Time series datasets with different types of machine learning

SUMMARY

- Datasets : Tesla (Ticker symbol = “TSLA”)
- From Yahoo Finance
- Range from "2012-05-01” to "2022-05-01“
- Feature to be tested : Adj Close Price
- Total Rows : 2304
- Total Column: 1

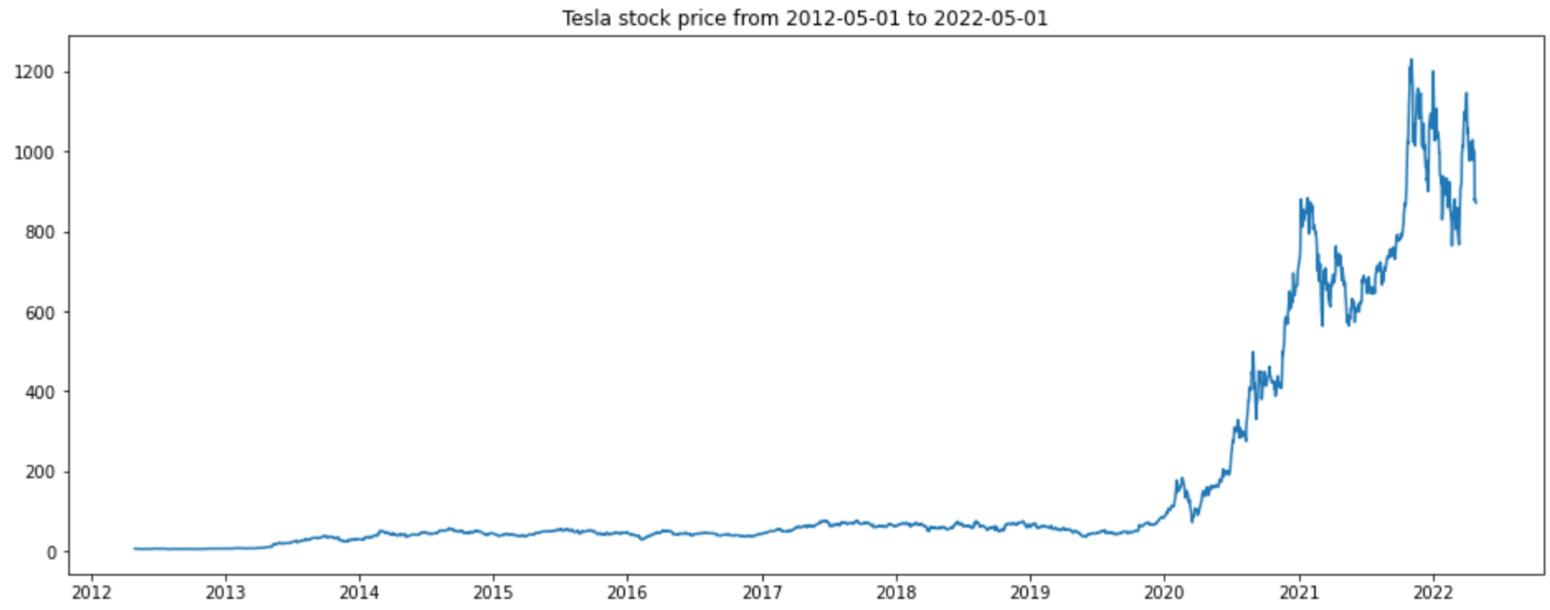
Selection of datasets

- The main reason of selecting this dataset was Tesla stock price went from $6 to $870, this extreme changes in the dataset will see whether the ML can accurately predict the price.

- Sample of the dataset

	Close
Date	
2012-04-30	6.626000
2012-05-01	6.756000
2012-05-02	6.788000
2012-05-03	6.492000
2012-05-04	6.366000
...	...
2022-04-25	998.020020
2022-04-26	876.419983
2022-04-27	881.510010
2022-04-28	877.510010
2022-04-29	870.760010

Chart of Tesla from "2012-05-01" to "2022-05-01"



Input Feature

- The price itself is not enough for prediction, additional indictors will be used for this input models:

- Moving Average (with periods 5, 10, 20, 50, 100 200.)

- Bollinger bands
- 20 periods, 2 standard deviations
- 20 periods, 1 standard deviation
- 10 periods, 1 standard deviation
- 10 periods, 2 standard deviations

OBJECTIVE

- To predict the close price of 5 days in the future.

MODEL TO BE USE:

- **Linear regression**
- **Random forest**
- **Gradient boosting regressor**
- **K Nearest Neighbors**
- **Neural network - Artificial Neural Network**
- **Linear regression with Bagging**
- **Linear regression with Adaboost**

MODE TO DETERMINE SUCCESS

- We will be using “Mean Absolute Error” for this test.

Why Mean Absolute Error

- It is the difference between the predicted value and real value.

