Vegetable Price Analysis

[A PROJECT ON Data Analytics and Visualization]

PROJECT REPORT

Submitted by:

ANMOL SAGAR (ID: 11840180)
MANU E (ID: 11840700)
PRINCE KUMAR PANSARI (ID: 11840860)
SAPTARSHI MUKHERJEE (ID: 11740860)
SIRAM NIKHIL (ID: 11841090)

Supervised by:

Dr. GAGAN RAJ GUPTA

INDIAN INSTITUTE OF TECHNOLOGY BHILAI

DS250

NOVEMBER 2020

CONTENTS

CODE and DATA (GITHUB REPOSITORY):

https://github.com/siramk/DS250_PROJECT_VEGETABLE_PRICE_ANALYSIS

Data Collection

Vegetable Data

- The data is available in XML and CSV formats in data.gov.in.
- It is present year-wise, so we have downloaded data files manually as there is no API for the data we require.
- The data files are very huge, each 30-50 MB of size. So opening in editors or IDE's is not possible. So we had to open the files with lightweight editors like vim and vi to check the XML structure.
- We wrote XML parsers for two different types of xml structures that are obtained in data and stored the final data in CSV format. We used the "xml" module in python to complete this task.
- There are a total of 542 districts and 2024 markets in the data that we had collected, each with their own statistics. So the primary challenge was to choose one market out of 2024 markets available so as to fix the starting point.
- Our aim in choosing the market was to maximize the data that we can perform analysis on, hence we filtered the markets having more than 500 days of data for at least 8 different vegetables each, programmatically. From the filtered cities we selected **Surat** as the market that we will be performing analysis on.
- As the data of few vegetables seemed to be sparse i.e missing on many days even after filtering, we searched at the source from where data.gov.in is getting information from, i.e. http://agmarknet.gov.in/. Luckily we got more data there.
- The data is available in the HTML format on that website, so we used BeautifulSoup module in python to do HTML parsing and stored the data in CSV format.
- Final format of data that we are working on is a CSV file containing the statistics of a particular vegetable at the chosen market of Surat.

Weather Data

- The data of weather is collected using API requests from https://www.worldweatheronline.com/
- We obtained the JSON response from the API and we parsed it with the help of JSON parser and stored the data in CSV format.
- Final format of weather data that we are working on is a CSV file which contains the weather data of Surat from 2010 to 2020.

Sources of data

1. https://data.gov.in/search/site?query=vegetable+prices

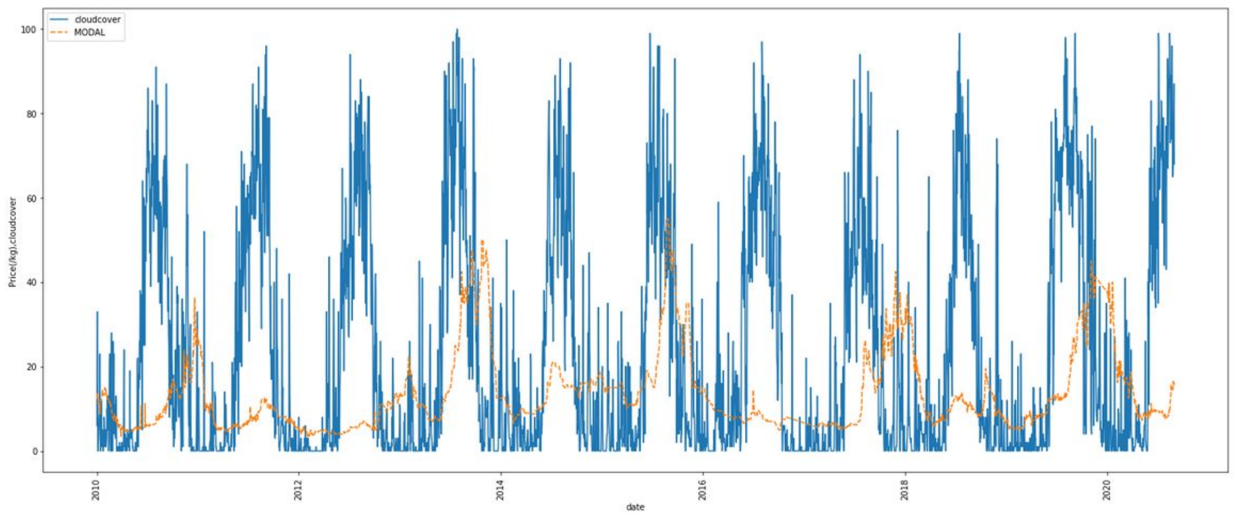
2. http://agmarknet.gov.in
3. https://www.worldweatheronline.com/

