US Macro Data Forecasting Report

[View on TensorFlow.org] [Run in Google Colab] [View source on GitHub] [Download notebook]

This is a report on analyzing and forecasting the US macro data using **Recurrent Neural N**
Convolutional Neural Network (CNNs) and **Generative Adversarial Net**
report is:

Part I. Statistical analysis

- Basic manipulation
- Correlation analysis
- Time series analysis with `ARIMA`

Part II. Deep learning models

- Basic model: single-step, single-feature forecasting with LSTM
- Generalized model: multi-step, multi-feature forcasting with LSTM
- Advanced model: Generative Adversarial Network (GAN) with RNN and CNN.

Part III. Conclusions and Next steps

- Conclusions
- Next steps

Introduction

1. The Notebook

Follow the notebook, we can recreate all the results, notice that

- Upload the `USMacroData.xls` file to the root folder on google colab.
- To navigate better, use the table of contents bottom on the upper-left sidebar.
- **For clarity, all code cells are hiden, double click on the cell to get the**
- Change the parameters as indicated in the comments to create more custom outputs.
- All source code can also be found in the project file folder

2. The US Macro dataset

This report uses a US Macro Dataset provided by the ADP.

Before analyzing the data with codes, we have the following observations.

- This dataset contains **6** different features (the **Inflation, Wage, Unemployment,**
  InterstRate) about the macro economy of the US.

- Data were collected every 1 month, beginning in $\mathbf{1965\text{-}01\text{-}01 \text{ to } 2015\text{-}12\text{-}01}$.
- In total, we have $\mathbf{612 \text{ rows (month)}}$ and $\mathbf{6 \text{ columns (features)}}$.

Part I.1 Basic manipulation

Code and examples

```
basic.py
```

read the file and show the head

	Month	Inflation	Wage	Unemployment	Consumption	Investment	InterestRat
0	1965-01-01	1.557632	3.200000	4.9	6.972061	12.3	3.9
1	1965-02-01	1.557632	3.600000	5.1	7.811330	13.2	3.9
2	1965-03-01	1.242236	4.000000	4.7	7.828032	18.7	4.0
3	1965-04-01	1.552795	3.585657	4.8	8.477938	9.8	4.0
4	1965-05-01	1.552795	3.968254	4.6	7.139364	10.2	4.1

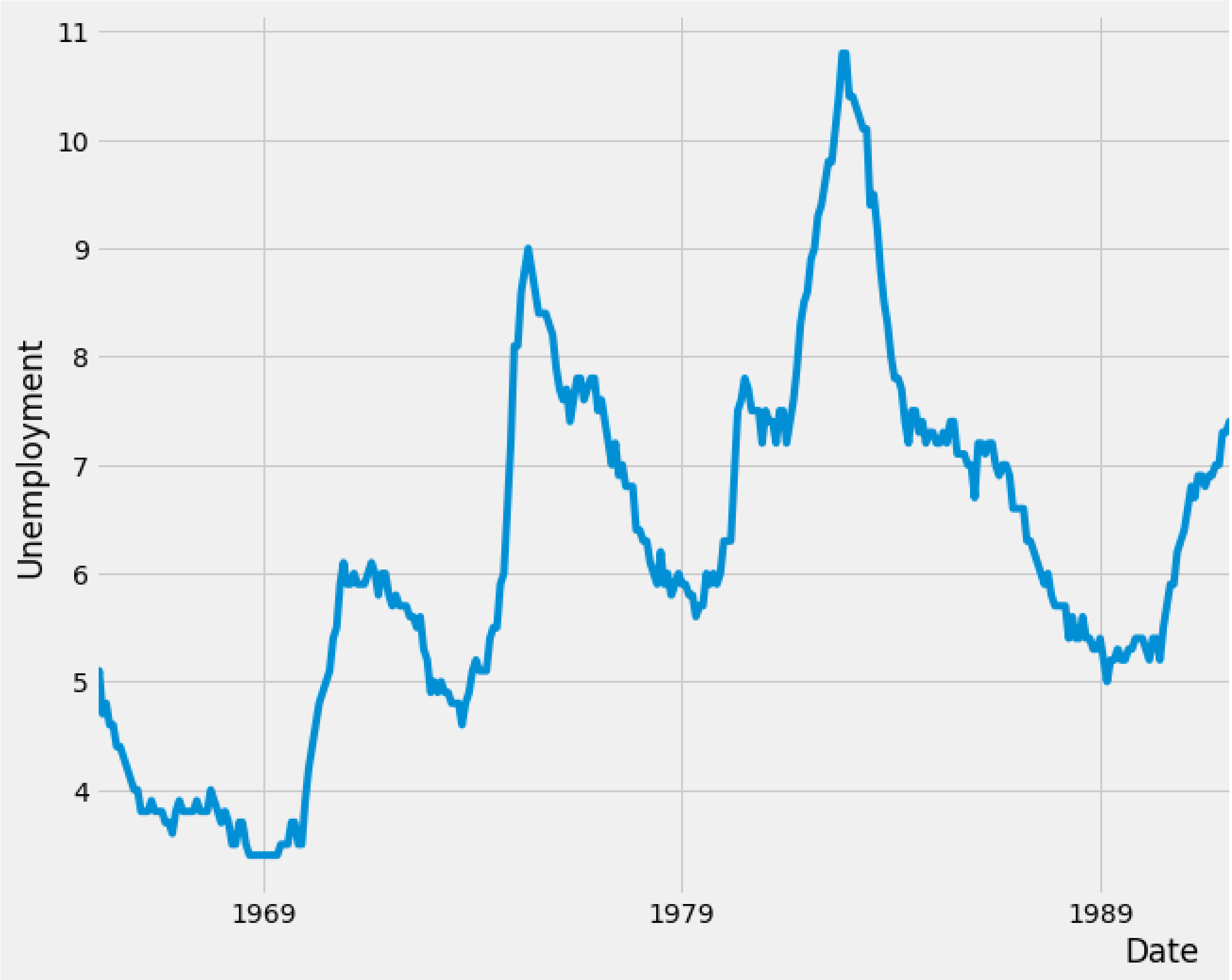
Basic checks: find null values and fill, set index, etc.

```
Null values summary:

Inflation       0
Wage            0
Unemployment    0
Consumption     0
Investment      0
InterestRate    0
dtype: int64
```

	Inflation	Wage	Unemployment	Consumption	Investment	InterestRate
Month						
1965-01-01	1.557632	3.200000	4.9	6.972061	12.3	3.90
1965-02-01	1.557632	3.600000	5.1	7.811330	13.2	3.98
1965-03-01	1.242236	4.000000	4.7	7.828032	18.7	4.04
1965-04-01	1.552795	3.585657	4.8	8.477938	9.8	4.09
1965-05-01	1.552795	3.968254	4.6	7.139364	10.2	4.10

Example: plot the "Inflation" column



Data Analysis

As a high level overview, some distinguishable patterns appear when we plot the data:

- **In the 80's (1979-1989), all features experienced some drastic chang**
- The time-series has **seasonality pattern**, for example, **Unemployment** has **long** goes through 1 or 2 major up and downs. We will examine the seasonality more carefully in

Part I.2 Correlation analysis

Though it's indicated that there's no obvious correlation among the 6 features, we compute severa **Naive correlation, Pearson correlation, local Pearson correlation, instan** and related statistics in order to

- Test the validity of the assumption (i.e. no two features are apprantly correlated).
- Chose source and target features for later model builds.