OUTCOME

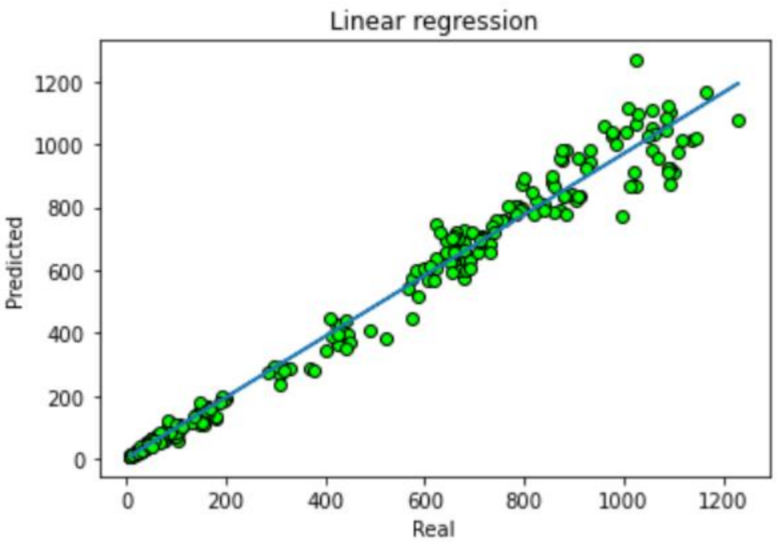
- The Lesser / Smaller the better.

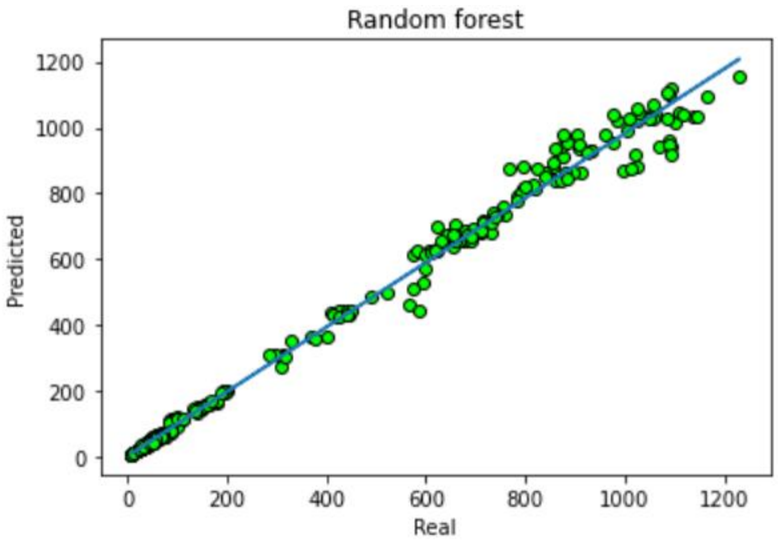
Test & Train Sizes

- X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=88)