Data organization and cleaning

- The data collected during the data collection contains many irrelevant columns. So, we have removed all unnecessary columns. We also removed the min price and the max price. We are only left with the date and the modal price of the vegetable with our chosen market, i.e Surat.
- The Modal price in the data collected is the price of the vegetable per quintal. We are typecasting that to integers.
- We converted the date into a datetime object and we have indexed the data on the basis of that.
- We have done the interpolation on our data to estimate unknown values that lie between the known values. The chosen method for the interpolation is linear. After doing this we have data from the year 2010 to 2020 for the selected vegetable and market.
- We observed a sudden high increase in price at a few points in data. As they may affect the overall performance of the model, we implemented an interesting method to remove them. We calculated mean and standard deviation of the 7 day window and replaced the points whose z score is greater than 3 i.e whose value is at least 3 standard deviations far from mean.

EDA

EDA with weather data



The above graph shows the relationship between cloud-cover (the blue curve) and vegetable prices (the orange curve). We hereby observe that the cloud-cover shows a decent periodicity, which has almost twice the frequency as that of the cloud-cover. This gives an impression that the cloud-cover statistics is not likely to give a very clear indication of the modal prices of the vegetable, onion. Moreover, the peaks of the orange curve doesn't show a very constant shift from the nearest peak of the blue curve, this indicates that figuring out a general correlation between them is not very likely.

