Ch6 – Final Assignment

Tesla stock price prediction using Times Series Prediction, Deep learning

Wang Xu finish at 2022-4-28

Friendly remind: This report only shows a part of code.

Please view whole code in my github link : ipynb link

Contents

- The main objectives of this assignment is focusing on **time series prediction** of Tesla stock price and choose a preferred model.

- Experiments specifically on parameter variations of models:
  - **ARIMA** (Autoregression Integrated Moving Average);
  - **SARIMA** (Seasonal Autoregression Integrated Moving Average) ;
  - **and deep learning LSTM** (Long-Short Term Memory Network).

- It may give benefits as advices or price trend predictions for individual investors, investment funds, any other stock research organizations, etc.

Brief Description of dataset

- Tesla's stock price data from its initial public offering (IPO) date 2010-6-29 to 2017-3-17. (1692 rows)
- Data download from: https://www.kaggle.com/datasets/rpaguirre/tesla-stock-price

7 Attributes:

1. "Date" - The date
2. "Open" - The opening price of the stock
3. "High" - The high price of that day
4. "Low" - The low price of that day
5. "Close" -The closed price of that day
6. "Volume" - The amount of stocks traded during that day
7. "Adjusted Close" - The stock's closing price that has been amended to include any distributions/corporate actions that occurs before next days open

- **In this case, "Close" price will be considered as main prediction.**

4.1 Import libraries

```
# imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from datetime import datetime
import os
os.chdir('data')
from colorsetup import colors, palette
sns.set_palette(palette)
# ignore warnings
warnings.filterwarnings('ignore')
pd.options.display.float_format = '{:,.1f}'.format
%matplotlib inline
plotsize = (12, 8)
from statsmodels.tsa.stattools import adfuller
```

```
df = pd.read_csv("Tesla_stock.csv")
```

```
df.head()
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.0	25.0	17.5	23.9	18766300	23.9
1	6/30/2010	25.8	30.4	23.3	23.8	17187100	23.8
2	7/1/2010	25.0	25.9	20.3	22.0	8218800	22.0
3	7/2/2010	23.0	23.1	18.7	19.2	5139800	19.2
4	7/6/2010	20.0	20.0	15.8	16.1	6866900	16.1

4.2 Data cleaning

```
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
```

```
df.head()
```

Date	Open	High	Low	Close	Volume	Adj Close
2010-06-29	19.0	25.0	17.5	23.9	18766300	23.9
2010-06-30	25.8	30.4	23.3	23.8	17187100	23.8
2010-07-01	25.0	25.9	20.3	22.0	8218800	22.0
2010-07-02	23.0	23.1	18.7	19.2	5139800	19.2
2010-07-06	20.0	20.0	15.8	16.1	6866900	16.1

```
df.describe()
```

	Open	High	Low	Close	Volume	Adj Close
count	1,692.0	1,692.0	1,692.0	1,692.0	1,692.0	1,692.0
mean	132.4	134.8	130.0	132.4	4,270,740.9	132.4
std	94.3	95.7	92.9	94.3	4,295,971.3	94.3
min	16.1	16.6	15.0	15.8	118,500.0	15.8
25%	30.0	30.6	29.2	29.9	1,194,350.0	29.9
50%	156.3	162.4	153.2	158.2	3,180,700.0	158.2
75%	220.6	224.1	217.1	220.0	5,662,100.0	220.0
max	287.7	291.4	280.4	286.0	37,163,900.0	286.0

4.3 Visualization

```
1 plt.figure(figsize=(18,8))
2 plt.plot(df['Close'],'b-')
3 plt.title("Tesla - Close price")
4 plt.ylabel("Stock price")
5 plt.grid()
```

Check its stationarity

- The prediction should be more reasonable on the close price.
- from close price plot, it seems that is non-stationarity.

4.4.1 split chunks to varify mean, variance

```
1 close_array = pd.Series(df['Close'].values)
2 chunks = np.split(close_array,indices_or_sections=9)
3 print("{}|{:7}|{}".format("chunks","mean","variance"))
4 print("-"*25)
5 for i, j in enumerate(chunks,1):
6     print("{:6}|{:.6}|{:.6}".format(i, np.mean(j), np.var(j)))
```

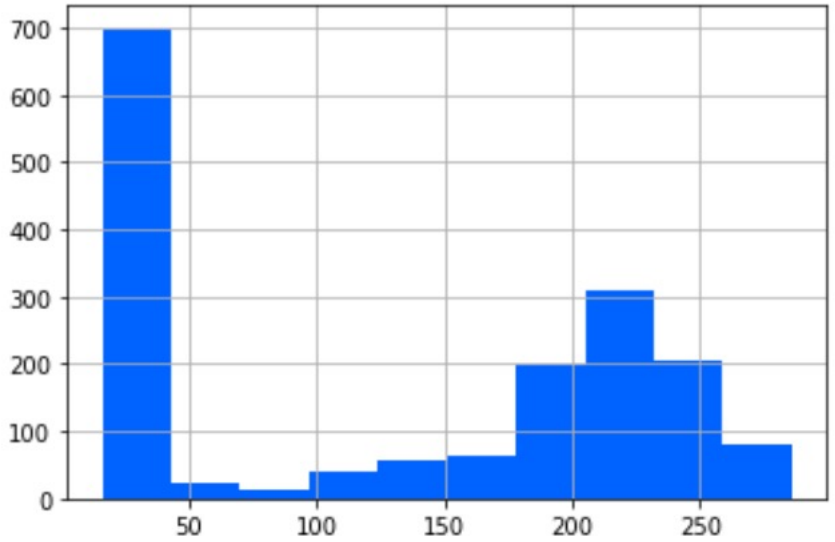
```
chunks|mean   |variance
-------------------------
     1|23.6027|18.6438
     2|27.5632|7.33095
     3|31.0821|7.76437
     4|46.2839|539.716
     5|164.471|1263.25
     6|230.328|557.292
     7|230.528|680.763
     8|218.87|580.545
     9|219.128|596.781
```

It has changing means and changing variances.

4.4.2 Whether is normal distribution

```
1 close_array.hist()
```

```
<AxesSubplot:>
```



It is not normal distribution.

4.4.3 Statstics - AdFuller test

Addiction to visualization, statstics proving whether it is stationarity helps as well.

```
1 adf, pvalue, usedlag, nobs, critical_values, icbest = adfuller(df['Close'])
2 print("Adf:",adf)
3 print("p-value:",pvalue)
```

```
Adf: -0.8137329693176628
p-value: 0.8150547908640395
```

p-value > 0.05 clearly, it is non-stationarity.

Non-stationarity!

Model preparing

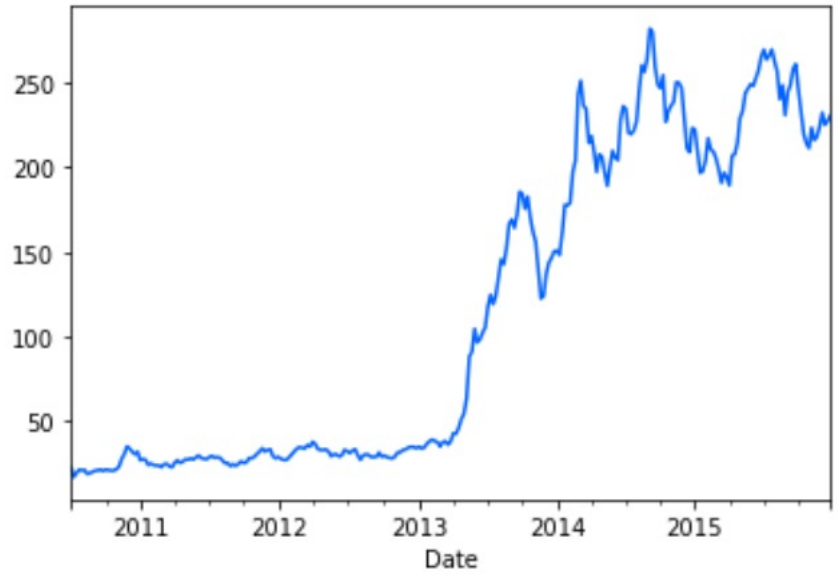
5.1 Split train(2010:2015) and test dataset

```
1  close_week = df['Close'].resample('W').mean()
```

```
1  stock_train = close_week['2010':'2015']
```

```
1  stock_train.plot()
```

```
<AxesSubplot:xlabel='Date'>
```



Choose train-set :
2010 – 2015 close price in weeks
&
Transfer to stationarity

5.2 Differencing - transfor to stationarity

```
: 1 stock_diff = stock_train.diff()
  2 stock_diff = stock_diff.dropna()
  3 plt.figure(figsize=(12,6))
  4 plt.plot(stock_diff,'r-')
  5 plt.title("Close price after differencing")
```