By doing so, we can get more understanding about the 'quality' and 'inner relations' of the data. If explanatory power to the feature that we want to predict (e.g. "Inflation"), then there is no need for learning models. On the other hand, if one feature has higher-than-random correlations to another the feature and the other as the target. In this case, **to determine which feature leads,** the **Dynamic time wrapping**.

Code and Examples

correlation.py

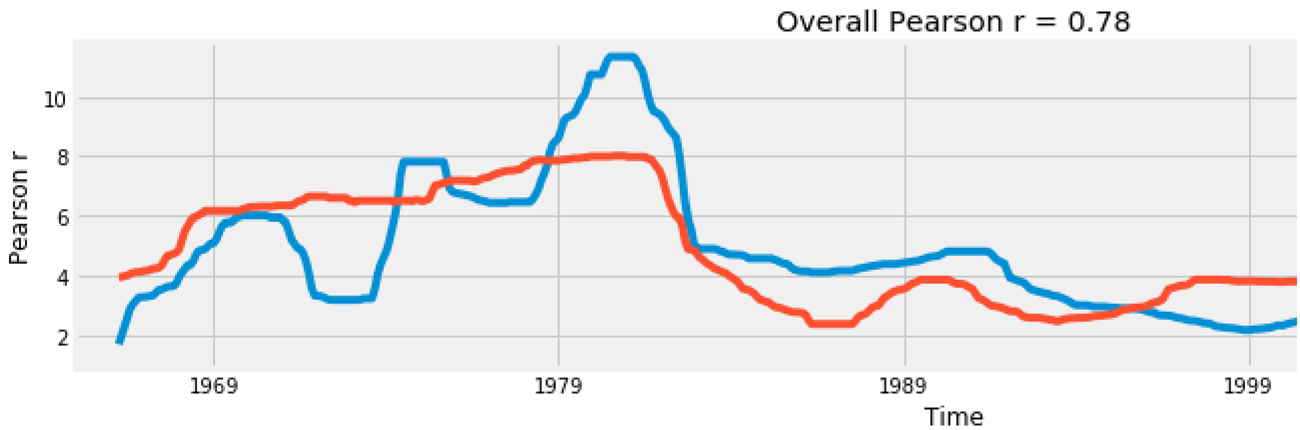
```
Requirement already satisfied: dtw in /usr/local/lib/python3.6/dist-packages (1.4.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from
```

Example: Naive correlation.

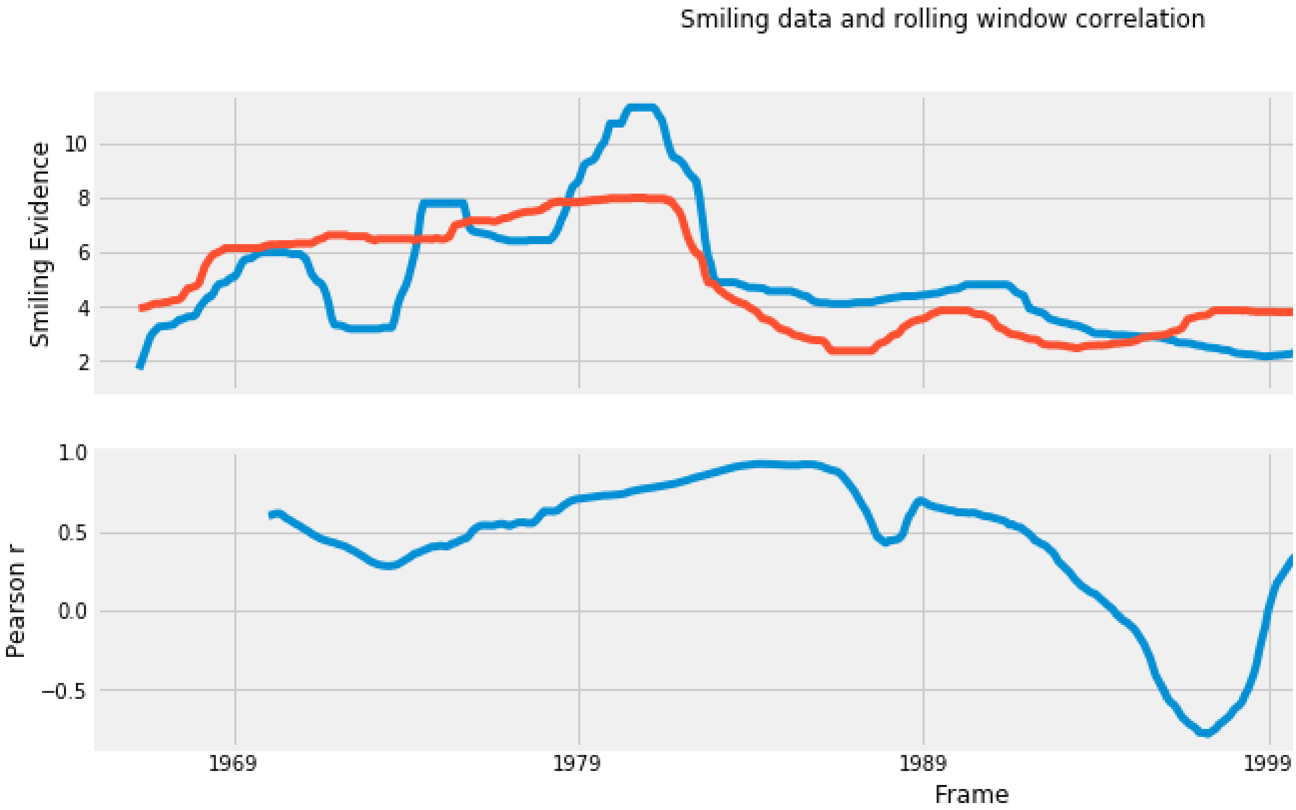
	Inflation	Wage	Unemployment	Consumption	Investment	InterestR
Inflation	1.000000	0.778155	0.191886	0.617820	-0.341421	0.773
Wage	0.778155	1.000000	-0.068529	0.703745	-0.125412	0.647
Unemployment	0.191886	-0.068529	1.000000	-0.097183	-0.038286	-0.027
Consumption	0.617820	0.703745	-0.097183	1.000000	0.203165	0.655
Investment	-0.341421	-0.125412	-0.038286	0.203165	1.000000	-0.234
InterestRate	0.773616	0.647482	-0.027809	0.655305	-0.234573	1.000

