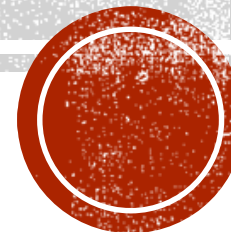


Time Series Analysis Results

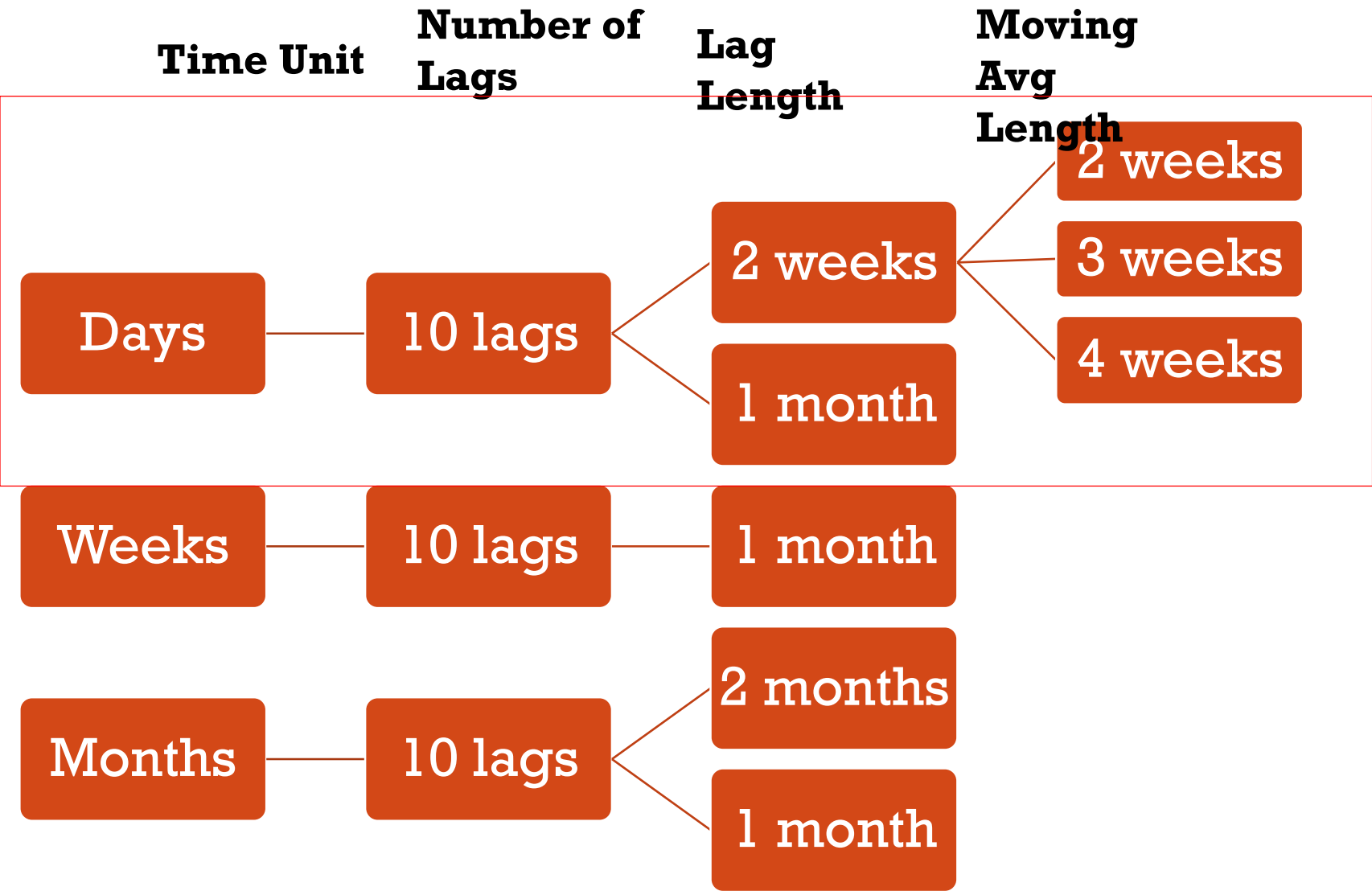
Oct 24, 2019

Chiemi Kato



PARAMETERS

NOTE: Focused on Days, since at a glance it performed much stronger



MAE RESULTS (ALL 1 YEAR GAP FROM Y TARGET)