```
: Text(0.5, 1.0, 'Close price after differencing')
```


```
1 adfuller(stock_diff)
```

```
(-13.684077570268537,
 1.3829853031786577e-25,
 0,
 285,
 {'1%': -3.4535050041524245,
  '5%': -2.8717352599720294,
  '10%': -2.5722024776854417},
 1841.0821374221925)
```

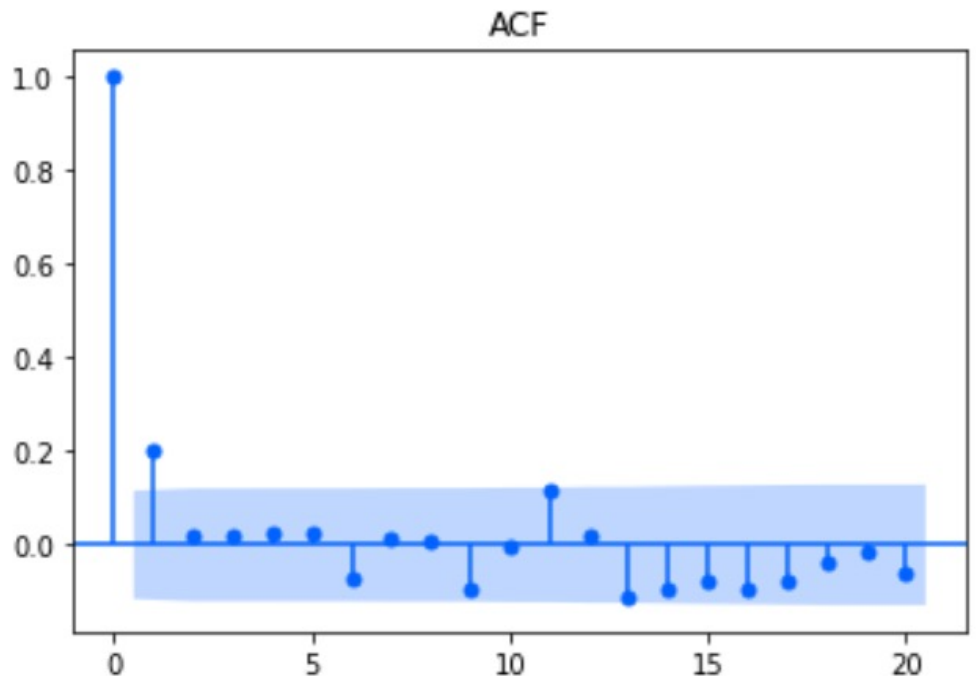
p-value < 0.05, now after differencing, the stock_diff is stationarity

Become more stationarity!

Parameters choosing from ACF (autocorrelation plot), PACF (partial autocorrelation plot) in (p,q)

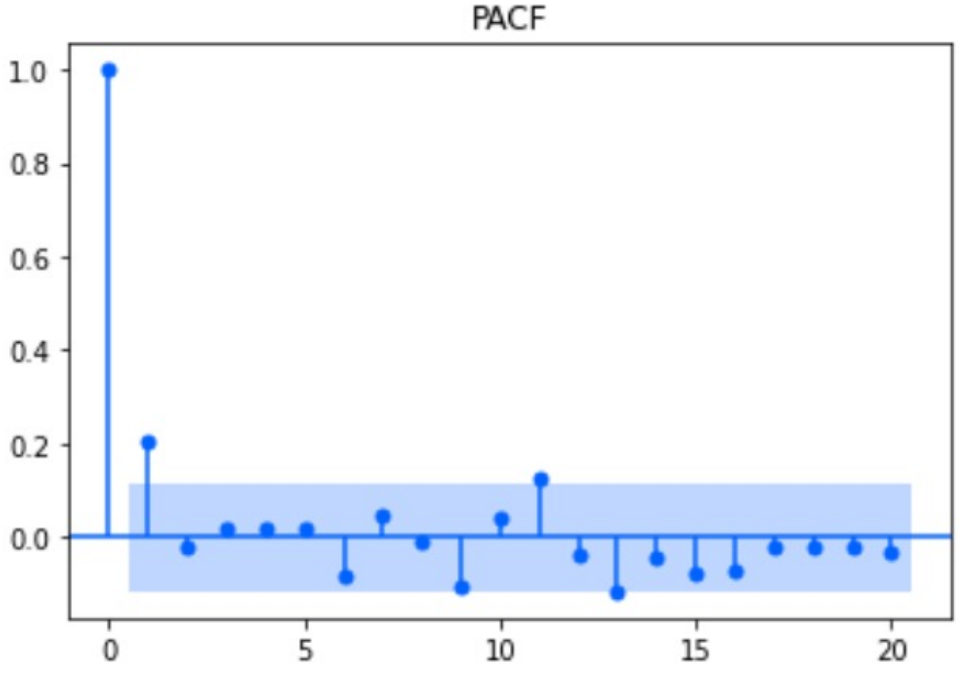
```
1 plot_acf(stock_diff,lags=20)
2 plt.title("ACF")
```

```
Text(0.5, 1.0, 'ACF')
```




```
1 plot_pacf(stock_diff,lags=20)
2 plt.title("PACF")
```

```
Text(0.5, 1.0, 'PACF')
```



From above plots, it seems that q=1, p=1 is preferred.

Baseline: Model_1, ARIMA, order=(1,1,1)

6.1 Parameters choosing from acf, pacf, order=(1,1,1)