- We perform a Test size of 30% and Train size of 70% with a random of 88

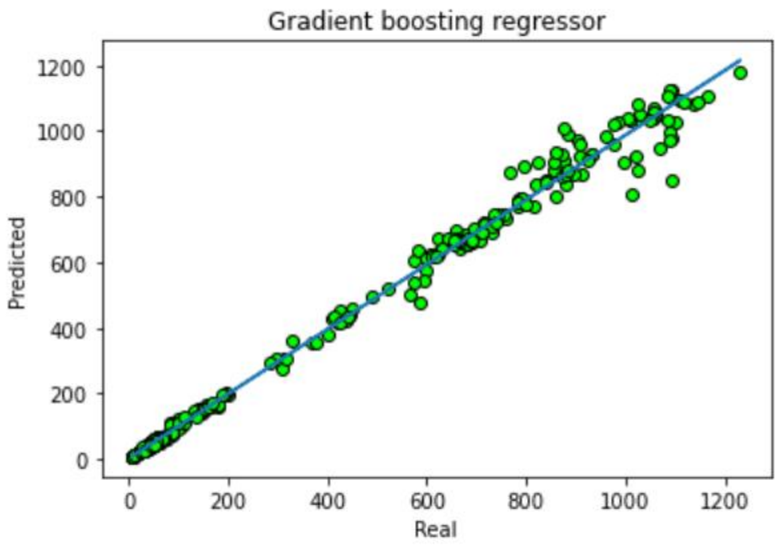
LINEAR REGRESSION & RANDOM FOREST

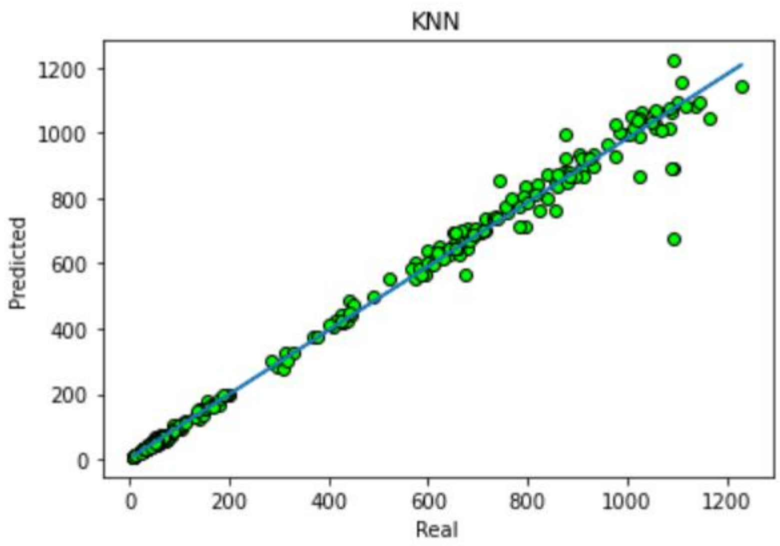




Mean Absolute Error	Mean Absolute Error
14.726219136464799	9.187292783722159

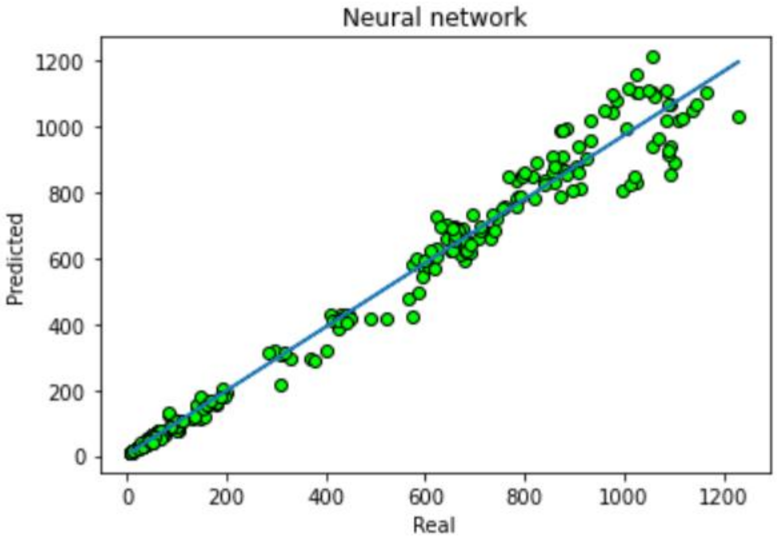
Gradient boosting regressor & K Nearest Neighbors