Model	Time Unit	# Lags	Lag Length	Rollavg Window	Parameters	MAE
KNN	Busi. Day	10	1 month	8 weeks	K=2	\$1199.13270
Polynomial	Busi. Day	10	1 month	8 weeks	Deg=3	\$1173.80122
Adaboost	Busi. Day	10	1 month	8 weeks	{'learning_rate': 0.0001, 'n_estimators': 150}	\$1841.94740
KNN	Busi. Day	10	1 month	2 weeks	K=7	\$1253.4538
Polynomial	Busi. Day	10	1 month	2 weeks	Deg=3	\$1264.15163
Adaboost	Busi. Day	10	1 month	2 weeks	{'learning_rate': 0.0001, 'n_estimators': 50}	\$1364.1369
KNN	Busi. Day	10	2 week	3 weeks	K=6	\$1045.21761
Polynomial	Busi. Day	10	2 week	3 weeks	Deg=3	\$1301.42426
Adaboost	Busi. Day	10	2 week	3 weeks	{'learning_rate': 0.0001, 'n_estimators': 50}	\$1336.64916
KNN	Busi. Day	10	1 month	4 weeks	K=7	\$974.02
Polynomial	Busi. Day	10	1 month	4 weeks	Deg=3	\$1417.294
Adaboost	Busi. Day	10	1 month	4 weeks	{'learning_rate': 0.0001, 'n_estimators': 50}	\$1401.205