```
1 from statsmodels.tsa.arima.model import ARIMA
```

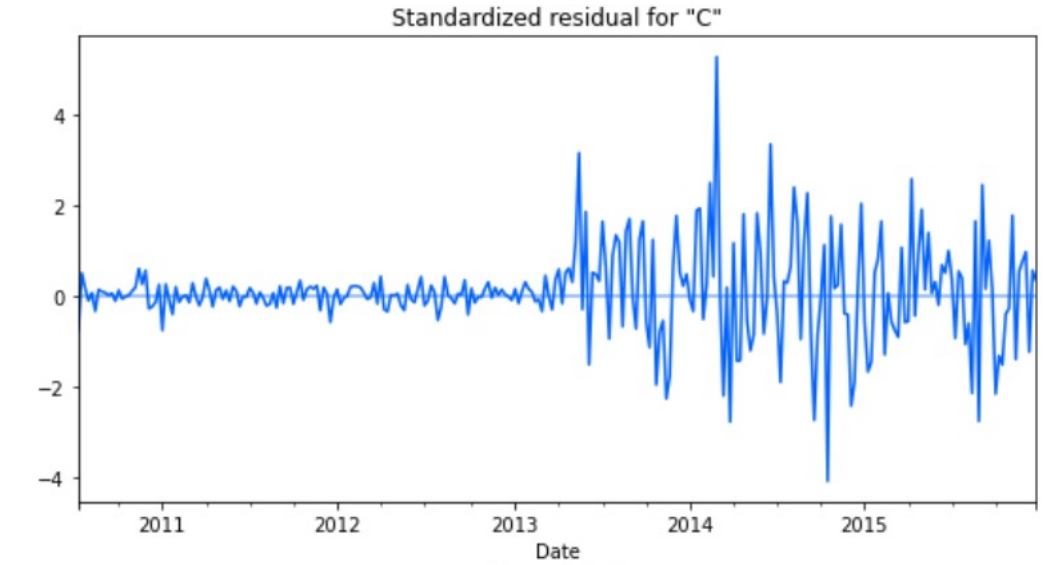
```
1 model_1 = ARIMA(stock_train,order=(1,1,1),freq='W').fit()
```

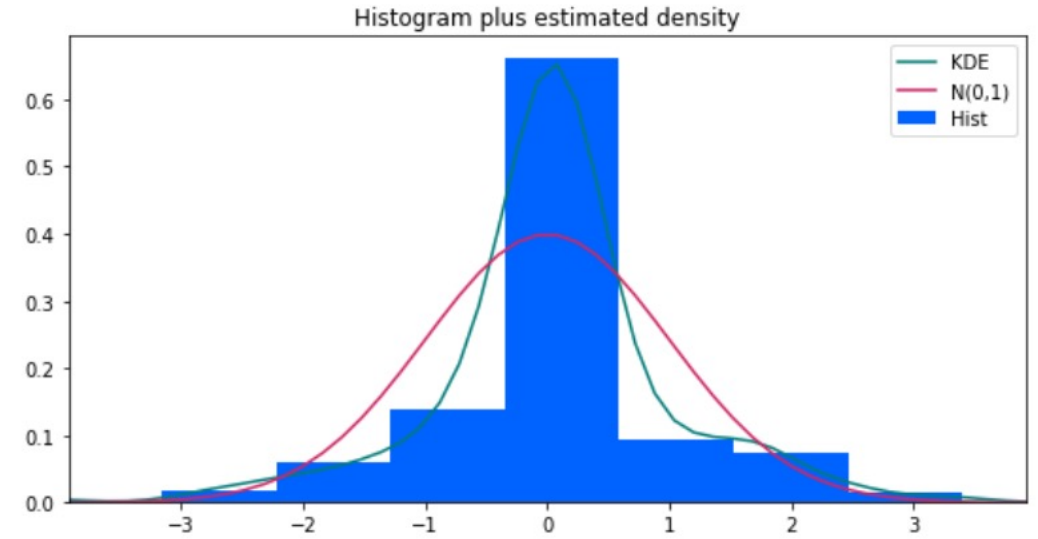
```
1 print(model_1.summary())
```

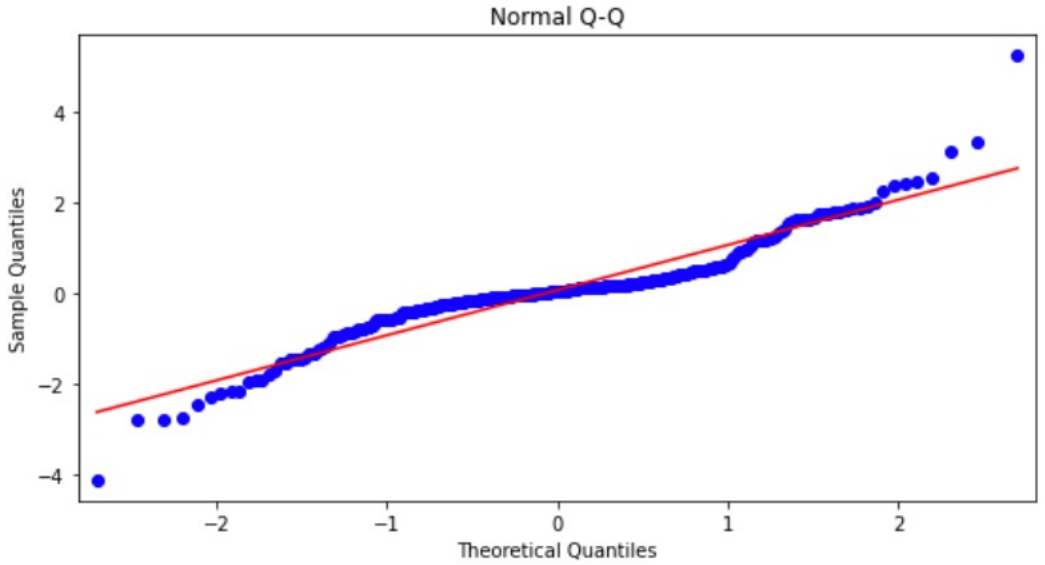
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  287
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -969.345
Date:                Thu, 28 Apr 2022   AIC                           1944.690
Time:                        10:37:50   BIC                           1955.658
Sample:                    07-04-2010   HQIC                          1949.086
                         - 12-27-2015
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1321      0.190      0.697      0.486      -0.240       0.504
ma.L1          0.0840      0.187      0.451      0.652      -0.282       0.450
sigma2        51.4491      2.571     20.014      0.000      46.411      56.487
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               213.03
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):              33.60   Skew:                             0.30
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.19
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

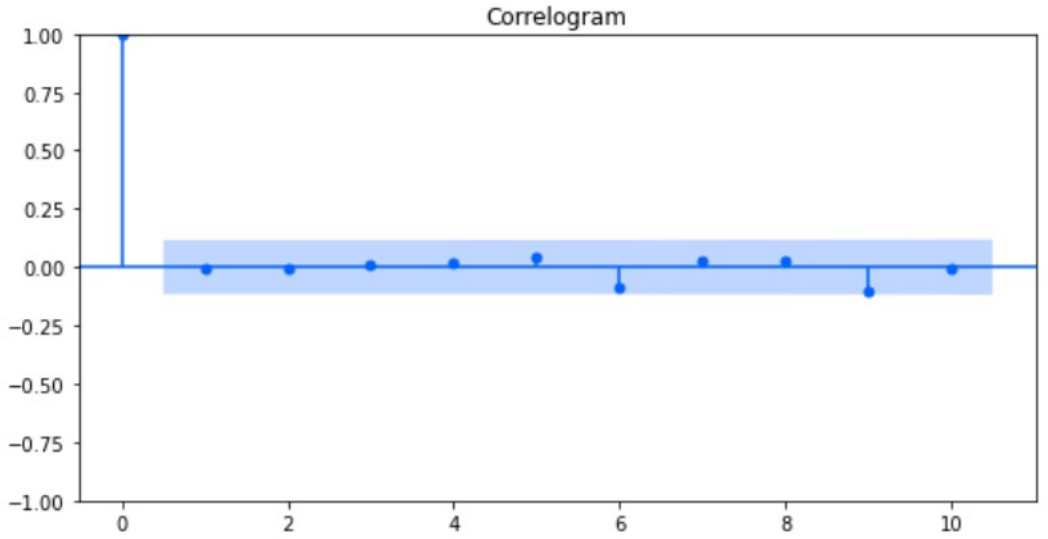
Information criteria (AIC, BIC) which penalize the number of parameters the model uses

```
1 model_1.plot_diagnostics(figsize=(20,10));
```



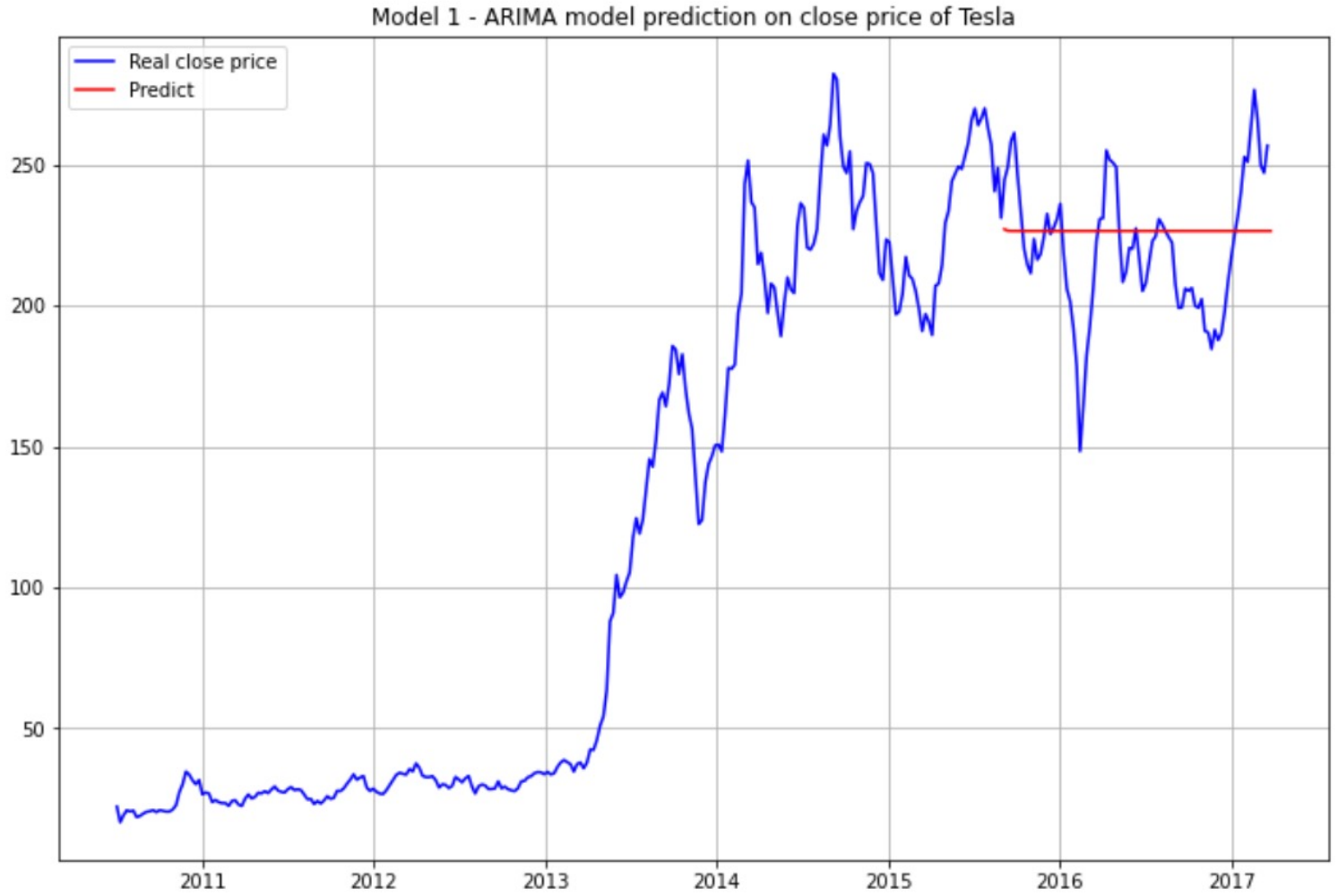






6.2 Prediction using model_1