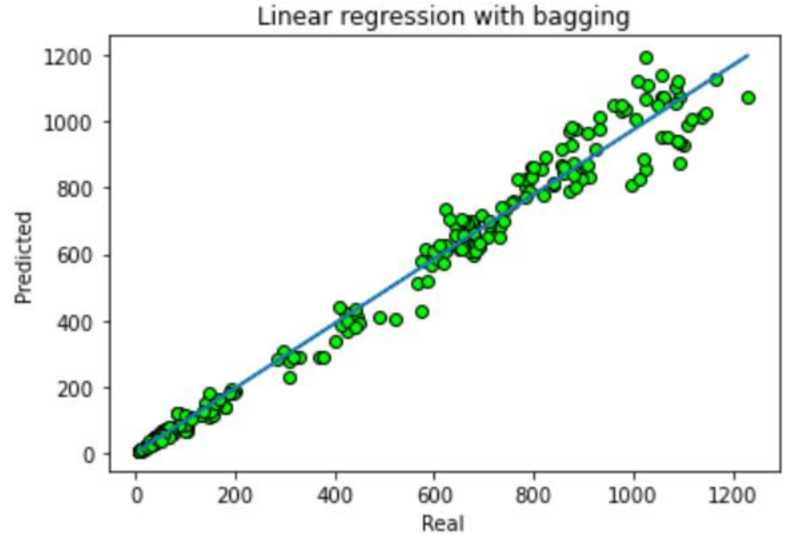




Mean Absolute Error	Mean Absolute Error
8.990123738174495	7.778635115981791

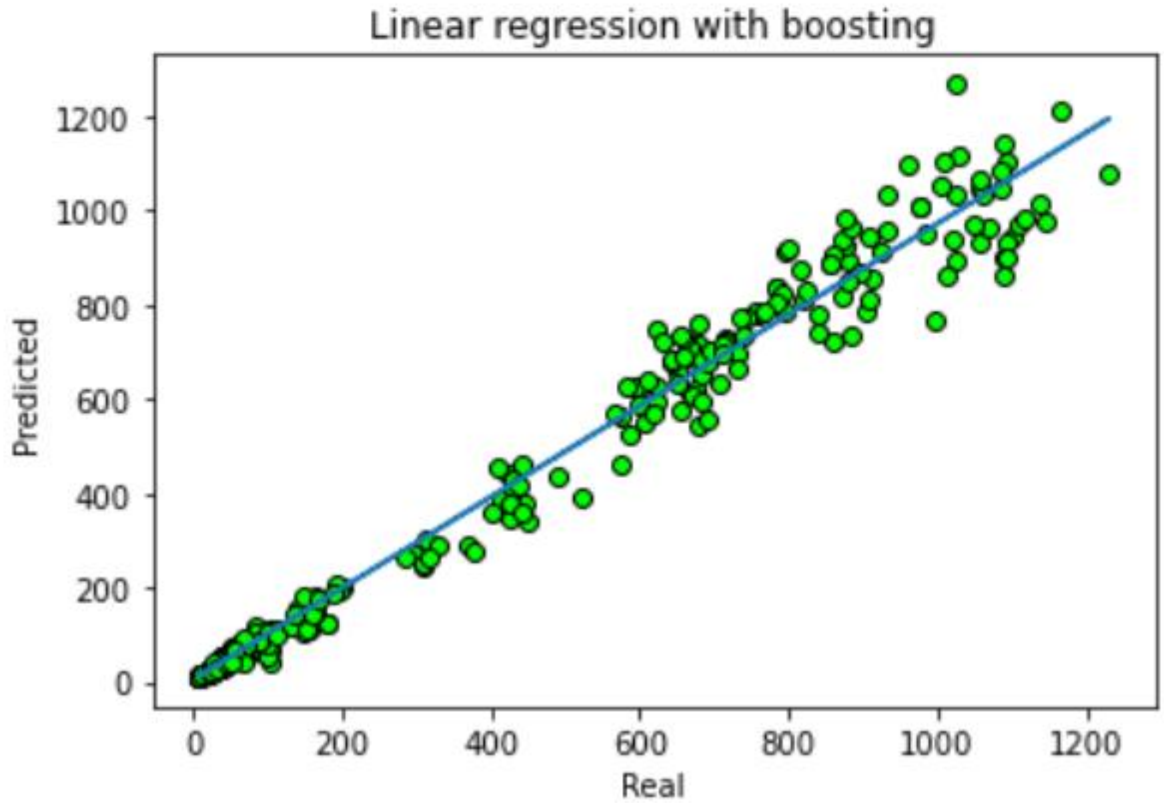
Neural network & Linear regression with Bagging