Spearman's correlation coefficient & p Value of onion

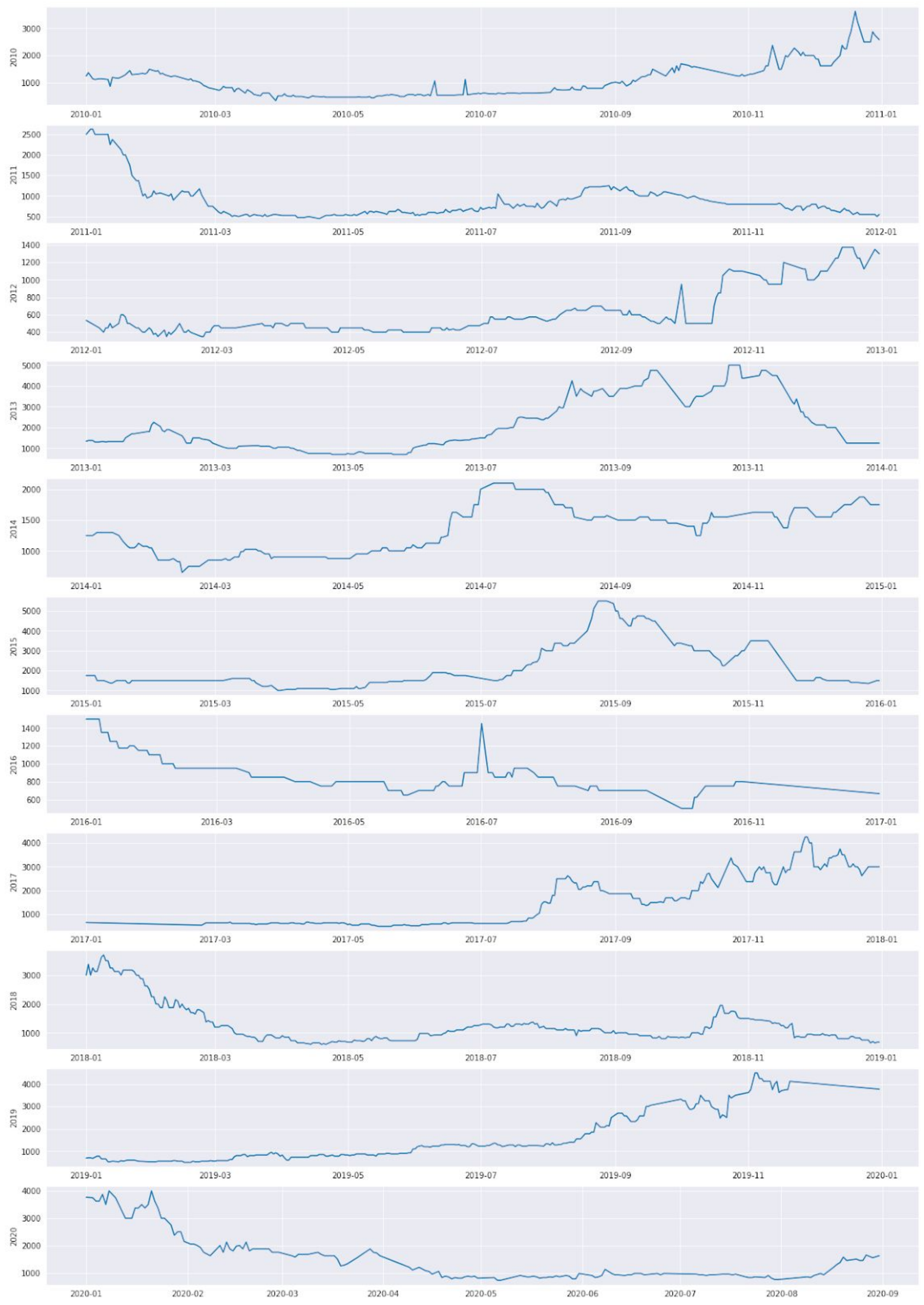
- Spearman's correlation coefficient of Modal price(per kg) and tempC: -0.2274891
  p Value =0.0000000
- Spearman's correlation coefficient of Modal price(per kg) and precipMM: 0.1077686
  p Value =0.0000000
- Spearman's correlation coefficient of Modal price(per kg) and humidity: 0.0381454
  p Value =0.0172629
- Spearman's correlation coefficient of Modal price(per kg) and cloud cover: 0.1467613
  p Value =0.0000000

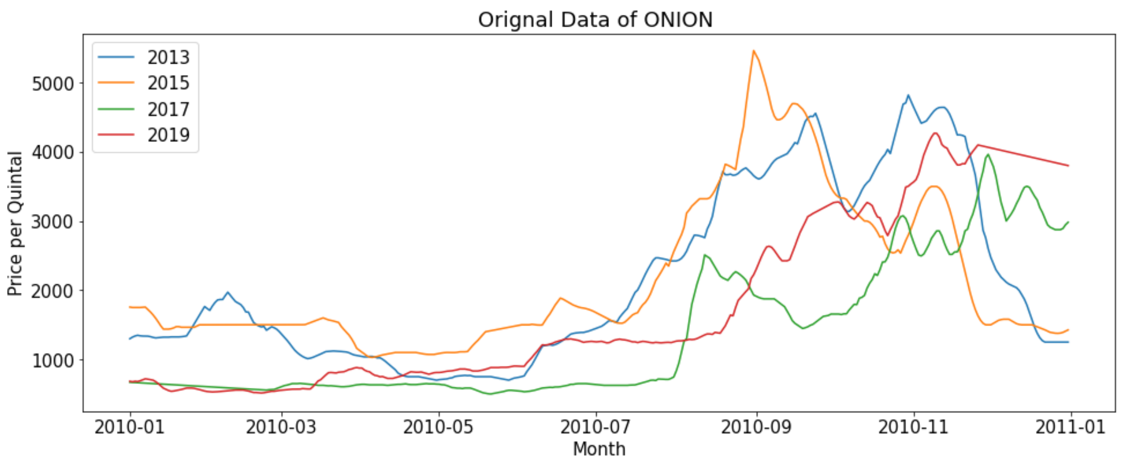
The above results show that there is no strong correlation between the modal prices of vegetables and the weather data.