Example: Pearson correlation

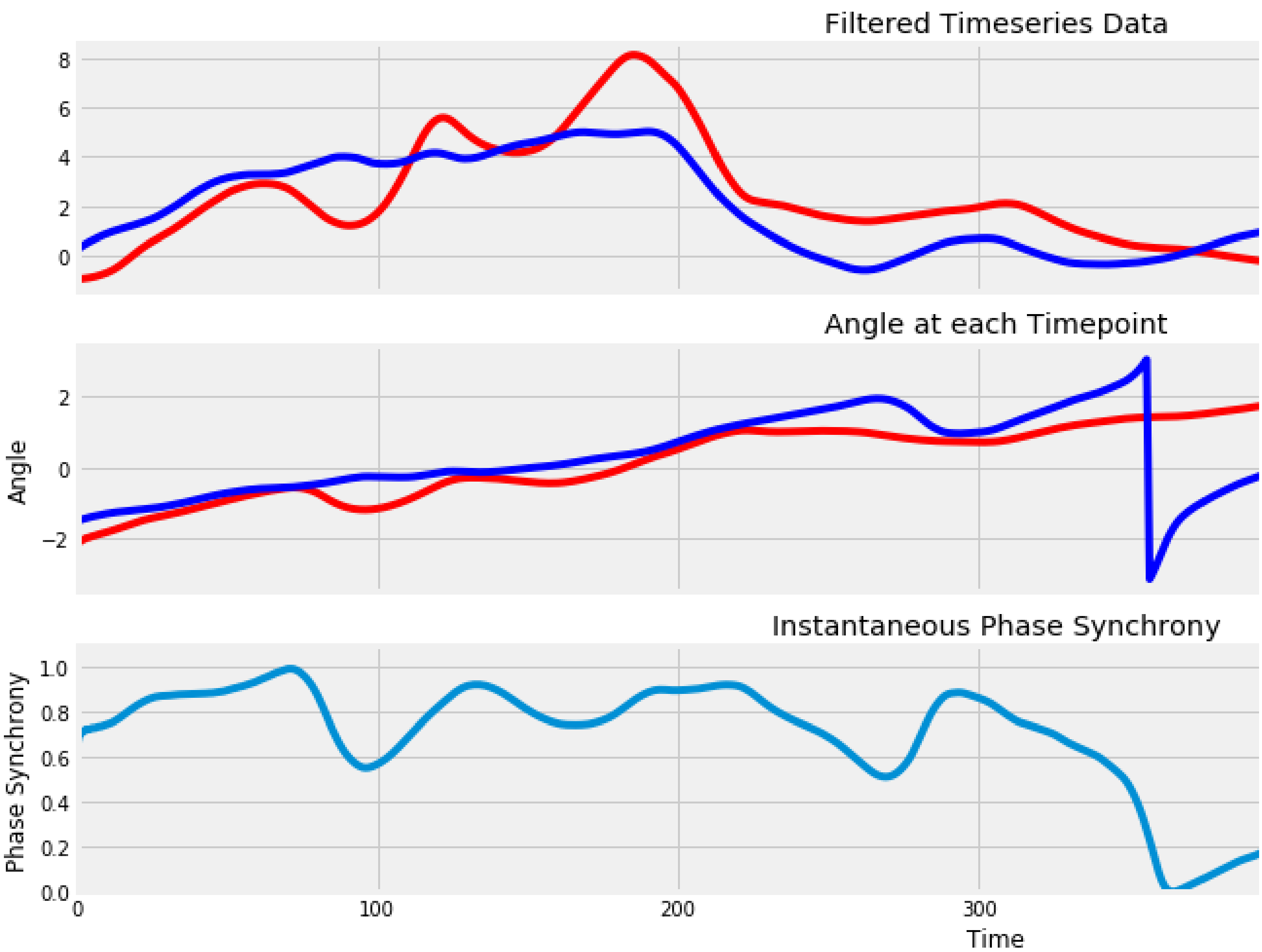
```
Pandas computed Pearson r: 0.7781551675438367
Scipy computed Pearson r: 0.7781551675438365 and p-value: 2.531376149037 59e-125
```



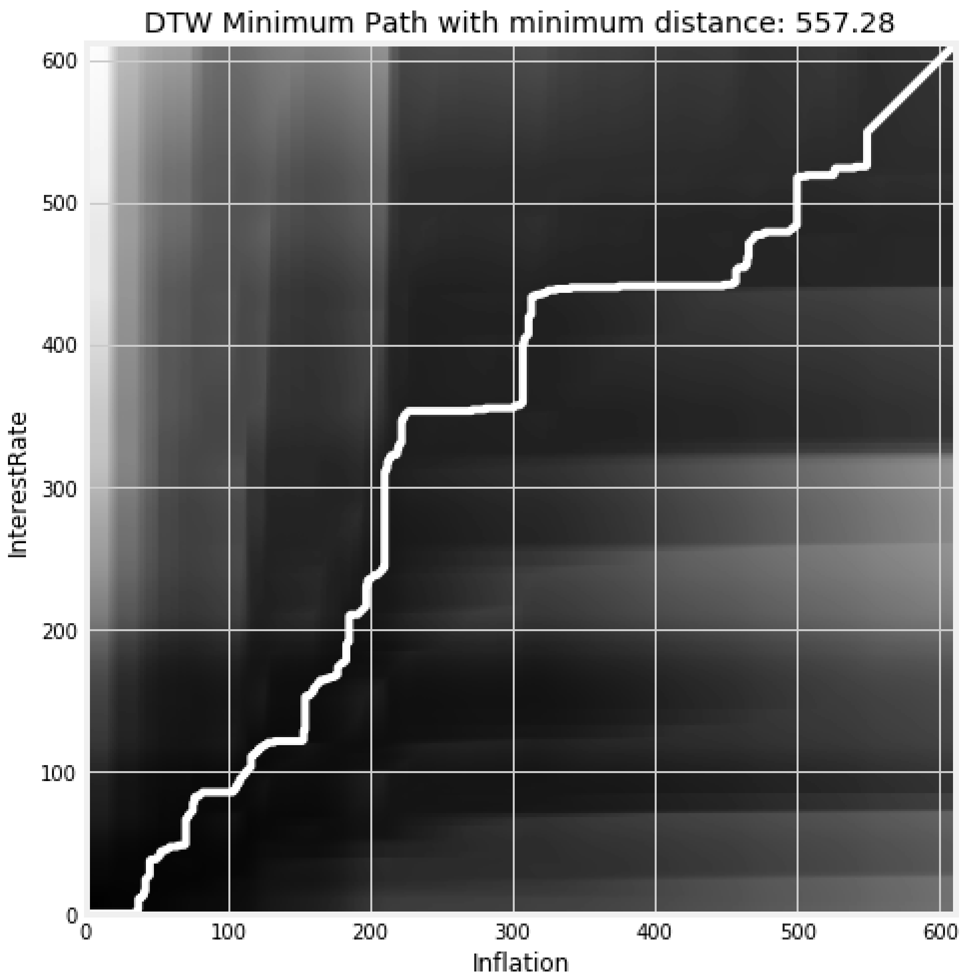
Example: local Pearson correlation



Example: instantaneous phase synchronization



Example: dynamic time wraping



Data analysis

Inspecting the correlations from different angles, we find

- **Inflation and Wage have the highest correlation, 0.778155,** among all th
- Inflation, Wage, Consumption and IntestRate show quite high positive correlation, and low n Unemployment and Investment.
- **Most features slightly leads the Inflation feature**.
- For the first 30 years, certain feature pairs show **high instantaneous phase synchr**

We conclude that

- **The assumption that no two features have apparent correlation is w**
- It's reasonable to **use Inflation as target and the other 5 features as source for forcastin**
.