```
predict_1 = model_1.predict(start=270,end=351,dynamic=True)
plt.figure(figsize=(12,8))
plt.plot(close_week,"b-",label="Real close price")
plt.plot(predict_1,'r-',label='Predict')
plt.legend(loc='best')
plt.title("Model 1 - ARIMA model prediction on close price of Tesla")
plt.grid(0.3)
```



6.3 Validation : Mse

```
num_weeks_test = len(predict_1)
close_test = close_week[-num_weeks_test:] #find th
```

```
def mse(observation, estimates):
    assert type(observation)==type(np.array([]))
    assert type(estimates)==type(np.array([]))
    assert len(observation)==len(estimates)
    diff = abs(observation) - abs(estimates)
    square = diff **2
    mse = sum(square)
    return mse
```

```
mse(close_test.values, predict_1.values)
```

```
50246.62748995037
```

Model_2, ARIMA, order=(1,0,1)

```
model_2 = pmd.auto_arima(stock_train,start_p=0,
                         start_q=0,max_p=3,max_q=3,
                         m=11, d=0, D=1,trace=True,
                         seasonal=False,
                         error_action='ignore',
                         suppress_warnings=True,stepwise=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=3680.755, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=inf, Time=0.10 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=inf, Time=0.13 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=1957.680, Time=0.11 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=1959.475, Time=0.15 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=1959.510, Time=0.10 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=2994.108, Time=0.13 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=inf, Time=0.11 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=inf, Time=0.33 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=1958.405, Time=0.18 sec

Best model:  ARIMA(1,0,1)(0,0,0)[0]
Total fit time: 1.365 seconds
```

SARIMAX Results

Dep. Variable:	Close	No. Observations:	287
Model:	ARIMA(1, 0, 1)	Log Likelihood	-975.209
Date:	Thu, 28 Apr 2022	AIC	1958.417
Time:	14:22:46	BIC	1973.055
Sample:	07-04-2010	HQIC	1964.284
	- 12-27-2015		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	113.4013	105.105	1.079	0.281	-92.602	319.404
ar.L1	0.9961	0.006	161.999	0.000	0.984	1.008
ma.L1	0.2132	0.047	4.553	0.000	0.121	0.305
sigma2	51.3949	2.440	21.065	0.000	46.613	56.177

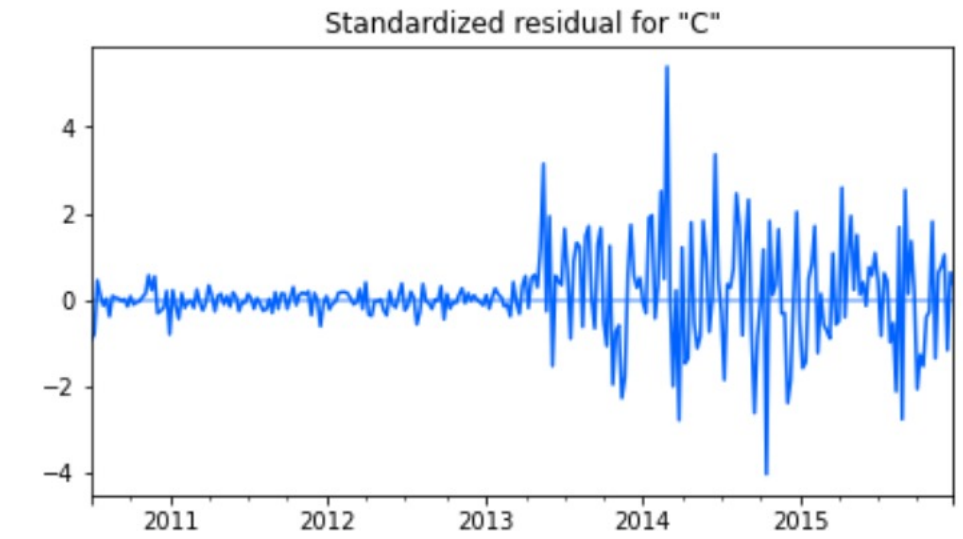
Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	228.76
Prob(Q):	0.97	Prob(JB):	0.00
Heteroskedasticity (H):	33.44	Skew:	0.43
Prob(H) (two-sided):	0.00	Kurtosis:	7.29

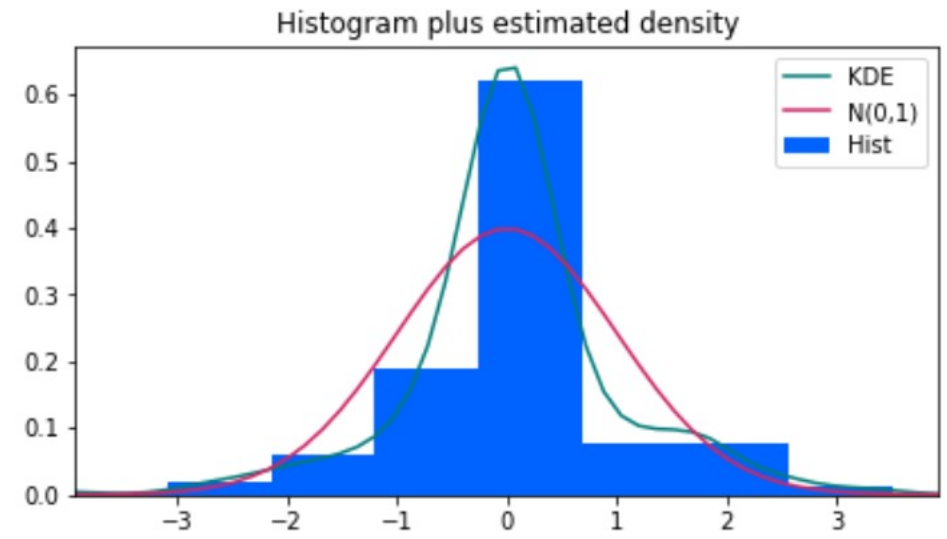
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

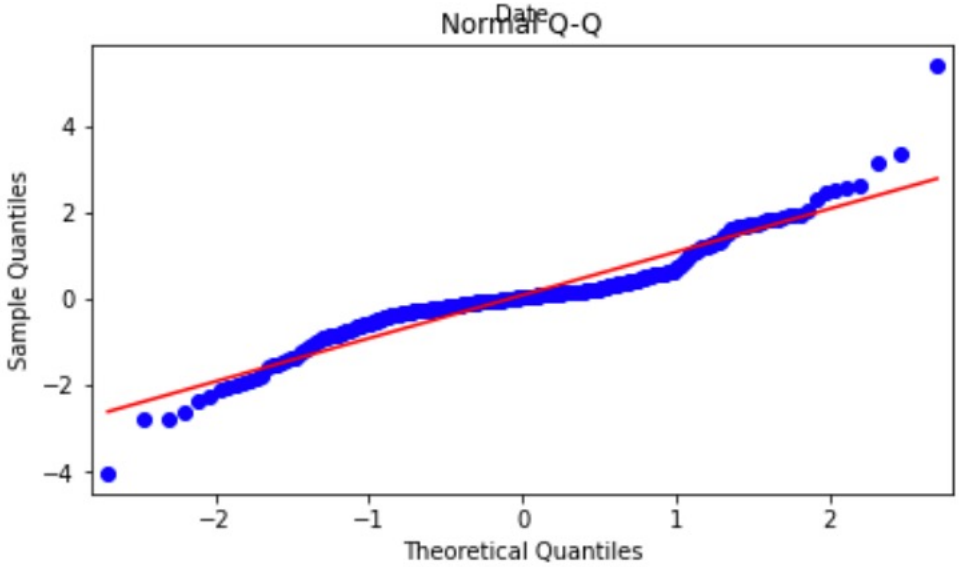
- **Using pmd module to find best parameters**
- **Information criteria (AIC, BIC) increased.**

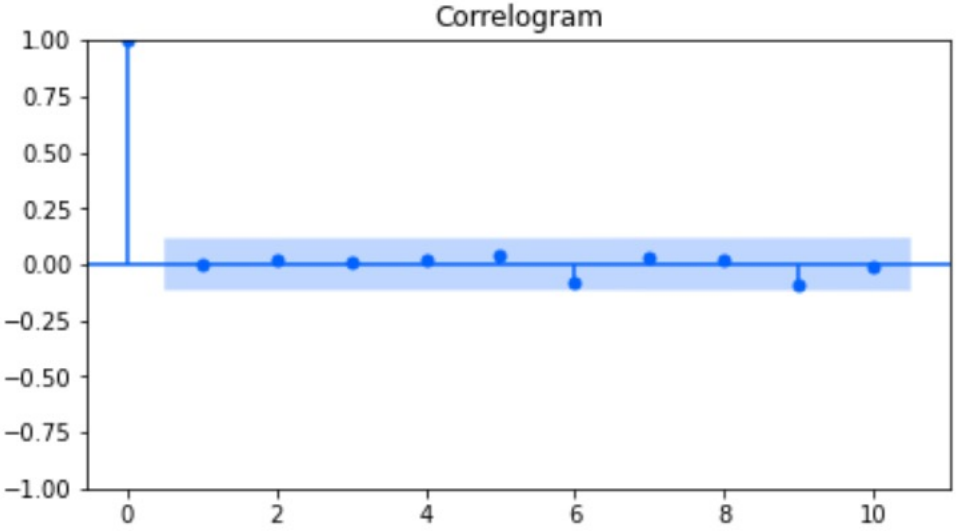
(penalize the number of parameters the model uses)