PREPROCESSING METHODS

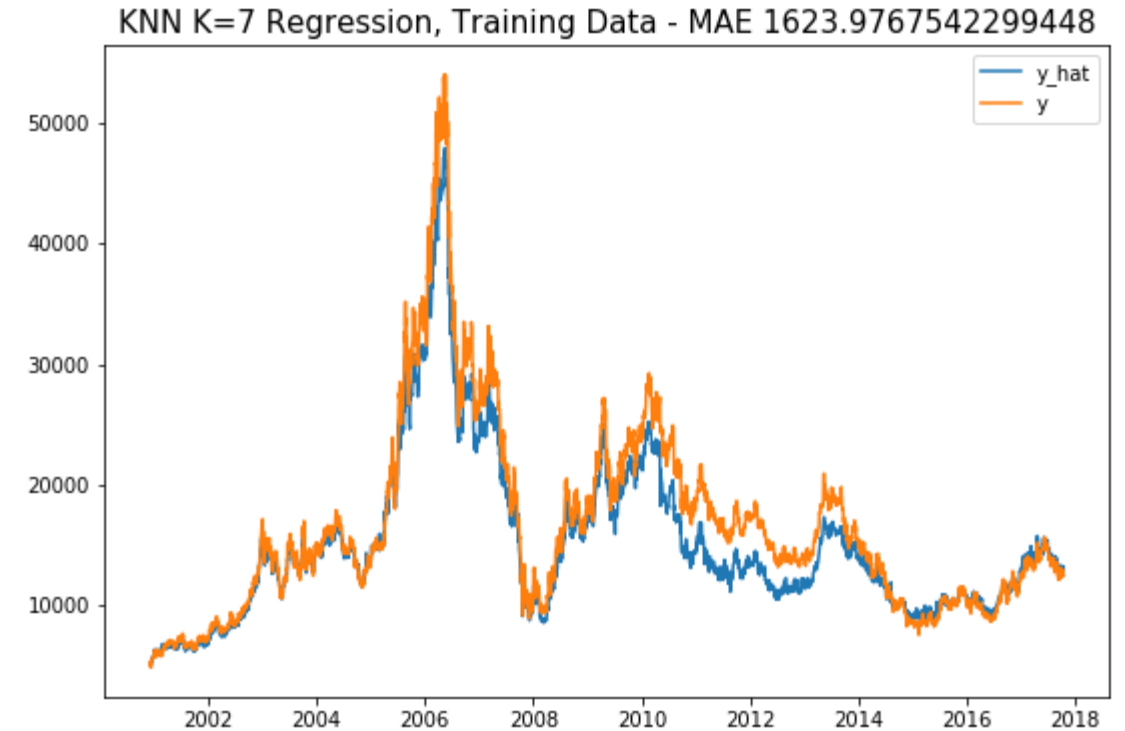
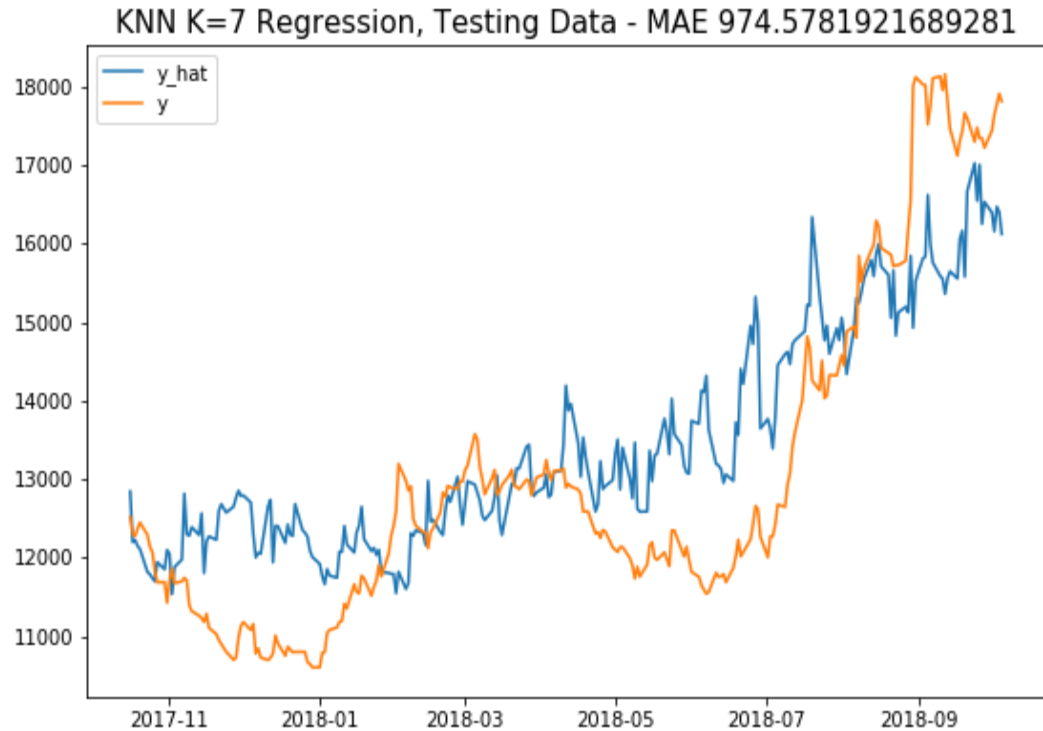
NOTE: window_setting refers to the rolling mean window column in results table

```
1 def preprocess(series, window_setting):
2     moving_avg = series.rolling(window=window_setting).mean().shift()
3     moving_avg_diff = series-moving_avg
4     return moving_avg_diff
5
6
```