Part I.3 Time series analysis with ARIMA

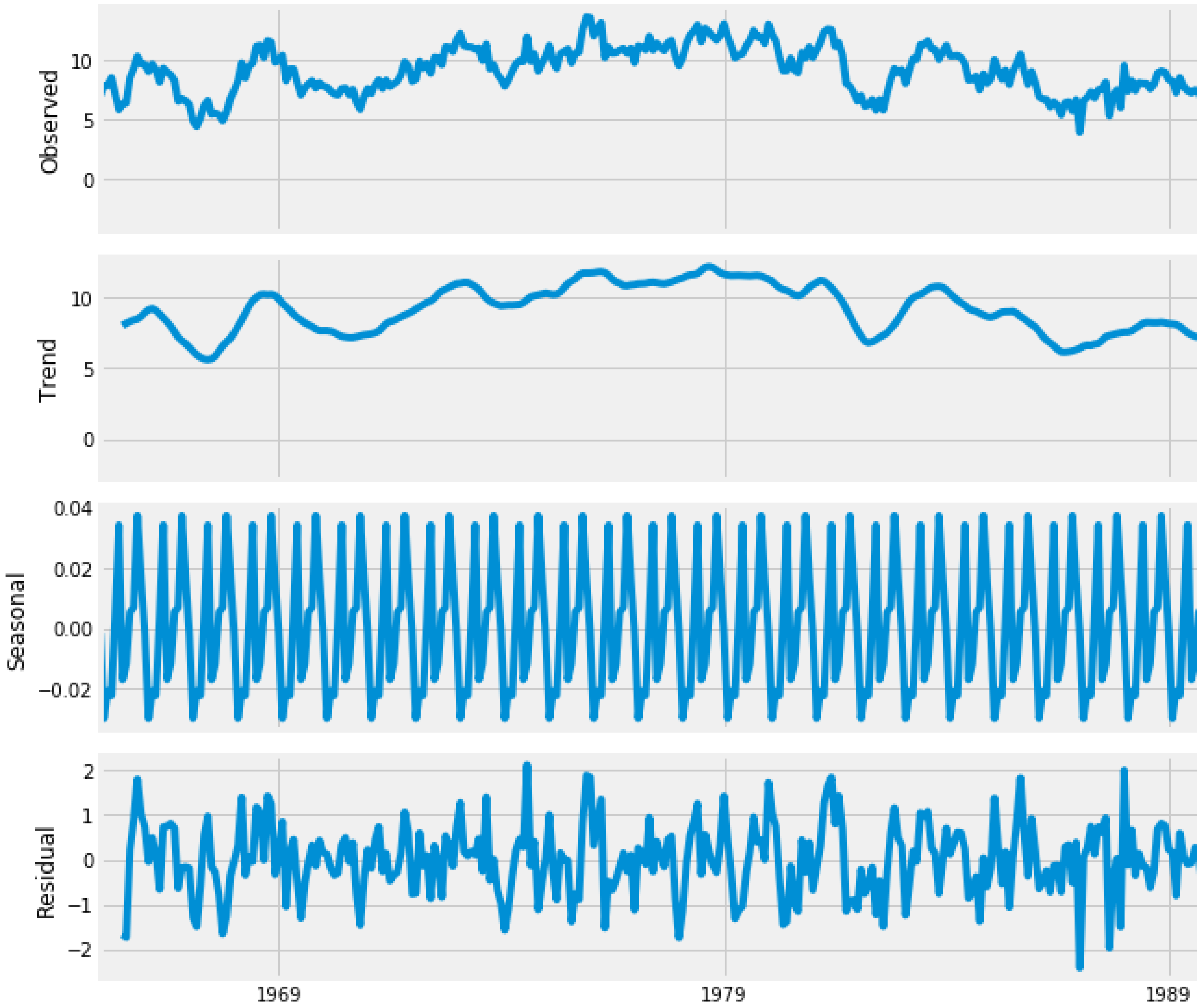
As we mentioned above, some remarkable patterns (e.g. seasonality pattern) naturally appear in c

- We visualize our data using **time-series decomposition** that allows us to decompos trend, seasonality, and noise.
- We **train an ARIMA (Autoregressive Integrated Moving Average) m** Inflation values. To get optimal output, we first
- Use **grid search** to get the optimal parameters for the ARIMA mode.
- We use **ARIMA diagnostics** to investigate any unusual behavior.