Mean Absolute Error	Mean Absolute Error
14.821284834868308	14.510198974367961

Linear regression with Adaboost



Mean Absolute Error
18.029284112810608

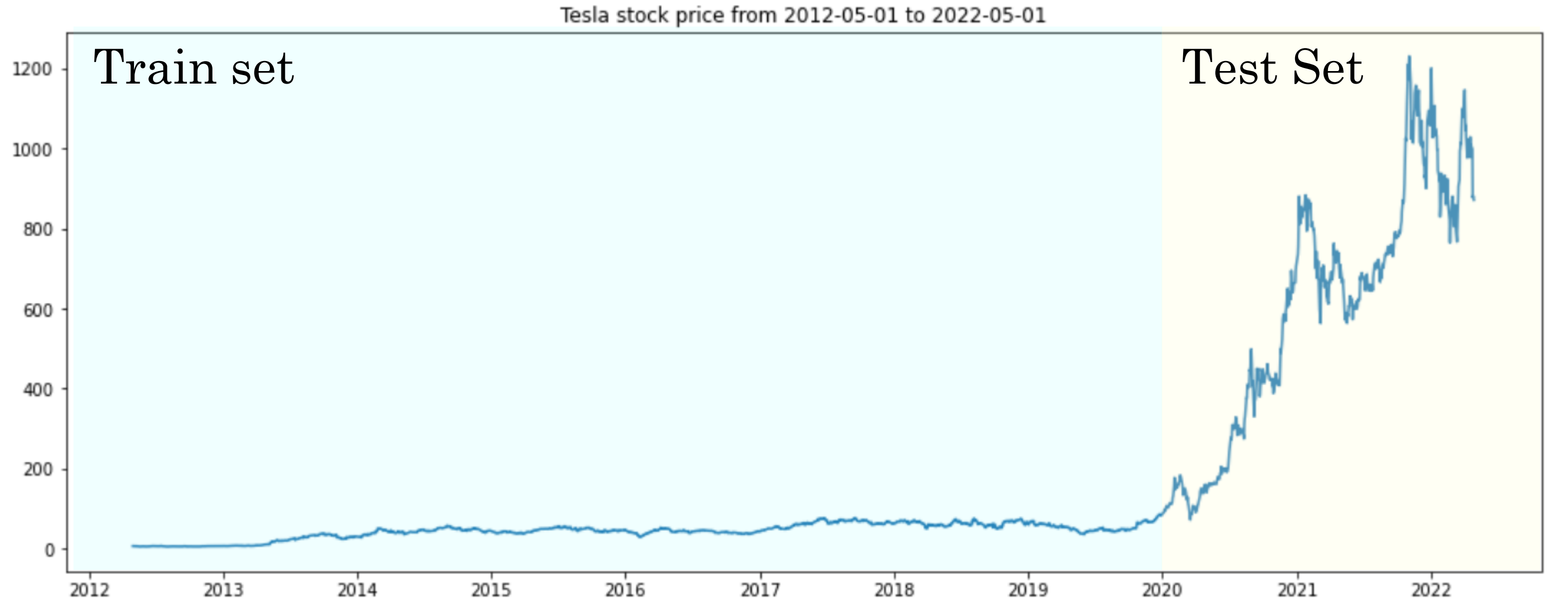
Summary of the test

Models	Mean Absolute Error
Linear regression	14.726219136464799
Random forest	9.187292783722159
Gradient boosting regressor	8.990123738174495
K Nearest Neighbors	7.778635115981791
Neural network - Artificial Neural Network	14.821284834868308
Linear regression with Bagging	14.510198974367961
Linear regression with Adaboost	18.029284112810608

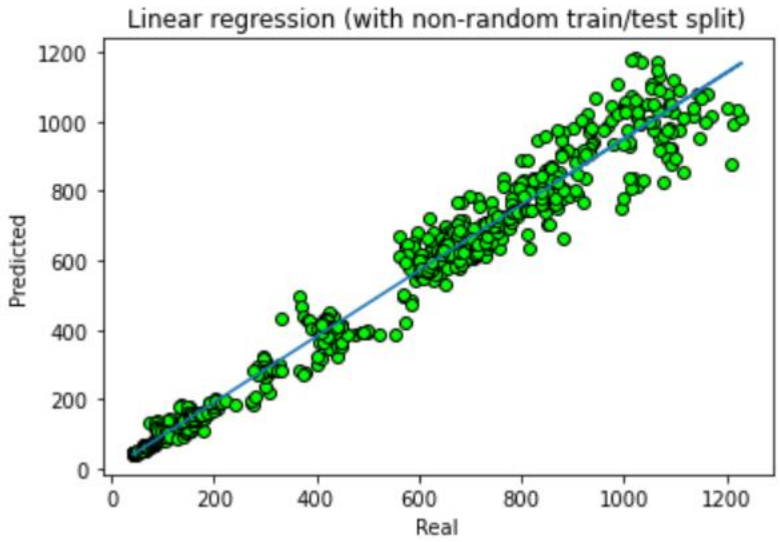
The Winner is **K Nearest Neighbors** (Or maybe not?)