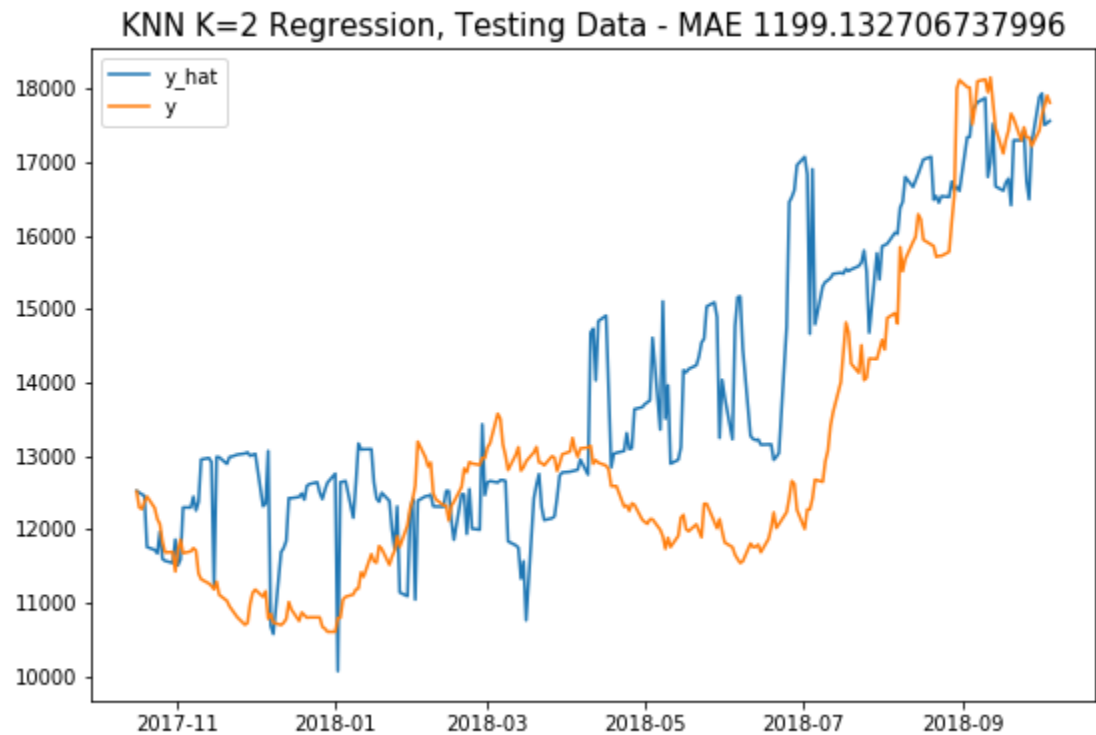
```
1 LME_stationary = preprocess(LME, window_setting)
2
3 list_one = list(LME_stationary.index)
4 list_two = list(LME_stationary)
5 df = pd.DataFrame(list(zip(list_one, list_two)), columns = ['ds', 'lag0'])
6 df.head()
7
8 # How many lag periods? - LAG DAYS
9 lag_length = 10
10
11 df['lag1'] = df.lag0.shift(periods=1*lag_length)
12 df['lag2'] = df.lag0.shift(periods=2*lag_length)
13 df['lag3'] = df.lag0.shift(periods=3*lag_length)
14 df['lag4'] = df.lag0.shift(periods=4*lag_length)
15 df['lag5'] = df.lag0.shift(periods=5*lag_length)
16
17 df['lag6'] = df.lag0.shift(periods=6*lag_length)
18 df['lag7'] = df.lag0.shift(periods=7*lag_length)
19 df['lag8'] = df.lag0.shift(periods=8*lag_length)
20 df['lag9'] = df.lag0.shift(periods=9*lag_length)
21 df['lag10'] = df.lag0.shift(periods=10*lag_length)
22
23 df.index = df['ds']
24 df = df.iloc[:, 1:]
25
26
27 df['y'] = df['lag0'].shift(-261)
28 df = df.dropna()
29 df.head()
```

Selected Result Graphs...