```
1 model_2_best.plot_diagnostics(figsize=(15,8));
```



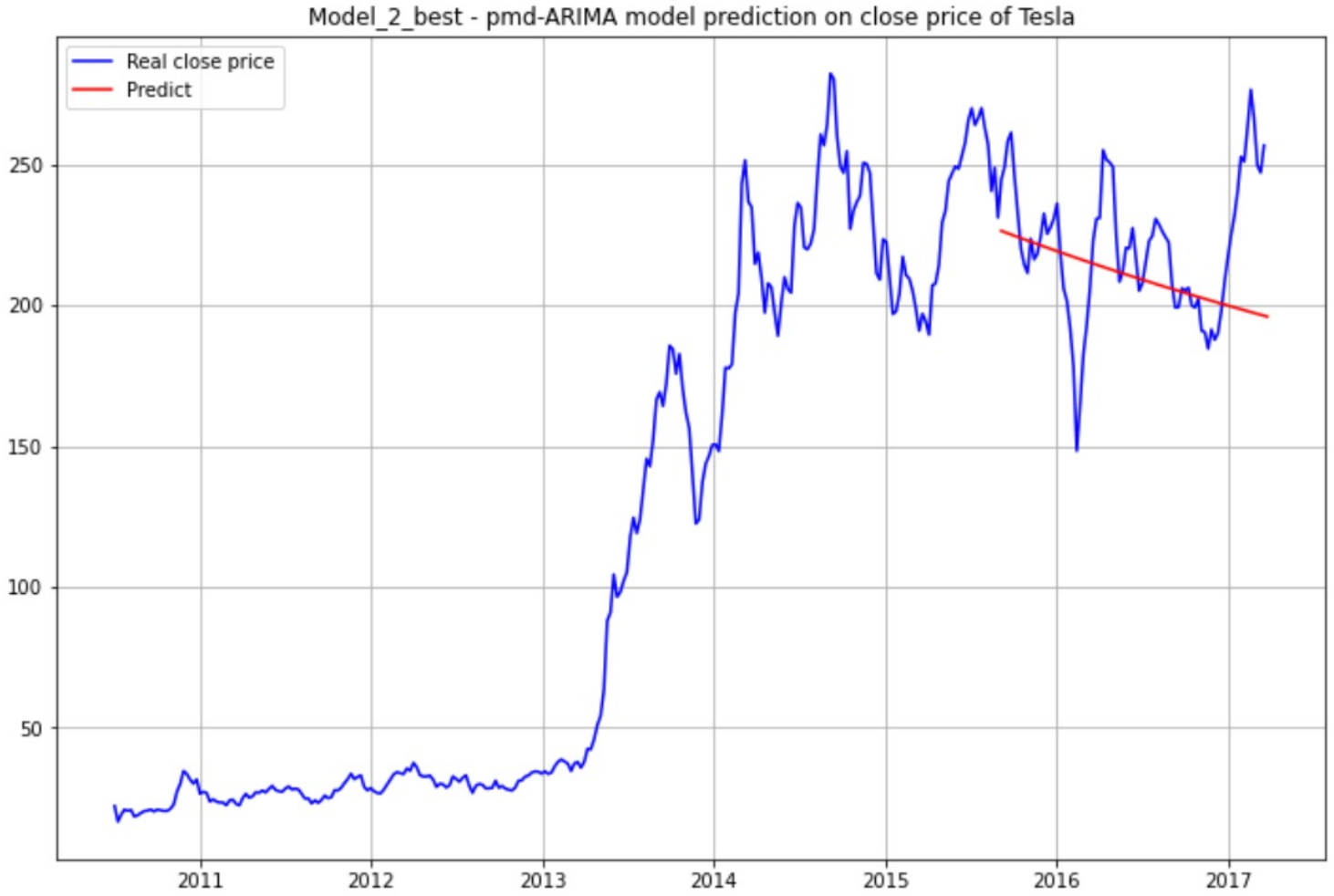






7.2 Prediction using model_2_best

```
predict_2 = model_2_best.predict(start=270,end=351,dynamic=True)
plt.figure(figsize=(12,8))
plt.plot(close_week,"b-",label="Real close price")
plt.plot(predict_2,'r-',label='Predict')
plt.legend(loc='best')
plt.title("Model_2_best - pmd-ARIMA model prediction on close price of Tesla")
plt.grid(0.3)
```



Mean square error

7.3 Validation: mse

```
print("MSE_model_1 (order = (1,1,1)):", mse(close_test.values,predi
print("MSE_model_2_best (order = (1,0,1)):",mse(close_test.values,
```

```
MSE_model_1 (order = (1,1,1)): 50246.62748995037
MSE_model_2_best (order = (1,0,1)): 61624.472763307844
```

from the comparison of model_1 and model_2, and their mse:

- Model_1's trend is more accurate than model_2.
- In this case, model_1 is preferred.

Model_3, SARIMA, order=(2,0,1), Seasonal order=(2,1,0,11)

- Using pmd module to find best parameters
- Similar information criteria (AIC, BIC)

```
print("best order:", model_3.order)
print("best seasonal order:",model_3.seasonal_order)
```

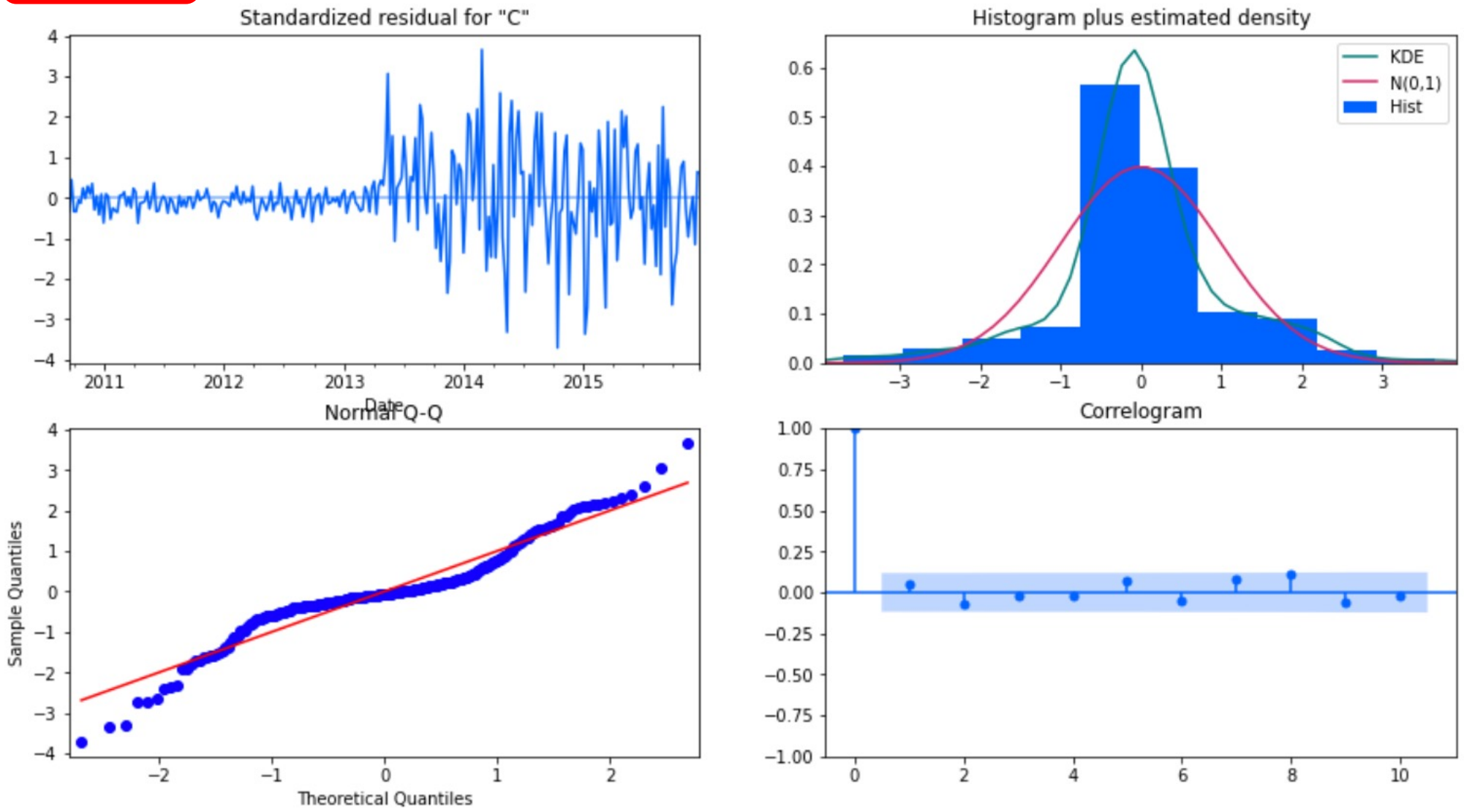
```
best order: (2, 0, 1)
best seasonal order: (2, 1, 0, 11)
```