Code and examples

`time_series.py`

Example: decompose "Consumption" column into trend, seasonal and residua

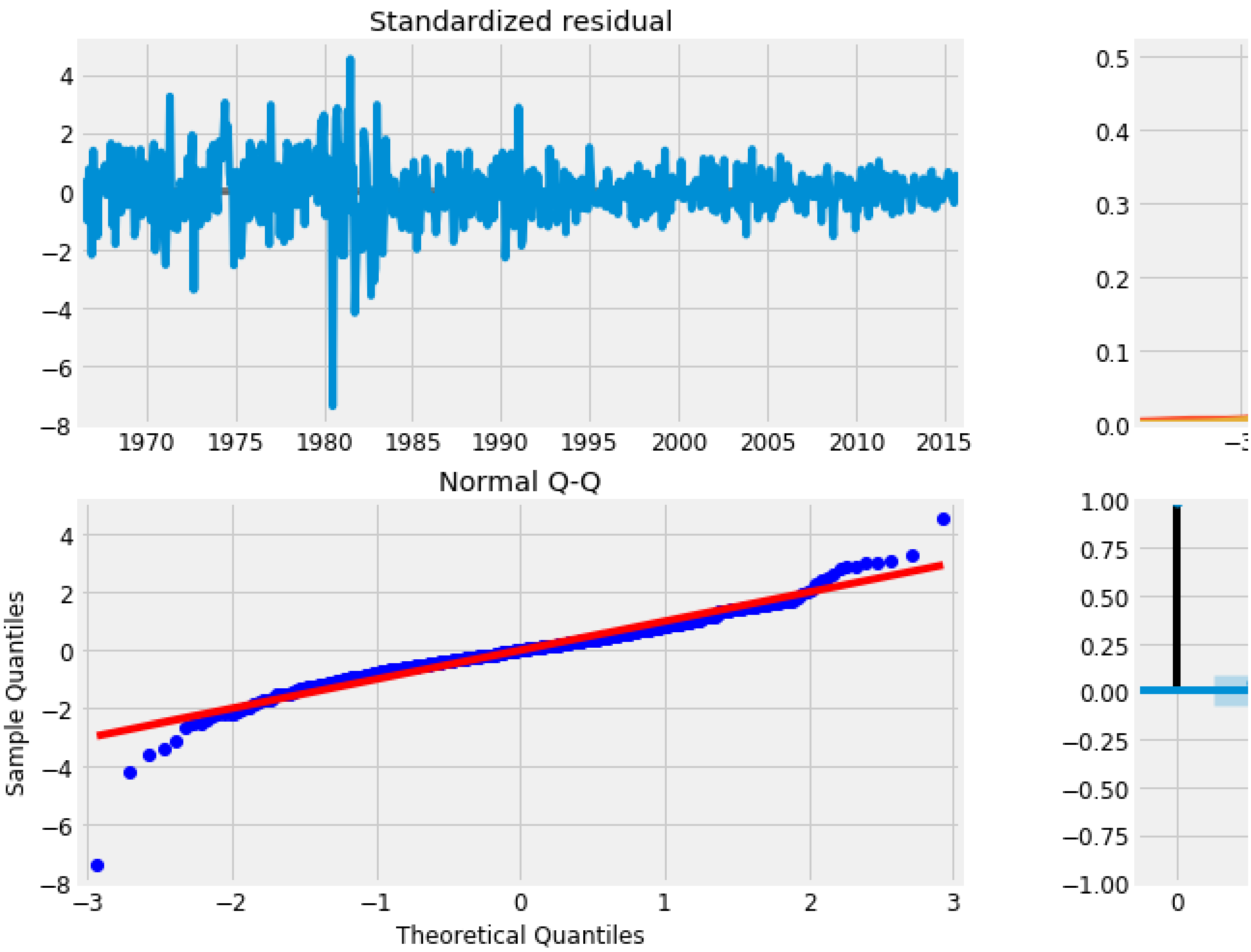


Time series analysis with `ARIMA`

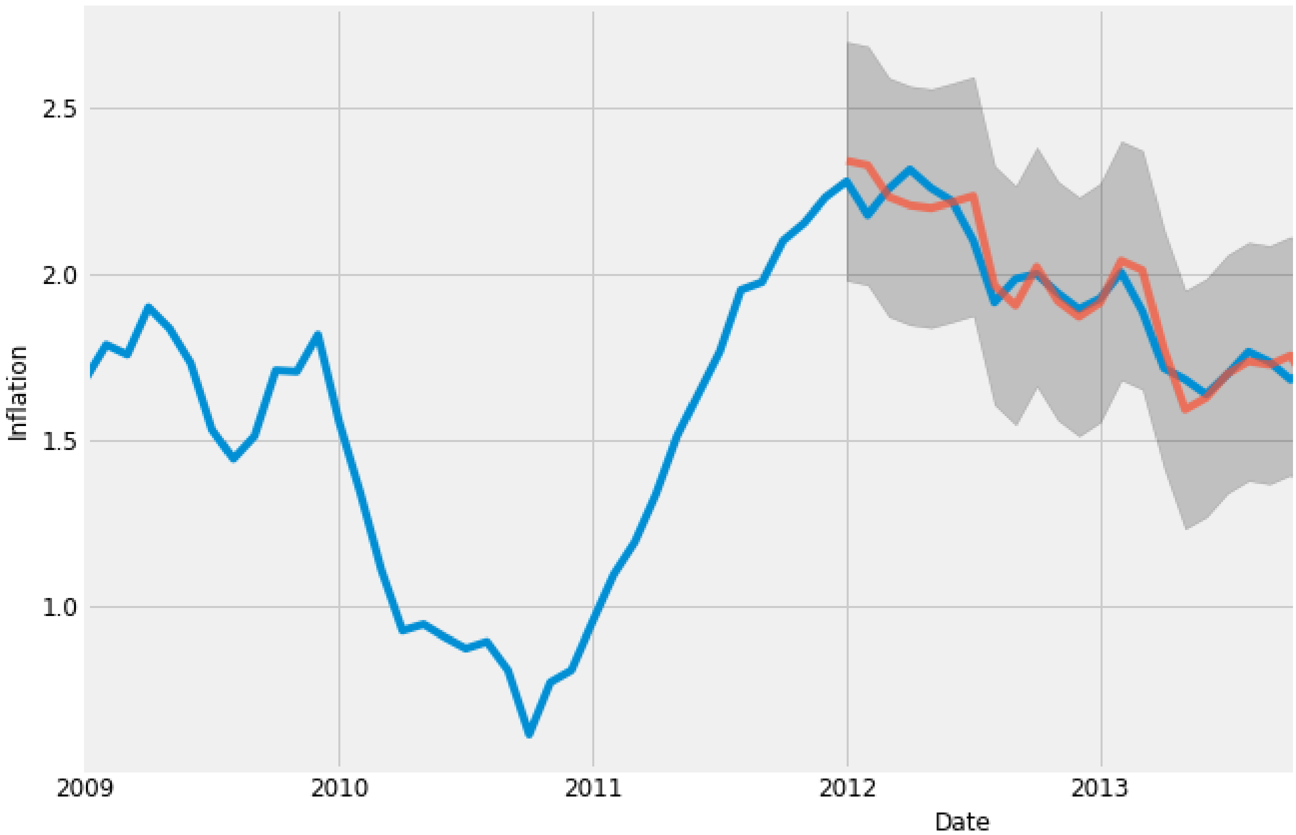
Grid search for optimal `ARIMA` parameters

`ARIMA` training

`ARIMA` diadonostics



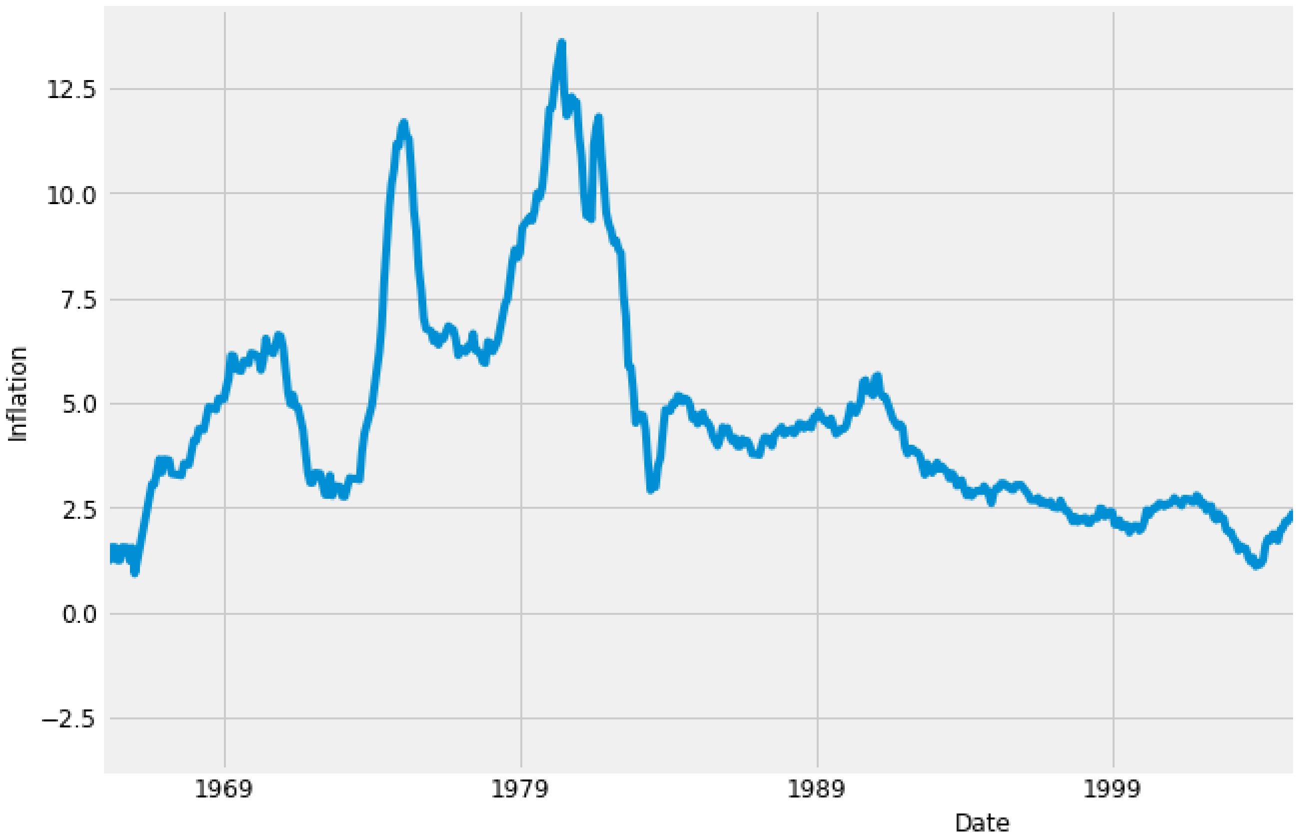
`ARIMA` predictions




```
The Mean Squared Error of our forecasts is 0.004963826636415743

The Root Mean Squared Error of our forecasts is 0.07
```