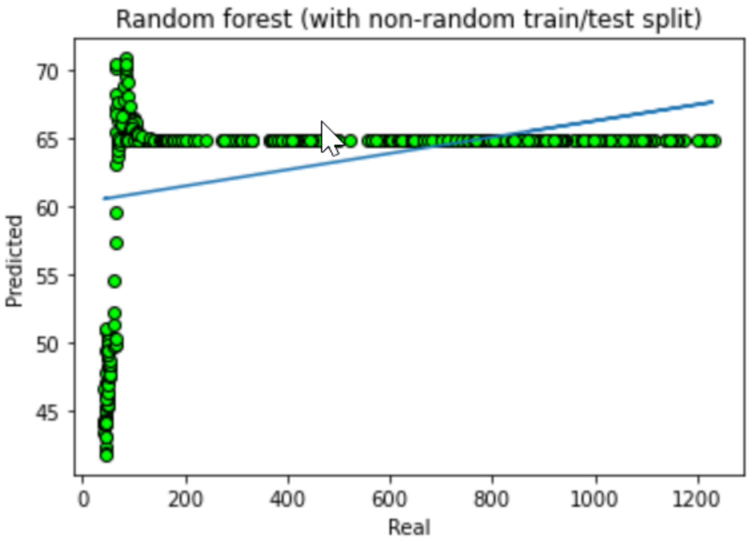
Test & Train Sizes without Random

- We perform a Test size of 30% and Train size of 70%



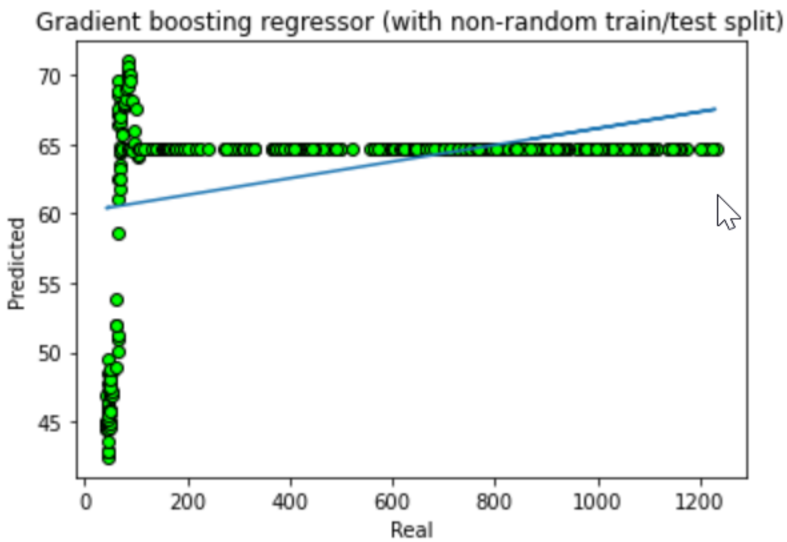
LINEAR REGRESSION & RANDOM FOREST

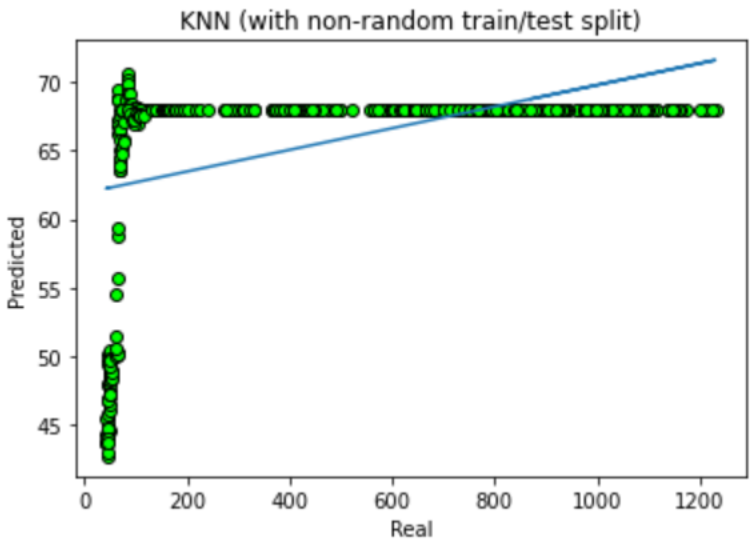




Mean Absolute Error	Mean Absolute Error
42.41960007910977	448.16292783049084

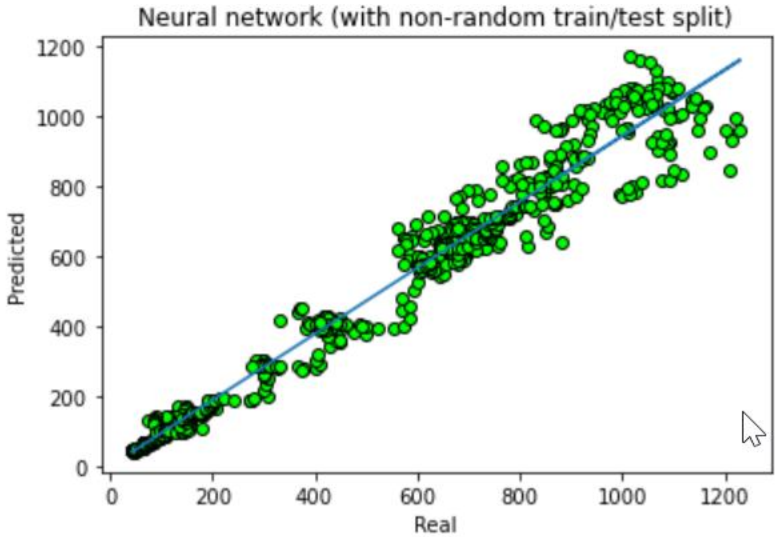
Gradient boosting regressor & K Nearest Neighbors