```
model_3_best = sm.tsa.statespace.SARIMAX(stock_train,
                                         order=(2,0,1),
                                         seasonal_order=(2,1,0,11),
                                         trend='c').fit()
```

```
#comparing model_2 order=(1,0,1) vs. model_3 adding seasonal effect
print(model_3_best.summary())
```

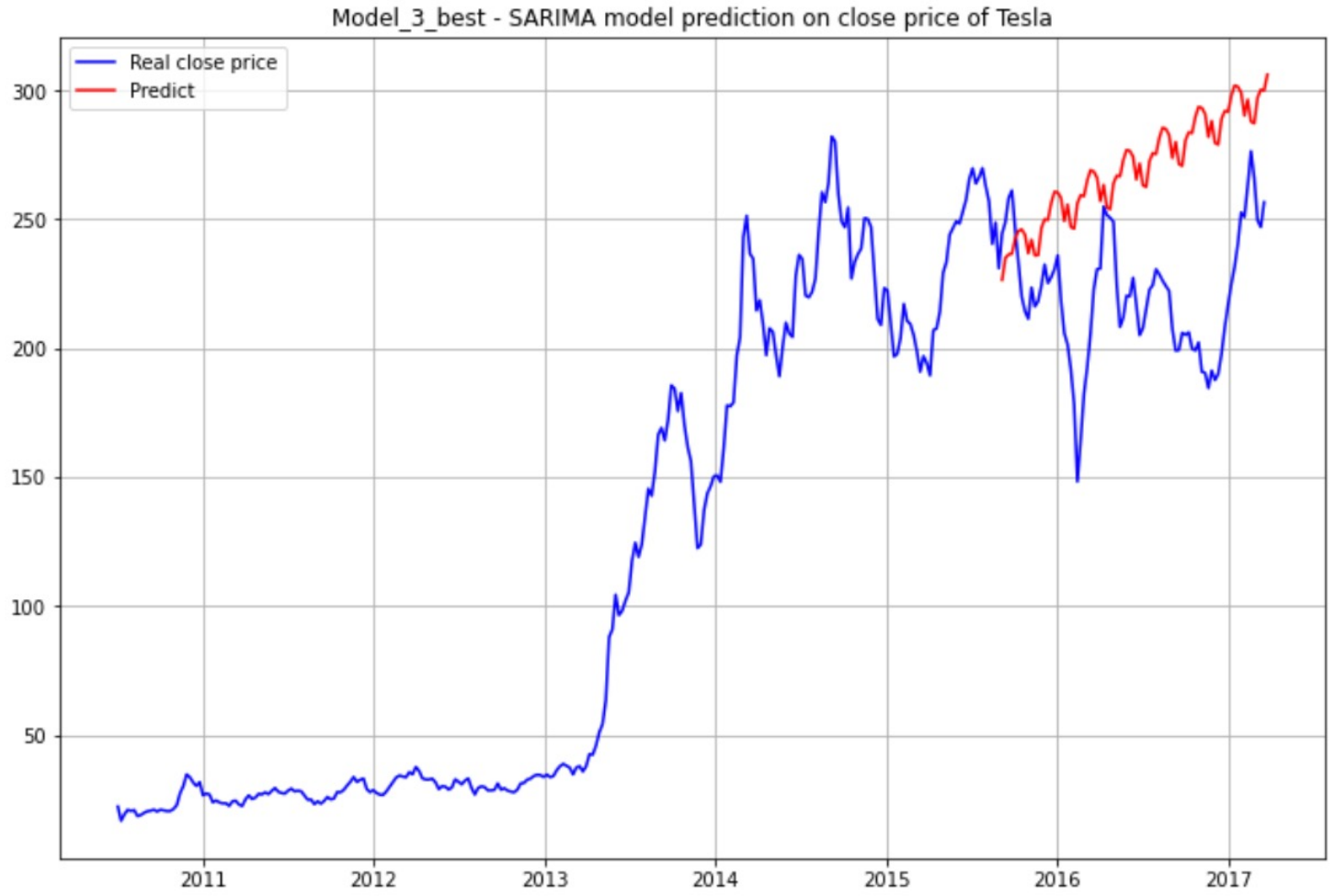
```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                               Close   No. Observations:                  287
Model:             SARIMAX(2, 0, 1)x(2, 1, [], 11)   Log Likelihood                -969.047
Date:                             Thu, 28 Apr 2022   AIC                           1952.094
Time:                                     23:08:26   BIC                           1977.437
Sample:                                 07-04-2010   HQIC                          1962.264
                                      - 12-27-2015
Covariance Type:                              opg
==========================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
intercept      0.2764      0.196      1.413      0.158      -0.107       0.660
ar.L1          1.7794      0.097     18.384      0.000       1.590       1.969
ar.L2         -0.7998      0.091     -8.771      0.000      -0.979      -0.621
ma.L1         -0.6886      0.124     -5.571      0.000      -0.931      -0.446
ar.S.L11      -0.5161      0.050    -10.286      0.000      -0.614      -0.418
ar.S.L22      -0.1221      0.040     -3.018      0.003      -0.201      -0.043
sigma2        64.3643      4.006     16.065      0.000      56.512      72.217
==========================================================================================
Ljung-Box (L1) (Q):                   0.80   Jarque-Bera (JB):                62.66
Prob(Q):                              0.37   Prob(JB):                         0.00
Heteroskedasticity (H):              34.49   Skew:                            -0.05
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.33
==========================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
1 model_3_best.plot_diagnostics(figsize=(15,8));
```



8.2 Prediction using model_3_best ¶

```
predict_3 = model_3_best.predict(start=270,end=351,dynamic=True)
plt.figure(figsize=(12,8))
plt.plot(close_week,"b-",label="Real close price")
plt.plot(predict_3,'r-',label='Predict')
plt.legend(loc='best')
plt.title("Model_3_best - SARIMA model prediction on close price of Tesla")
plt.grid(0.3)
```



8.3 Validation: mse