ARIMA forcasts



Data Analysis

- Components plot show the obvious seasonality, for example, in every 10 years, the "**Inflati**
a **half-year seasonality**.
- The optimal ARIMA parameters for "Inflation" are `(1, 1, 1)x(0, 0, 1, 12)`
- The ARIMA diagonostics show that the **noise distribution is narrower than the**
- **The one-step ahead forcast captures the overall trend well**.
- As we forecast further out into the future, we becomes less confident in our values. This is r
by our model, which grow larger as we move further out into the future.

Part II.1 Basic model: single-step, single-feature fo

Recurrent Neural Networks (RNNs) are good fits for time-series analysis because F
designed to capture patterns developing through time.

However, vanilla RNNs have a major disadvantage---the vanishing gradient problem---"the changes
so small, making the network unable to converge to a optimal solution.

LSTM (Long-Short Term Memory) is a variation of vanilla RNNS,it overcomes the va
problem by clipping gradients if they exceed some constant bounds.

In this section, we will

- Process the data to fit the LSTM model
- **Build and train the LSTM model for single-step, single-feature pred**
tomorrow value with only today's values of the other 5 features).

imports

Data preparation

Build and train the LSTM model

Make sure data forms are correct

```
(427, 1, 5)
(427, 1)
(184, 1, 5)
(184, 1)
```