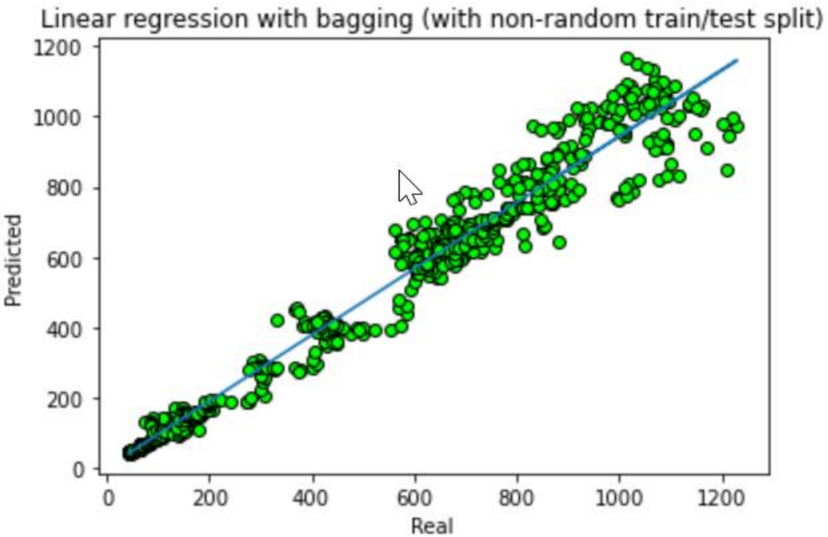




Mean Absolute Error	Mean Absolute Error
448.26785197108103	445.5859685256048

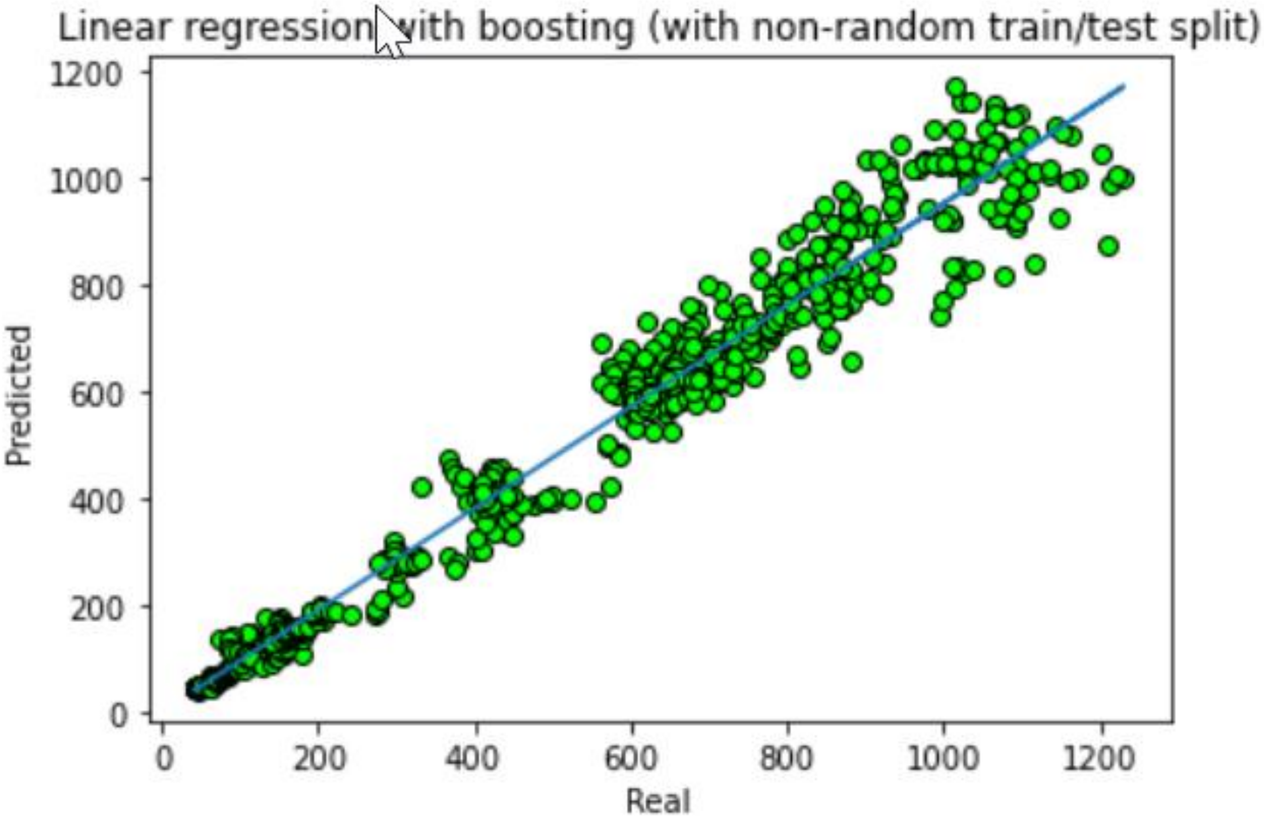
Neural network & Linear regression with Bagging





Mean Absolute Error	Mean Absolute Error
45.37511151052321	44.63833804555751

Linear regression with Adaboost



Mean Absolute Error
43.03194206392055

Summary of the test without random

Models	Mean Absolute Error
Linear regression	42.41960007910977
Random forest	448.16292783049084
Gradient boosting regressor	448.26785197108103
K Nearest Neighbors	445.5859685256048
Neural network - Artificial Neural Network	45.37511151052321
Linear regression with Bagging	44.63833804555751
Linear regression with Adaboost	43.03194206392055

Winner – Linear Regression ☺☺☺

Conclusion

- Historical data are not completely uncorrelated from each other so a random train/test split may be wrong.

- Understanding which ML model is suitable for the datasets is important to achieve the outcome.