```
print("MSE of model_1:", mse(close_test
print("MSE of model_2_best:", mse(close_
print("MSE of model_3_best:", mse(close_
```

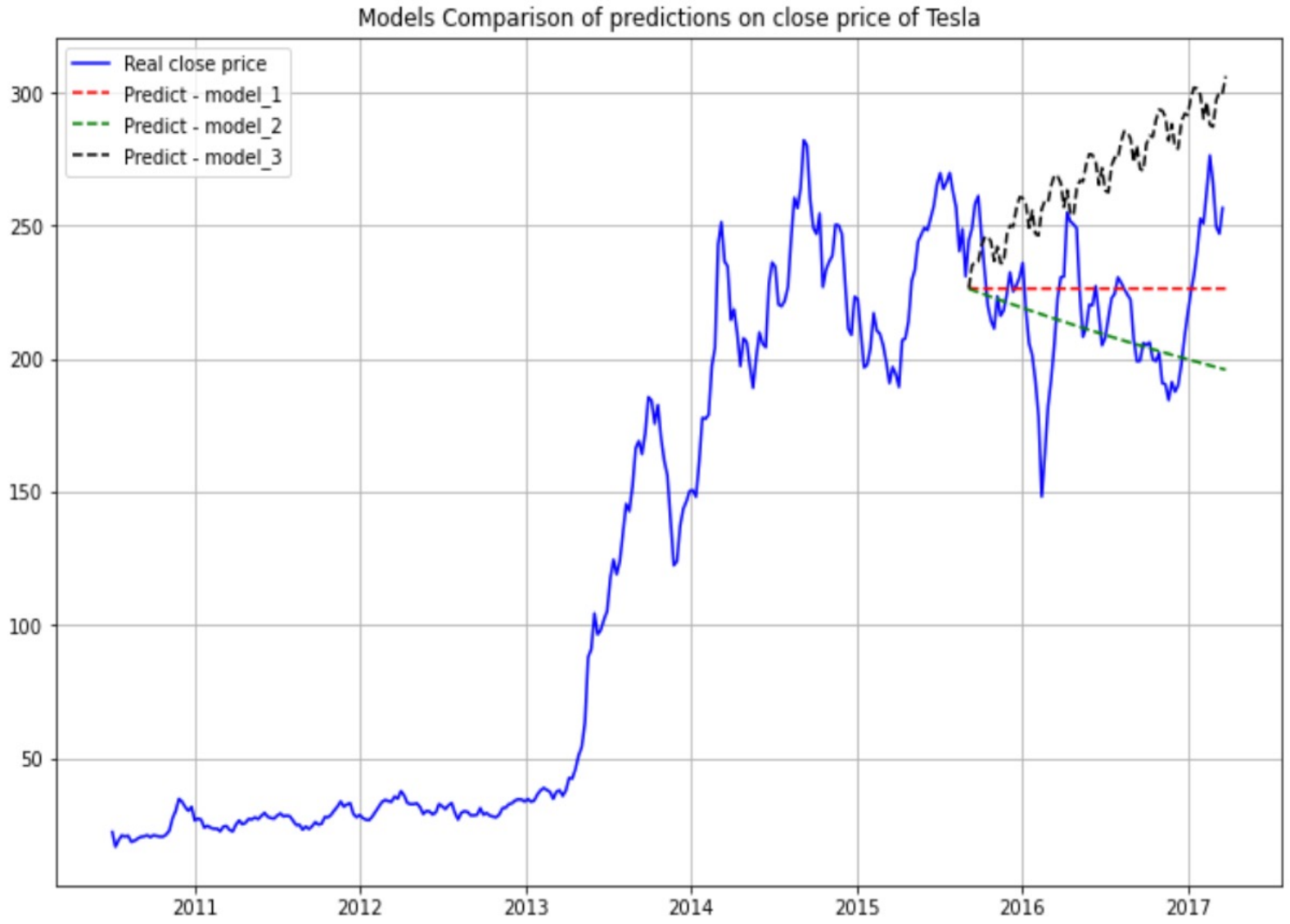
```
MSE of model_1: 50246.62748995037
MSE of model_2_best: 61624.472763307844
MSE of model_3_best: 280403.85610155656
```

Much worse...with higher mse.

Summary - key findings

- From model_3 prediction plot, it's clearly worse than other models, although some part of trends is similar.
- It shows that adding the seasonal effects in this dataset is misleading the predicted (increasing) trends which is far from real data.
- Comparing with model_1 and model_2, model_3 is much complicated but has much larger MSE.
- Model_1's order is much easier got by just checking acf / pacf than getting from pmd arima modeling (model_2, model_3).

In this case, from the comparing of ARIMA, pmd-ARIMA, pmd-SARIMA models, the close price prediction will prefer the model_1 ARIMA model as at least final choice (may not the best choice).



Insights and Suggestions

- The first date of stock price is at IPO date (2010-6-29), and normally a new stock will have big fluctuations in price after quite few periods. Tesla's stock prices stay stable (<$50) for over 2 years.
- Then, very huge increase with fluctuations come after 2013. It is obviously that, during 2013, there are business actions boosting its stock price, comparing with time sequence factors.

1. **What if remove the first "stable" 2 years in the whole data? The train-set using 2013-2016 instead of 2010-2015? Will it affect on results of model_3?**

2. **If no changes in original dataset, whether deep learning models will more accurate on predictions? I guess LSTM should get a try.**

Follow-up 1 : change train data for SARIMA model

```
new_train = close_week['2013':'2016']
```

```
len(new_train)
```

```
208
```

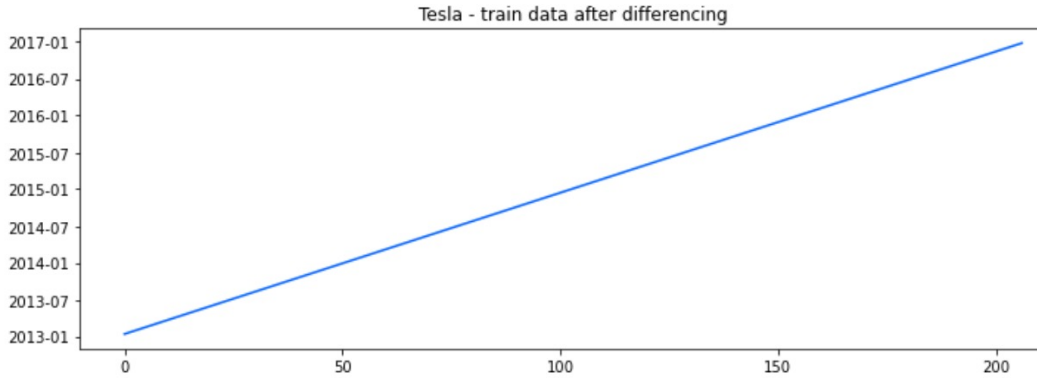
```
plt.figure(figsize=(18,6))
plt.plot(new_train,'b-')
plt.title("Tesla - Close price - 2013:2016")
plt.ylabel("Stock price")
plt.grid()
```

11.1.1 transfer to stationarity by differencing

```
new_train_diff = new_train.diff()
new_train_diff = new_train_diff.dropna()
```

```
plt.figure(figsize=(12,4))
plt.plot(new_train_diff.index)
plt.title("Tesla - train data after differencing")
```

```
Text(0.5, 1.0, 'Tesla - train data after differencing')
```




```
adfuller(new_train_diff)
```

```
(-11.120357537288317,
 3.4783714548659886e-20,
 0,
 206,
 {'1%': -3.4624988216864776,
  '5%': -2.8756749365852587,
  '10%': -2.574304154962767 7},
 1428.0276199980033)
```

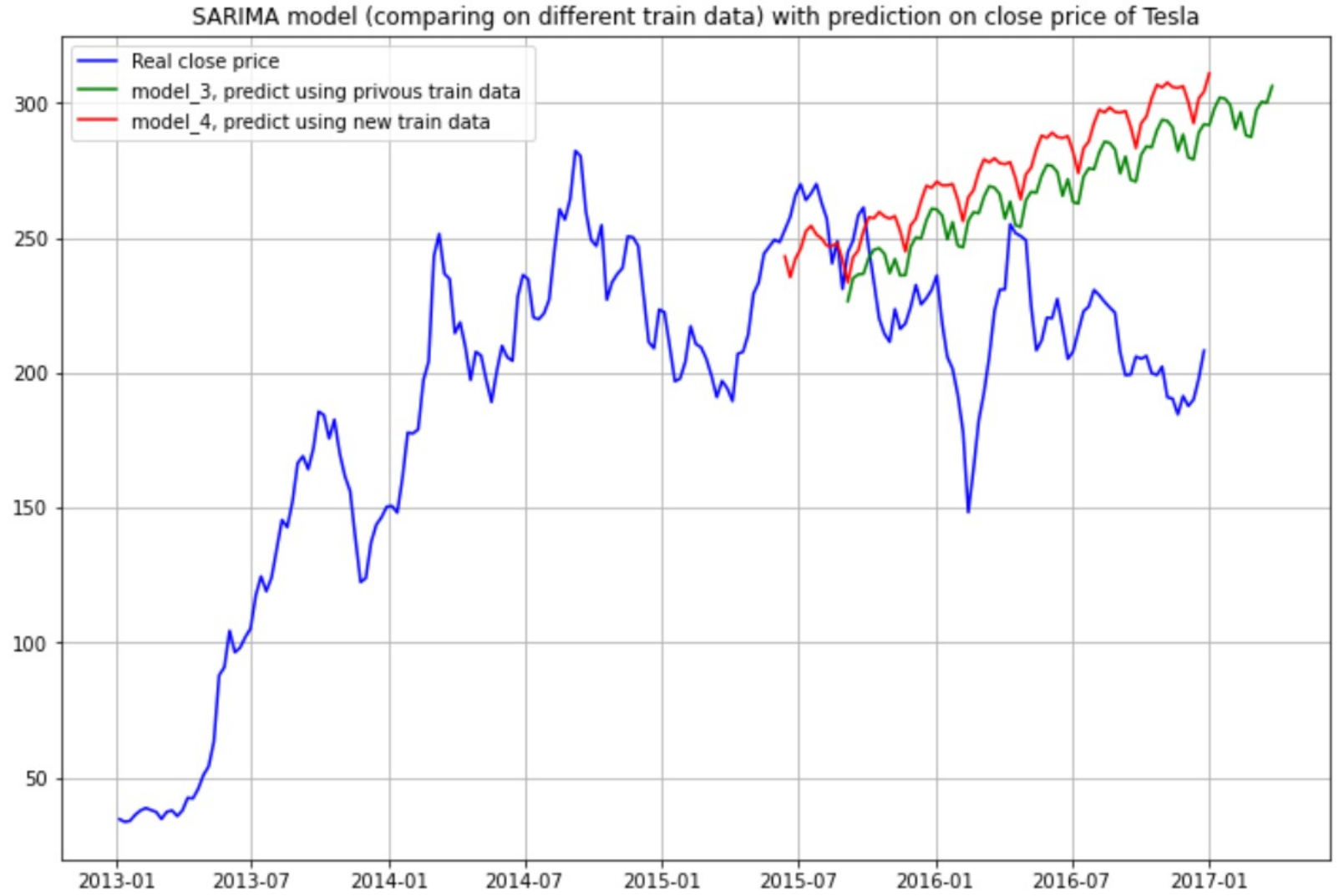
p-value < 0.05, after differencing, it is stationarity.

Model_4, SARIMA, order=(2,0,0), Seasonal order=(2,1,0,11)

```
previous MSE: 280403.85610155656
new Mse: 314807.81882714224
```

Changing train-data brings No improving!

