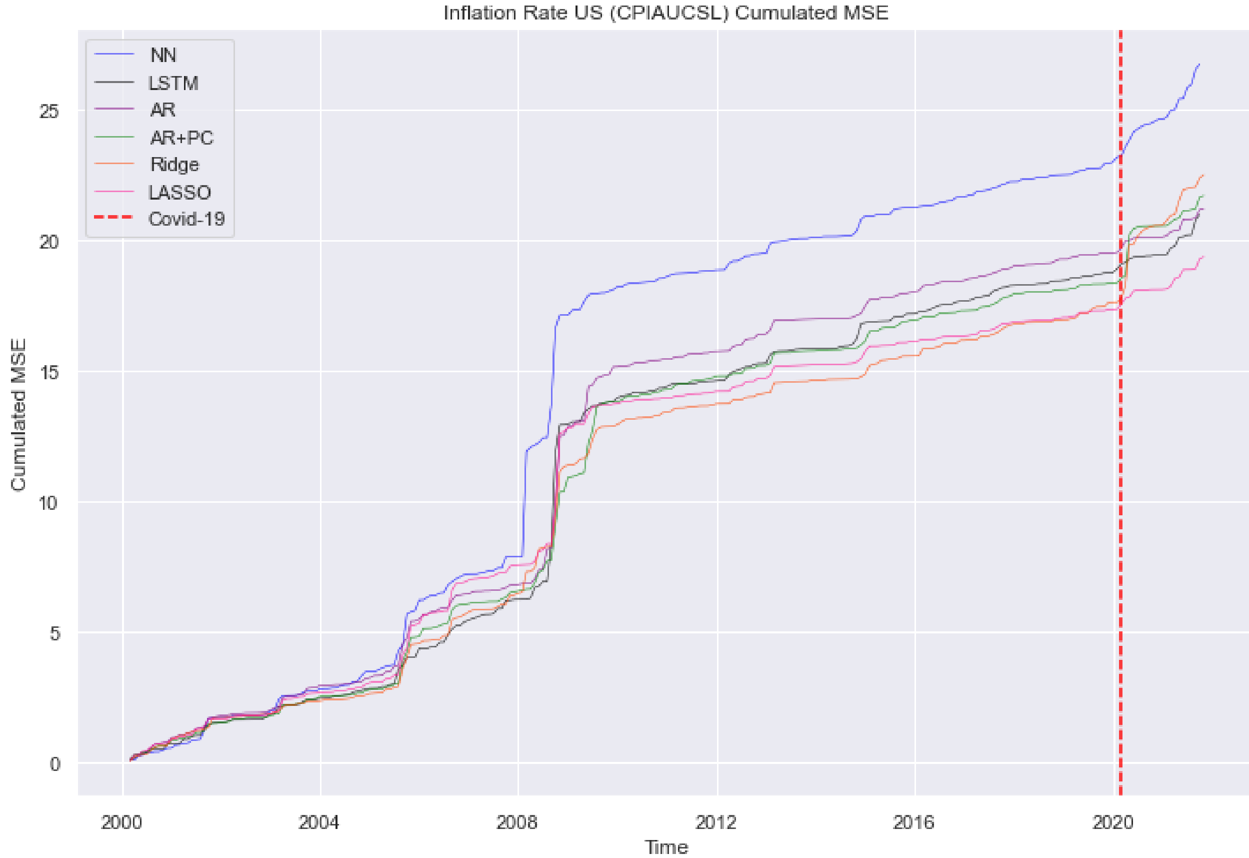
date	AR	AR+PC	Ridge (4 lags)	Ridge	LASSO	RF (4 lags)	RF
2000-03-01	0.056481	0.077681	0.051405	0.039072	0.065451	0.087837	0.086347
2000-04-01	0.289967	0.288942	0.226516	0.202956	0.278926	0.309426	0.268405
2000-05-01	0.290059	0.291156	0.259602	0.254299	0.285988	0.313885	0.269424
2000-06-01	0.399404	0.377594	0.308793	0.290743	0.413642	0.416084	0.330281
2000-07-01	0.415446	0.395612	0.314027	0.299689	0.426557	0.417525	0.341706
...	...	...	...	...	...	...	...
2021-07-01	20.777157	21.104799	29.034805	21.953238	18.848510	19.511802	18.618540
2021-08-01	20.783992	21.139650	29.036419	21.994010	18.851793	19.551800	18.643675

	AR	AR+PC	Ridge (4 lags)	Ridge	LASSO	RF (4 lags)	RF
date							
2021-09-01	20.826024	21.180928	29.068351	22.011110	18.892047	19.590289	18.671658
2021-10-01	21.149622	21.634798	29.559752	22.359695	19.267982	20.023417	19.094611
2021-11-01	21.178869	21.683002	29.846179	22.482222	19.337216	20.144240	19.209014

261 rows × 7 columns

In [ ]:

```
# plot of cumulated MSE
plt.plot(mse["Cumulated_MSE_NN"], color='blue', label="NN", linewidth = 0.5)

# plot of cumulated MSE
plt.plot(mse["Cumulated_MSE_LSTM"], color='black', label="LSTM", linewidth = 0.5)

# plot of cumulated MSE
plt.plot(cum_mse["AR"], color='purple', label="AR", linewidth = 0.5)

# plot of cumulated MSE
plt.plot(cum_mse["AR+PC"], color='green', label="AR+PC", linewidth = 0.5)

# plot of cumulated MSE
plt.plot(cum_mse["Ridge"], color='orangered', label="Ridge", linewidth = 0.5)

# plot of cumulated MSE
plt.plot(cum_mse["LASSO"], color='deeppink', label="LASSO", linewidth = 0.5)

# adding a vertical line at 2020, January
plt.axvline(x=mse.index[240], color='red', linestyle='--', label='Covid-19')

plt.xlabel('Time')
plt.ylabel('Cumulated MSE')
plt.title('Inflation Rate US (CPIAUCSL) Cumulated MSE')
plt.legend()
plt.show()
```


Inflation Rate US (CPIAUCSL) Cumulated MSE
NN
LSTM
AR
AR+PC
Ridge
LASSO
Covid-19
Cumulated MSE
0
5
10
15
20
25
2000
2004
2008
2012
2016
2020
Time