LSTM with SGD, RMSprop, Adam optimizers, epochs = 100

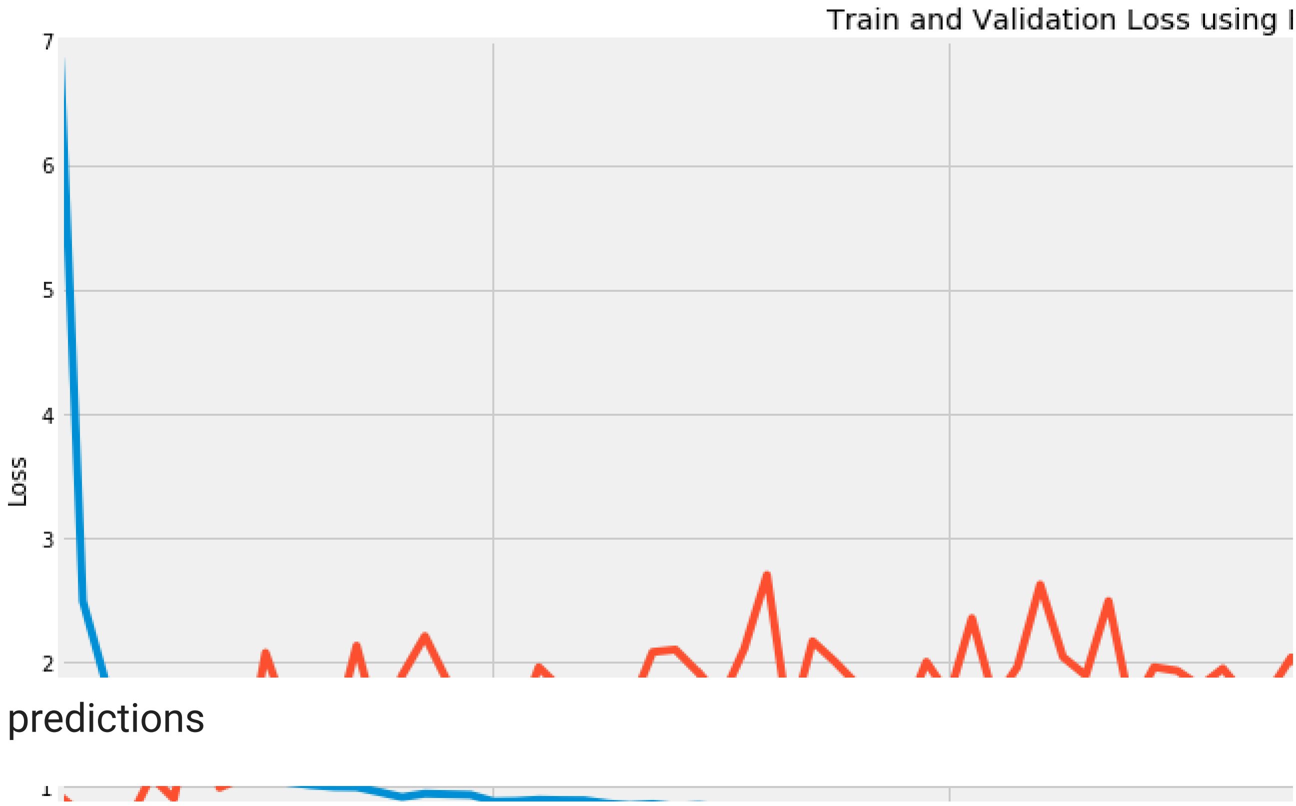
```
Epoch 27/100
427/427 [==============================] - 1s 3ms/step - loss: 0.8588 - acc: 0.0000e+
Epoch 28/100
427/427 [==============================] - 1s 3ms/step - loss: 0.8365 - acc: 0.0000e+
Epoch 29/100
427/427 [==============================] - 1s 3ms/step - loss: 0.8479 - acc: 0.0000e+
Epoch 30/100
427/427 [==============================] - 1s 3ms/step - loss: 0.8323 - acc: 0.0000e+
Epoch 31/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7964 - acc: 0.0000e+
Epoch 32/100
427/427 [==============================] - 1s 3ms/step - loss: 0.8261 - acc: 0.0000e+
Epoch 33/100
427/427 [==============================] - 1s 3ms/step - loss: 0.8114 - acc: 0.0000e+
Epoch 34/100
427/427 [==============================] - 1s 3ms/step - loss: 0.8281 - acc: 0.0000e+
Epoch 35/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7679 - acc: 0.0000e+
Epoch 36/100
427/427 [==============================] - 1s 3ms/step - loss: 0.8105 - acc: 0.0000e+
Epoch 37/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7924 - acc: 0.0000e+
Epoch 38/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7719 - acc: 0.0000e+
Epoch 39/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7930 - acc: 0.0000e+
Epoch 40/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7406 - acc: 0.0000e+
Epoch 41/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7874 - acc: 0.0000e+
Epoch 42/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7816 - acc: 0.0000e+
Epoch 43/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7781 - acc: 0.0000e+
Epoch 44/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7825 - acc: 0.0000e+
Epoch 45/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7498 - acc: 0.0000e+
Epoch 46/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7355 - acc: 0.0000e+
Epoch 47/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7383 - acc: 0.0000e+
Epoch 48/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7791 - acc: 0.0000e+
Epoch 49/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7407 - acc: 0.0000e+
Epoch 50/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7473 - acc: 0.0000e+
Epoch 51/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7301 - acc: 0.0000e+
Epoch 52/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7511 - acc: 0.0000e+
Epoch 53/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6949 - acc: 0.0000e+
Epoch 54/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7170 - acc: 0.0000e+
Epoch 55/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7276 - acc: 0.0000e+
Epoch 56/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7010 - acc: 0.0000e+
Epoch 57/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7188 - acc: 0.0000e+
```

```
Epoch 58/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7016 - acc: 0.0000e+
Epoch 59/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7295 - acc: 0.0000e+
Epoch 60/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7264 - acc: 0.0000e+
Epoch 61/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6682 - acc: 0.0000e+
Epoch 62/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7222 - acc: 0.0000e+
Epoch 63/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6951 - acc: 0.0000e+
Epoch 64/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7004 - acc: 0.0000e+
Epoch 65/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7102 - acc: 0.0000e+
Epoch 66/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7045 - acc: 0.0000e+
Epoch 67/100
427/427 [==============================] - 1s 3ms/step - loss: 0.7047 - acc: 0.0000e+
Epoch 68/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6977 - acc: 0.0000e+
Epoch 69/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6832 - acc: 0.0000e+
Epoch 70/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6627 - acc: 0.0000e+
Epoch 71/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6873 - acc: 0.0000e+
Epoch 72/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6946 - acc: 0.0000e+
Epoch 73/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6873 - acc: 0.0000e+
Epoch 74/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6566 - acc: 0.0000e+
Epoch 75/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6901 - acc: 0.0000e+
Epoch 76/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6839 - acc: 0.0000e+
Epoch 77/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6527 - acc: 0.0000e+
Epoch 78/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6459 - acc: 0.0000e+
Epoch 79/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6751 - acc: 0.0000e+
Epoch 80/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6639 - acc: 0.0000e+
Epoch 81/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6555 - acc: 0.0000e+
Epoch 82/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6601 - acc: 0.0000e+
Epoch 83/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6115 - acc: 0.0000e+
Epoch 84/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6476 - acc: 0.0000e+
Epoch 85/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6256 - acc: 0.0000e+
Epoch 86/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6768 - acc: 0.0000e+
Epoch 87/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6399 - acc: 0.0000e+
Epoch 88/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6266 - acc: 0.0000e+
```

```
427/427 [==============================] - 1s 3ms/step - loss: 0.6266 - acc: 0.0000e+
Epoch 89/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6240 - acc: 0.0000e+
Epoch 90/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6269 - acc: 0.0000e+
Epoch 91/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6289 - acc: 0.0000e+
Epoch 92/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6331 - acc: 0.0000e+
Epoch 93/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6190 - acc: 0.0000e+
Epoch 94/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6248 - acc: 0.0000e+
Epoch 95/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6257 - acc: 0.0000e+
Epoch 96/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6182 - acc: 0.0000e+
Epoch 97/100
427/427 [==============================] - 1s 3ms/step - loss: 0.5983 - acc: 0.0000e+
Epoch 98/100
427/427 [==============================] - 1s 3ms/step - loss: 0.5794 - acc: 0.0000e+
Epoch 99/100
427/427 [==============================] - 1s 3ms/step - loss: 0.5872 - acc: 0.0000e+
Epoch 100/100
427/427 [==============================] - 1s 3ms/step - loss: 0.6300 - acc: 0.0000e+
```