```
Previous sarima model order: (2, 0, 1)
Previous sarima model seasonal order: (2, 1, 0, 11)
New sarima model order: (2, 0, 0)
New sarima model seasonal order: (2, 1, 0, 11)
```



Follow-up 2: Deep learning LSTM model

Model preparing

- Data preprocessing (Min-max)
- Split train/ test dataset with x, y
- Transform to deep learning network data shape…

```
import math
train_data_len = math.ceil(len(data)*0.8)
train_data_len
```

```
1354
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
trans_data = scaler.fit_transform(data)
trans_data
```

```
array([[0.02993635],
       [0.02971433],
       [0.02279455],
       ...,
       [0.88784039],
       [0.91122698],
       [0.9091918 ]])
```

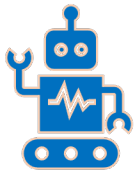
```
train_data = trans_data[0: train_data_len,:]
train_x = []
train_y = []

for i in range(60,len(train_data)):
    train_x.append(train_data[i - 60:i,0])
    train_y.append(train_data[i,0])
    if i<60:
        print(train_x)
        print(train_y)
        print()
```

```
train_x, train_y = np.array(train_x), np.array(train_y)

train_x = np.reshape(train_x, (train_x.shape[0],train_x.shape[1],1))
train_x.shape
```

```
(1294, 60, 1)
```

```
test_data = trans_data[train_data_len-60:,:]
test_x = []
test_y = data[train_data_len:,:]
for i in range(60,len(test_data)):
    test_x.append(test_data[i-60:i,0])
test_x = np.array(test_x)
test_x = np.reshape(test_x, (test_x.shape[0],test_x.shape[1],1))
test_x.shape
```

```
(338, 60, 1)
```



LSTM model building

```
1 import tensorflow as tf
2 import keras
3 from keras.models import Sequential
4 from keras.layers import Dense, SimpleRNN, LSTM, Activation, Dropout
```

```
1 model_5 = Sequential()
2 model_5.add(LSTM(64,input_shape=(train_x.shape[1],1)))
3 model_5.add(Dense(32))
4 model_5.add(Dense(1))
5 model_5.compile(optimizer='adam', loss='mean_squared_error')
6 model_5.fit(train_x, train_y,batch_size=1, epochs=10)
```

```
Epoch 1/10
1294/1294 [==============================] - 21s 15ms/step - loss: 0.0018
Epoch 2/10
1294/1294 [==============================] - 19s 14ms/step - loss: 9.2596e-04
Epoch 3/10
1294/1294 [==============================] - 18s 14ms/step - loss: 7.2786e-04
Epoch 4/10
1294/1294 [==============================] - 19s 14ms/step - loss: 5.3970e-04
Epoch 5/10
1294/1294 [==============================] - 19s 15ms/step - loss: 6.0790e-04
Epoch 6/10
1294/1294 [==============================] - 19s 15ms/step - loss: 5.2898e-04
Epoch 7/10
1294/1294 [==============================] - 19s 14ms/step - loss: 4.8845e-04
Epoch 8/10
1294/1294 [==============================] - 17s 13ms/step - loss: 4.6672e-04
Epoch 9/10
1294/1294 [==============================] - 17s 13ms/step - loss: 4.9823e-04
Epoch 10/10
1294/1294 [==============================] - 16s 12ms/step - loss: 4.5805e-04

<keras.callbacks.History at 0x7fce154cecd0>
```

```
1 model_5.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_4 (LSTM)                (None, 64)                16896

 dense (Dense)                (None, 32)                2080

 dense_1 (Dense)              (None, 1)                 33

=================================================================
Total params: 19,009
Trainable params: 19,009
Non-trainable params: 0
```

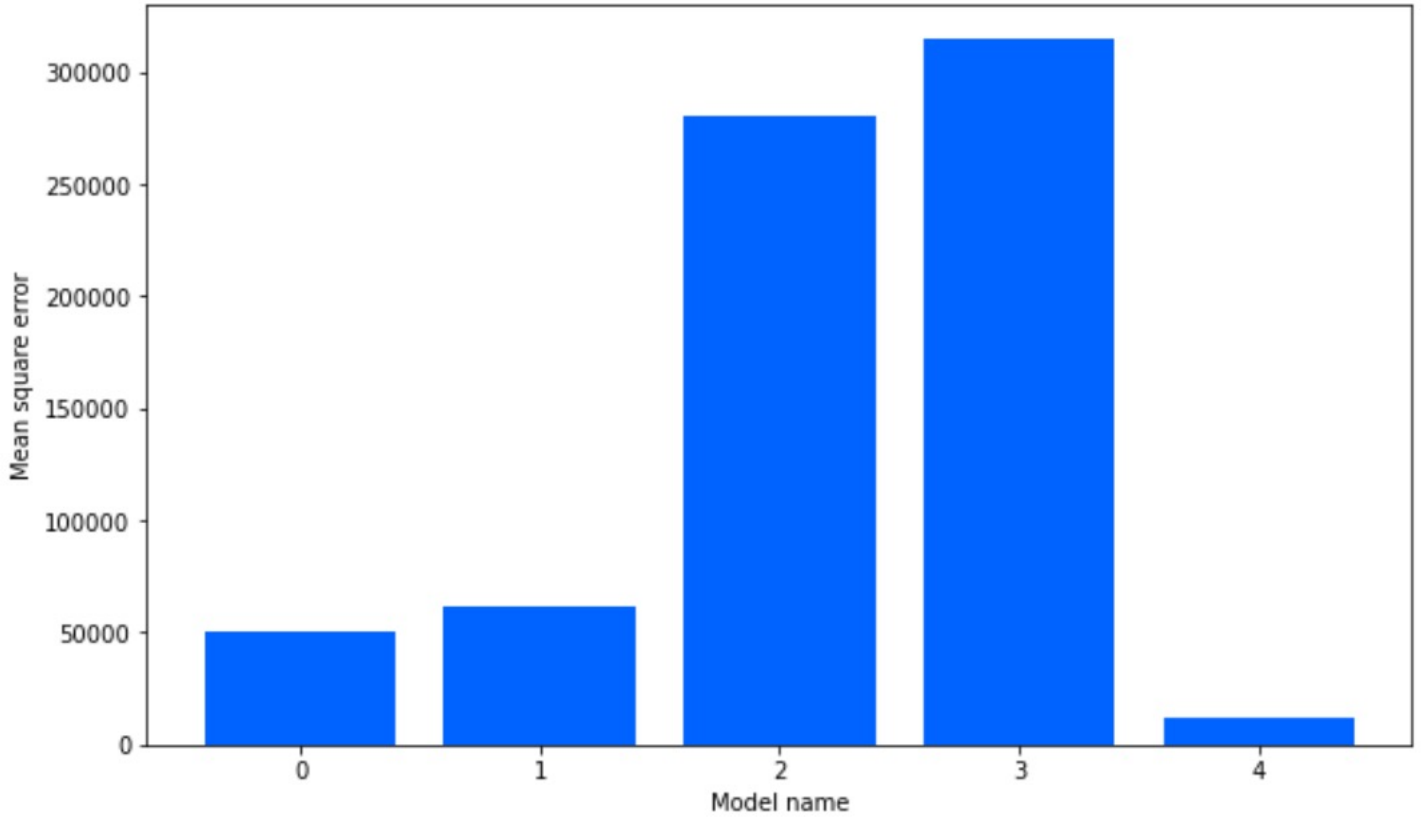
LSTM model prediction

Significantly improved!

```
plt.figure(figsize=(20,12))
plt.title("LSTM model - Close stock price of Tesla")
plt.plot(train_df['Close'],'b-')
plt.plot(valid_df[['Close','Predict']],linewidth = 4)
plt.xlabel("Time")
plt.ylabel("Close price")
plt.legend(['Train','Test','Predict'],loc='best');
```

Final conclusion

	Model name	Mean square error
0	ARIMA model	50,246.6
1	PMD-ARIMA model	61,624.5
2	SARIMA model	280,403.9
3	(NEW)SARIMA model	314,807.8
4	LSTM model	11,475.6



Conclusion:

Clearly, **LSTM** will be the best choice in prediction of stock price via 5 models comparing.
Predictions(LSTM) are very similar to real data with the lowest MSE.
Of course, as a disadvantage of deep learning models, it often occurs overfitting.