**Below is graph of 10 lags, lag length = 4 weeks,
rolling window length = 4 weeks**

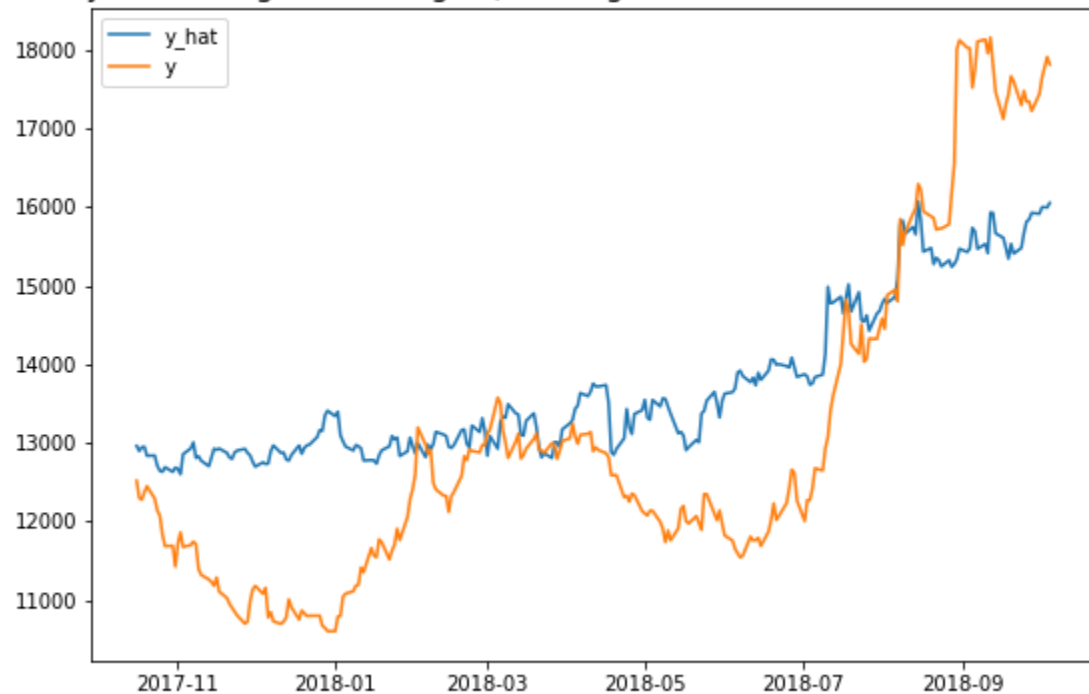


Graph of 10 lags, lag length = 4 weeks, rolling window length = 8 weeks



Graph of 10 lags, lag length = 4 weeks, rolling window length = 8 weeks

Polynomial Regression deg=3, Testing Data - MAE 1173.8012205546922