Data visualization

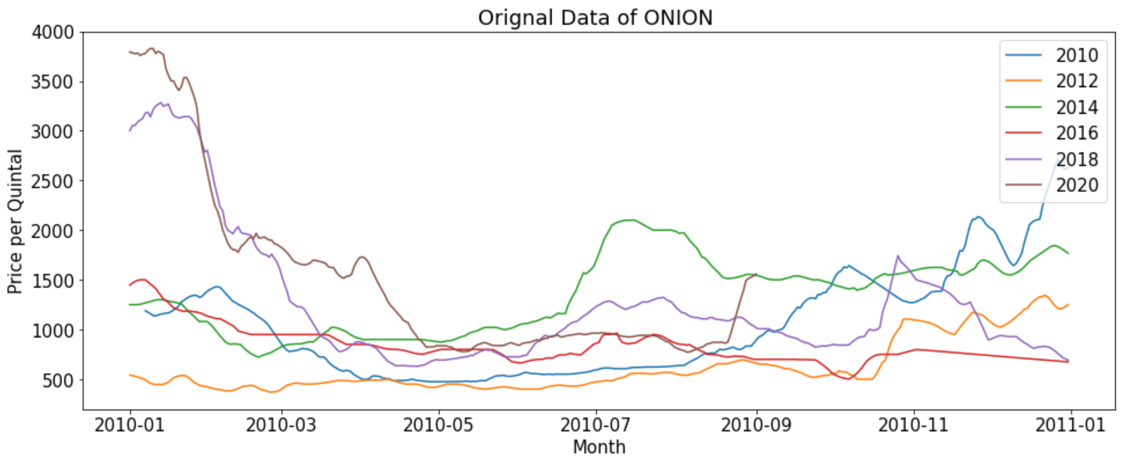
For visualization of onion data over the many years, we've used different types of a plot like a line plot, Scatter plot, Time Series Plot, Time Series Decomposition Plot, Histograms, Moving average plot, Box plots, Autocorrelation (ACF), and Partial Autocorrelation (PACF) Plot.

- First, we plotted a **line plot** for every year (2010-2020); from those plots, we observed that the alternate year plots are similar, then we have plotted the same data on different scales, **resampling** the same, taking into consideration the mean and associated details for the shorter periods.



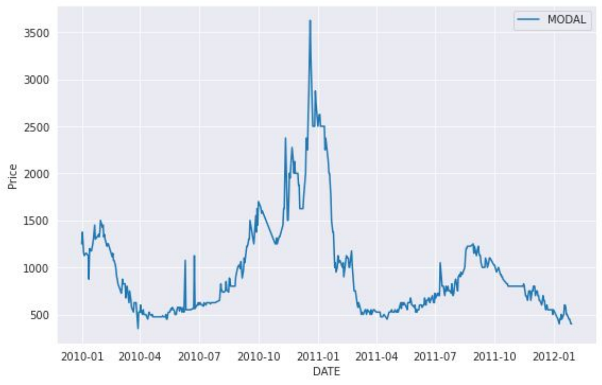


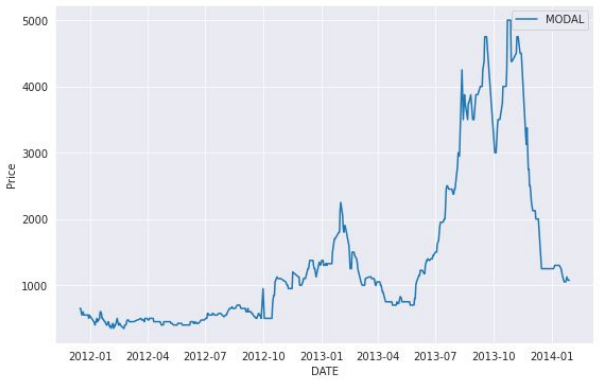
Orignal Data of ONION
2013
2015
2017
2019
Price per Quintal
5000
4000
3000
2000
1000
2010-01
2010-03
2010-05
2010-07
2010-09
2010-11
2011-01
Month