Plot result

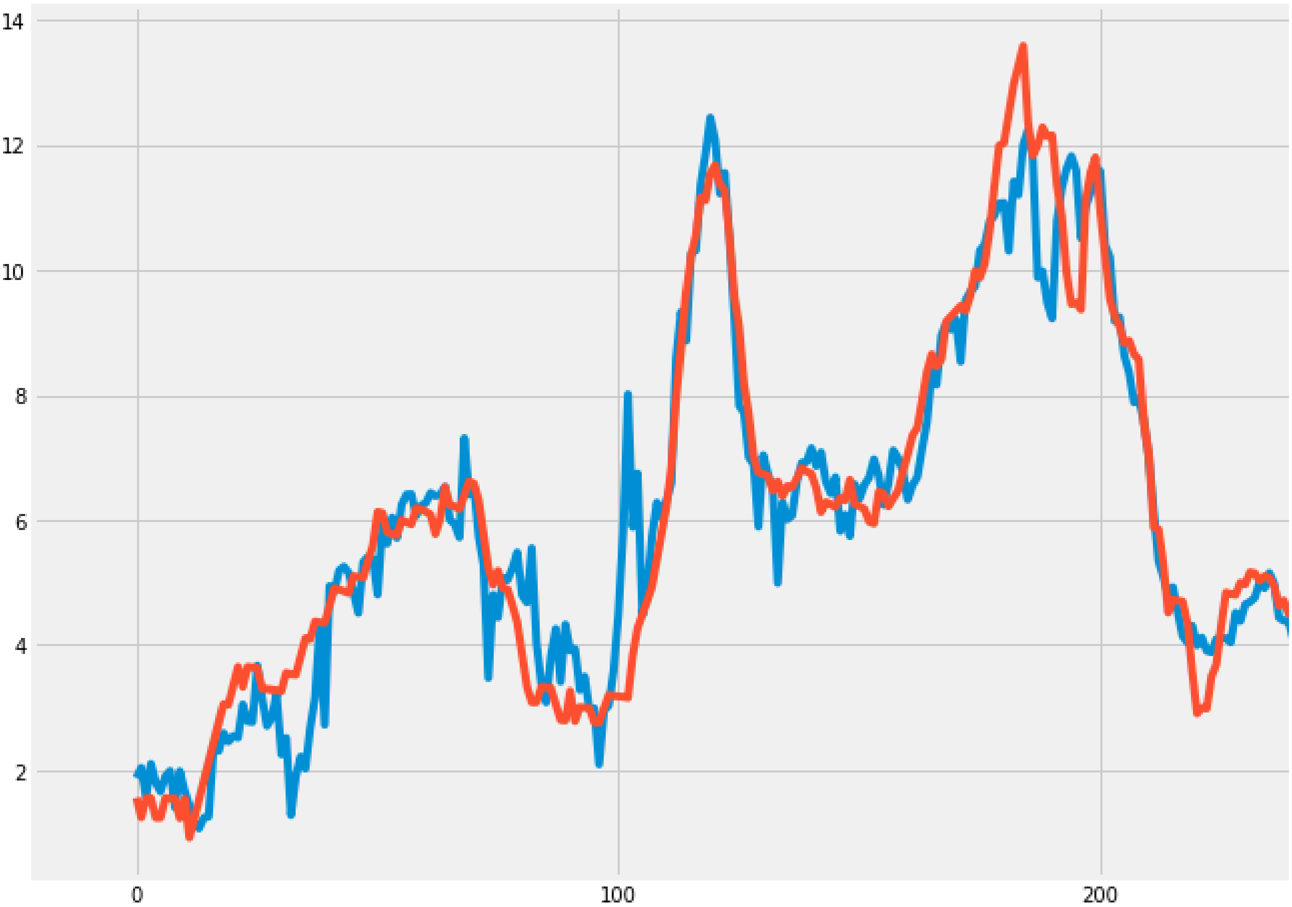
Plot result

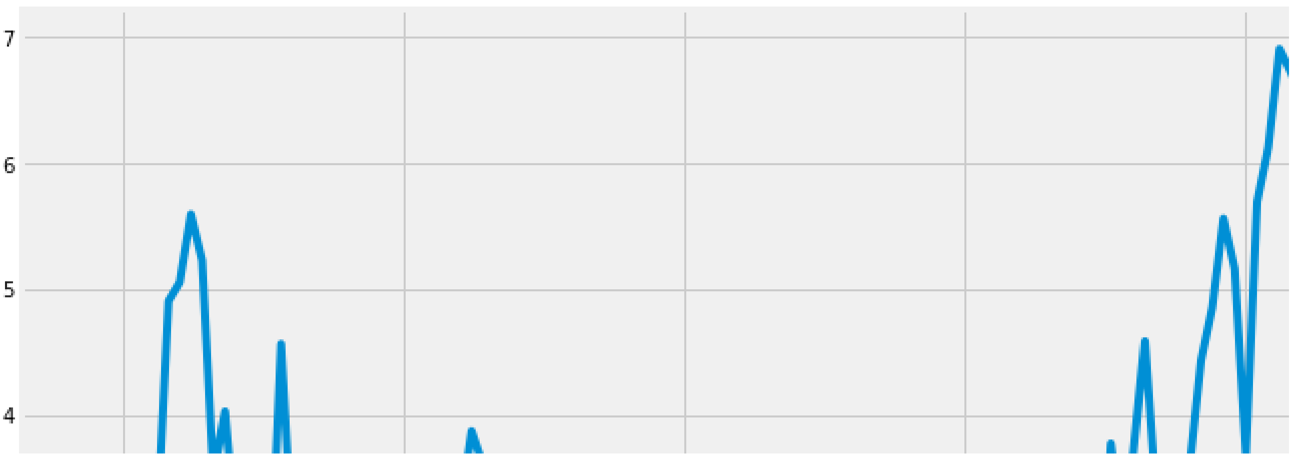


Plot predictions

Plot predictions

```
Train Score: 0.71 RMSE
Test Score: 1.53 RMSE
```





Data Analysis

We only trained the model for 100 epochs, feel free to modify it to any number as long as we have results we find during the experiments

- LSTM with Adam or RMSprop optimizers work better than the SGD optimizer in this project.
- Each model fits the training dataset very well.
- **The prediction captures the range and characteristics of the real dat**
- **The model doesn't predict the rapid increasing near the 100th test d**

Part II.2 Generalized model: multi-step, multi-fea

We build a multi-step, multi-feature LSTM model in this section. That means we can use several-d
features in the future.

For example, we can use last 12-month's data of Wage, Consumption, Ir
. In this section, we

- Process the data to fit the requirements of all possible multi-step, multi-feature prediction ta
- We modify the LSTM model accordingly.
- Plot the 3-month prediction for Inflation and Unemployment with last 12-month's data of Wa

Data preparation

Make the data forms are all correct

```
(418, 12, 4)
(418, 6)
(180, 12, 4)
(418, 6)
```

Scaling, vectorize and de_vectorize

Multi-step LSTM model, change the input_shape and Dense layer parameter to

Train the model. Change the optimizer parameter to use other optimizers, e.g.

```
418/418 [==============================] - 5s 11ms/step - loss: 0.0412 - acc: 0.5598
Epoch 165/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0393 - acc: 0.5359
Epoch 166/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0408 - acc: 0.5646
Epoch 167/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0359 - acc: 0.5789
Epoch 168/200
418/418 [==============================] - 4s 10ms/step - loss: 0.0431 - acc: 0.5742
Epoch 169/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0365 - acc: 0.5742
Epoch 170/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0400 - acc: 0.5646
Epoch 171/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0396 - acc: 0.5789
Epoch 172/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0393 - acc: 0.5718
Epoch 173/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0373 - acc: 0.5789
Epoch 174/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0394 - acc: 0.5909
Epoch 175/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0359 - acc: 0.5478
Epoch 176/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0361 - acc: 0.5670
Epoch 177/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0336 - acc: 0.5718
Epoch 178/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0420 - acc: 0.5766
Epoch 179/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0380 - acc: 0.5622
Epoch 180/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0358 - acc: 0.5502
Epoch 181/200
418/418 [==============================] - 4s 10ms/step - loss: 0.0352 - acc: 0.5407
Epoch 182/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0404 - acc: 0.5885
Epoch 183/200
418/418 [==============================] - 4s 10ms/step - loss: 0.0381 - acc: 0.6077
Epoch 184/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0359 - acc: 0.5718
Epoch 185/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0364 - acc: 0.5526
Epoch 186/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0351 - acc: 0.5742
Epoch 187/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0344 - acc: 0.5646
Epoch 188/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0348 - acc: 0.5598
Epoch 189/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0348 - acc: 0.5789
Epoch 190/200
418/418 [==============================] - 4s 10ms/step - loss: 0.0370 - acc: 0.5742
Epoch 191/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0361 - acc: 0.5622
Epoch 192/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0327 - acc: 0.5550
Epoch 193/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0346 - acc: 0.5837
Epoch 194/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0342 - acc: 0.5718
Epoch 195/200
```

```
Epoch 195/200
418/418 [==============================] - 4s 11ms/step - loss: 0.0324 - acc: 0.5670
Epoch 196/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0328 - acc: 0.5646
Epoch 197/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0396 - acc: 0.5789
Epoch 198/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0418 - acc: 0.5957
Epoch 199/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0340 - acc: 0.5502
Epoch 200/200
418/418 [==============================] - 5s 11ms/step - loss: 0.0350 - acc: 0.5239
```