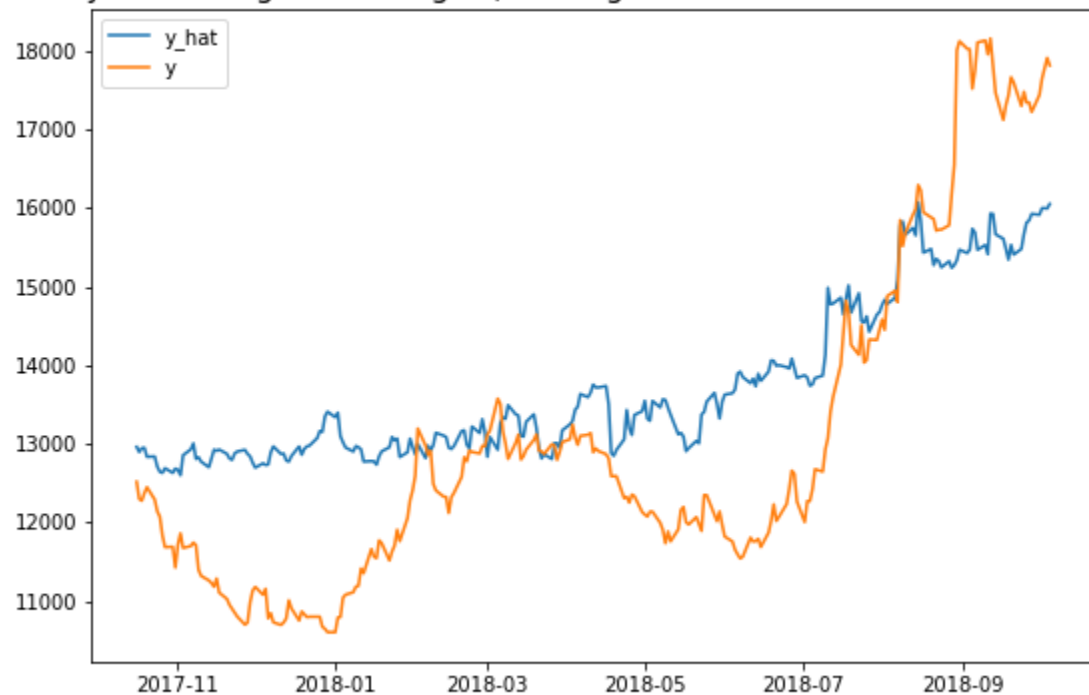


Polynomial Regression deg=3, Training Data - MAE 648.4059769831357

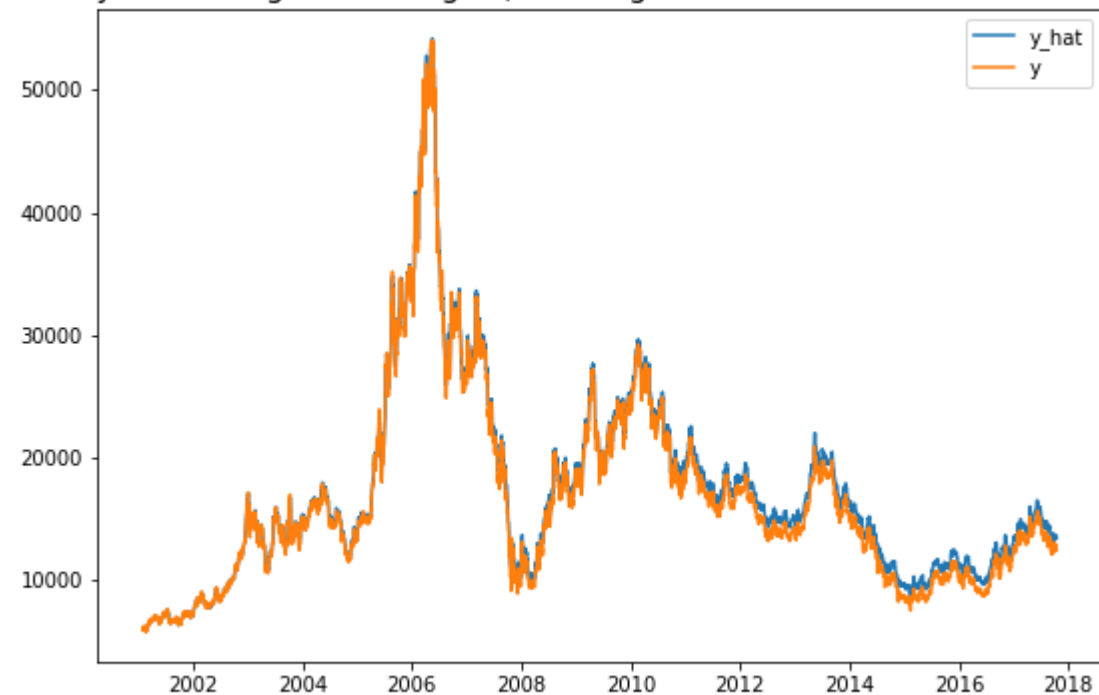


Graph of 10 lags, lag length = 4 weeks, rolling window length = 8 weeks

Polynomial Regression deg=3, Testing Data - MAE 1173.8012205546922

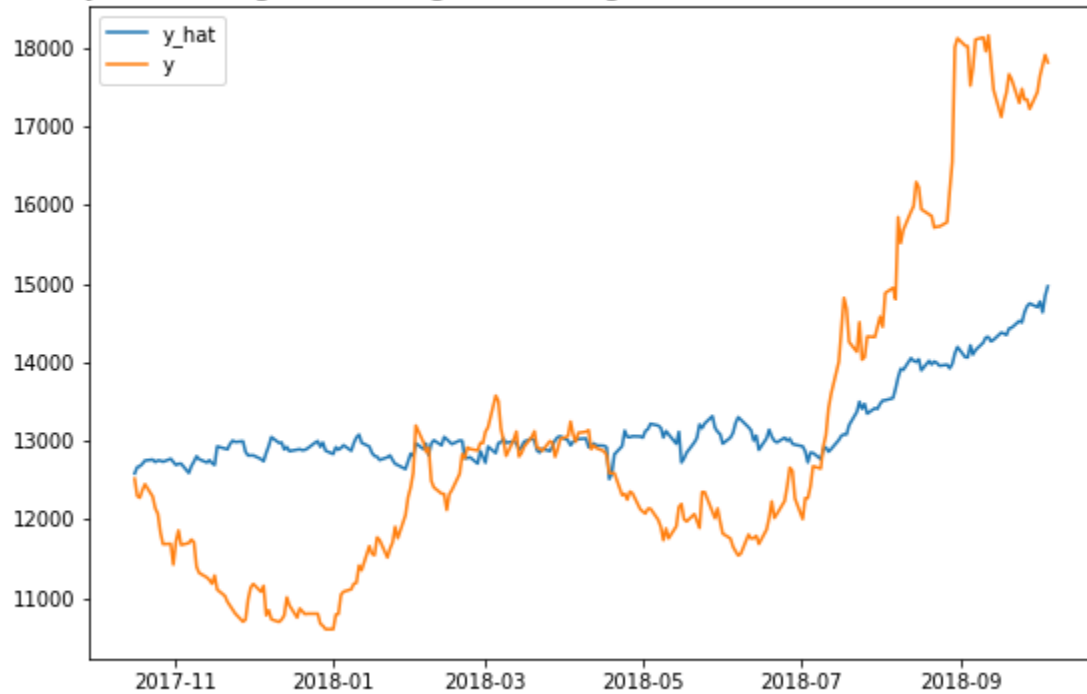


Polynomial Regression deg=3, Training Data - MAE 648.4059769831357