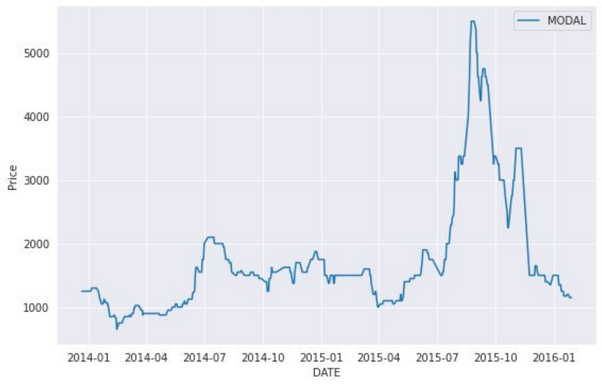


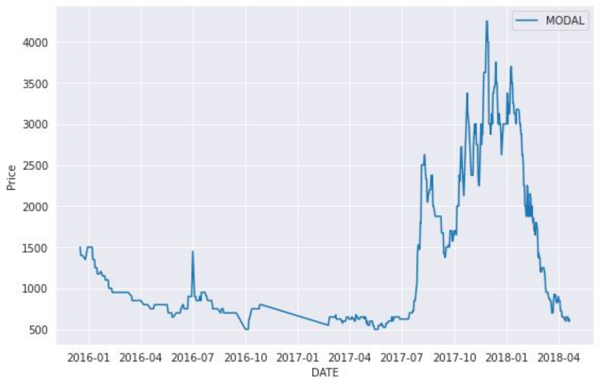
Orignal Data of ONION
2010
2012
2014
2016
2018
2020
Price per Quintal
4000
3500
3000
2500
2000
1500
1000
500
2010-01
2010-03
2010-05
2010-07
2010-09
2010-11
2011-01
Month

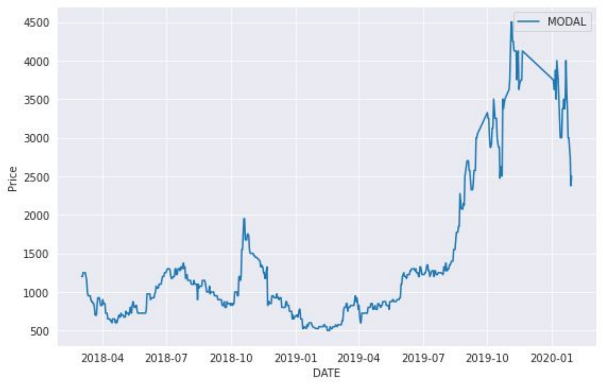
- For every two years, we've observed the pattern, it appears that they are largely similar, thereby depicting a decent periodicity on a biannual basis.