ARIMA Model

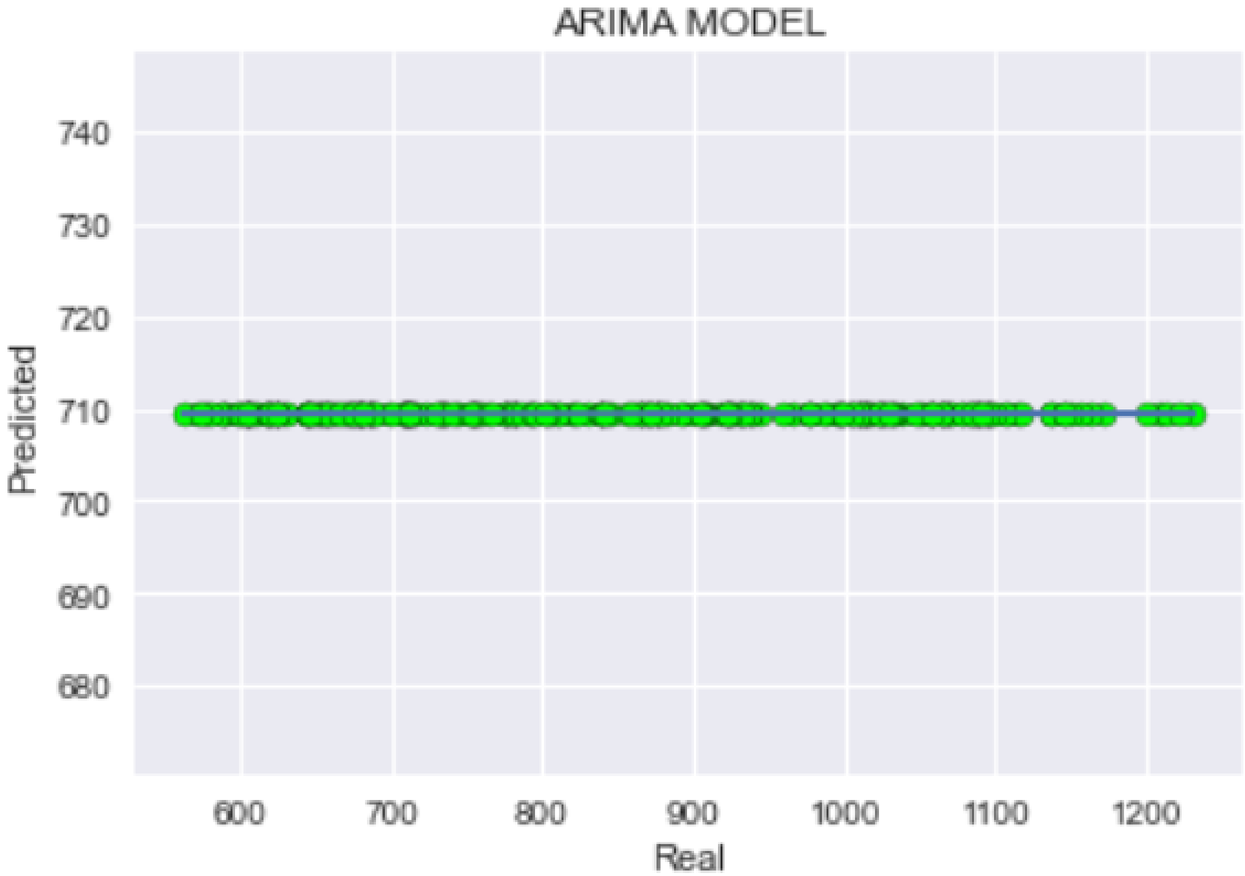
	model	AIC
0	ARIMA (5 1 4)	15732.194124
1	ARIMA (5 1 5)	15733.864721
2	ARIMA (4 1 5)	15735.685310
3	ARIMA (3 1 2)	15741.929749
4	ARIMA (3 1 3)	15745.416404
...	...	...
31	ARIMA (1 1 0)	15841.852618
32	ARIMA (0 1 2)	15842.762396
33	ARIMA (1 1 2)	15842.947490
34	ARIMA (1 1 1)	15843.082463
35	ARIMA (0 1 0)	15852.610959

The best fit are
ARIMA (5,1,4)
ARIMA(5,1,5)

I also use ARIMA(0,1,0)
Computationally, the lower the p and q, it will reduce the complexity cost

Summary:
ARIMA (5, 1, 5)
5 = Auto-Regressive Parameters
1 = the difference between response variable data
4 = Moving Average Parameters

ARIMA Chart


ARIMA MODEL
740
730
720
710
700
690
680
Predicted
600
700
800
900
1000
1100
1200
Real

Summary of the test

Models	Mean Absolute Error
ARIMA (5, 1, 4)	174.5870726063732
ARIMA (5, 1, 5)	174.64176843973715
ARIMA (0, 1, 0)	177.0600363110739