Graph of 10 lags, lag length = 4 weeks, rolling window length = 2 weeks

Polynomial Regression deg=3, Testing Data - MAE 1264.1516329231376

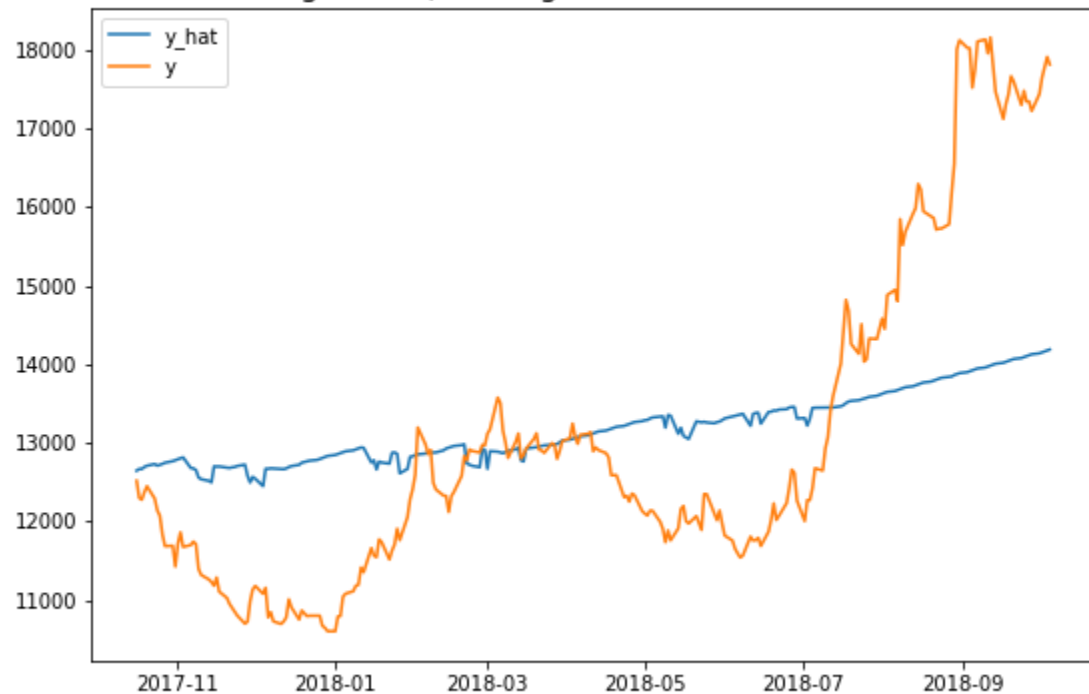


Polynomial Regression deg=3, Training Data - MAE 7074.6374805570695



Graph of 10 lags, lag length = 2 weeks, rolling window length = 2 weeks

Adaboost Regression, Testing Data - MAE 1334.5964229057886

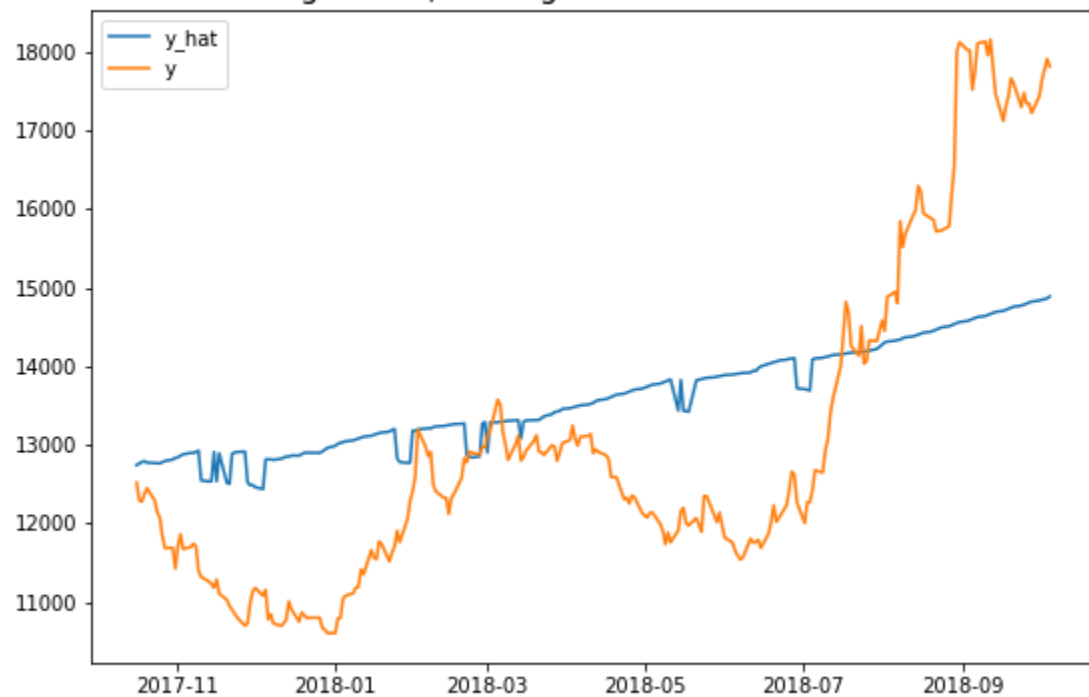


Adaboost Regression, Training Data - MAE 4998.09009775068



Graph of 10 lags, lag length = 4 weeks, rolling window length = 4 weeks

Adaboost Regression, Testing Data - MAE 1401.2053402117742



Adaboost Regression, Training Data - MAE 1623.9767542299448