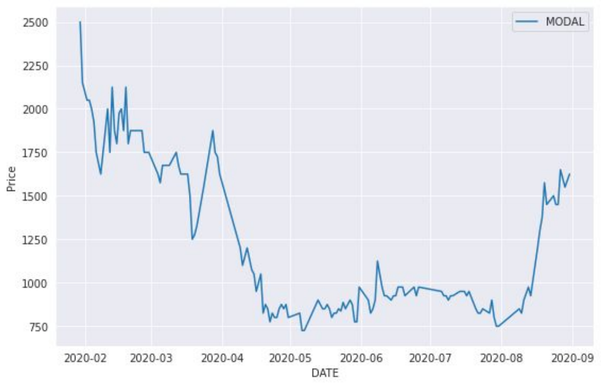




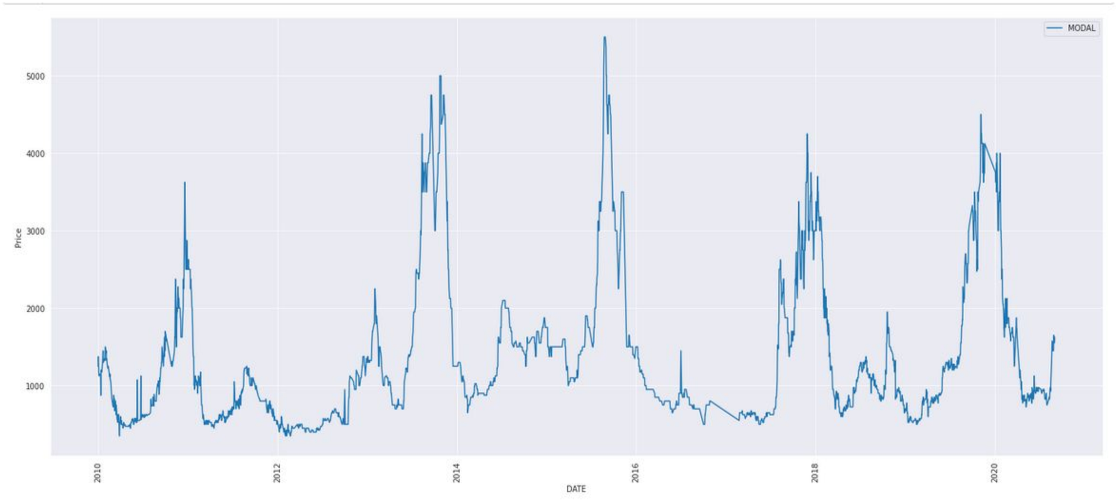




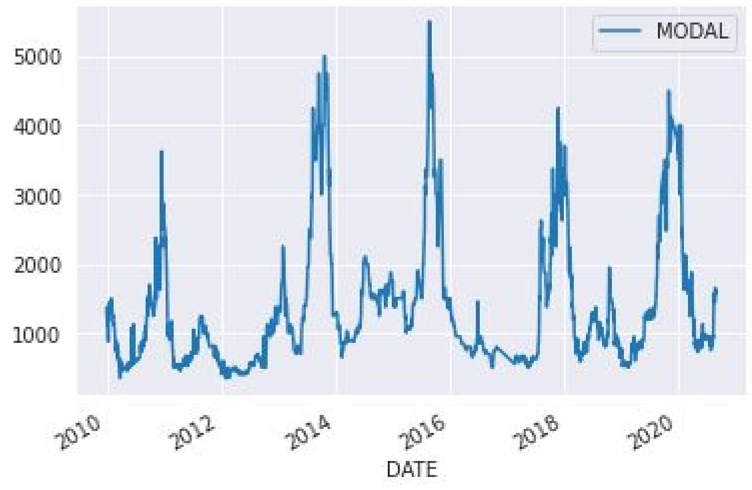


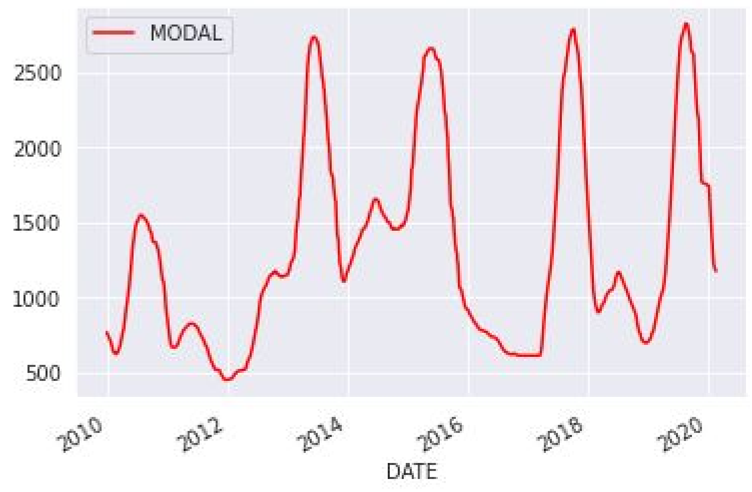


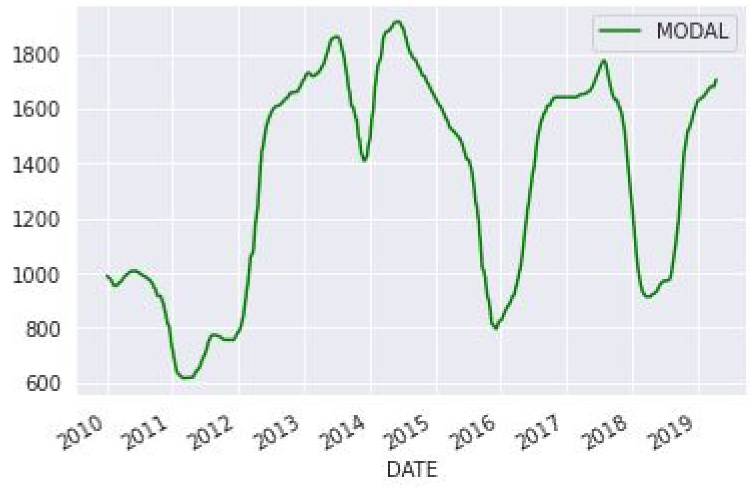
- Again we plot lineplot over 10 years, from these plots we conclude that the seasonality should be 730 days (2 years). We've plotted scatter plots for each year, then for two years, and then for 10 years so as to get more insights on the data.



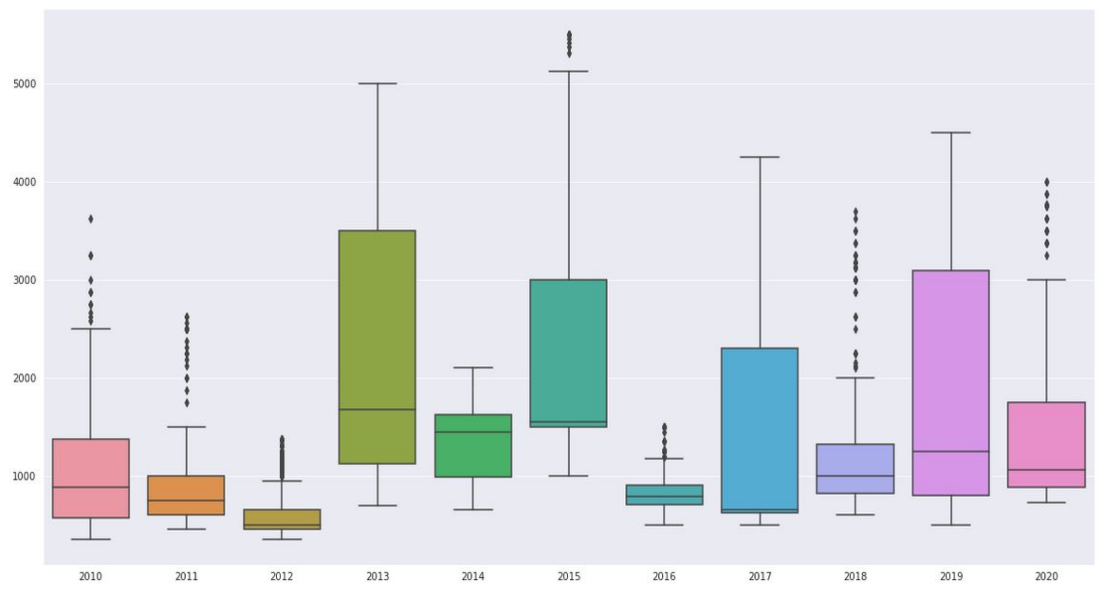
- While observing the year-wise plot we find that the prices of onion in alternate years show an upsurge towards the ending months in the city of Surat, somewhere attributing to the pattern of **rainfall** as per the observed weather data.
- The observed trend of increasing price with time strongly resembles the real statistics (even the currently witnessed trend), as obtained from other sources, thereby affirming that the trend has been captured correctly.
- The next crucial step was to use **Moving Average** with periods of 130 days and 365 days for providential visualisations.



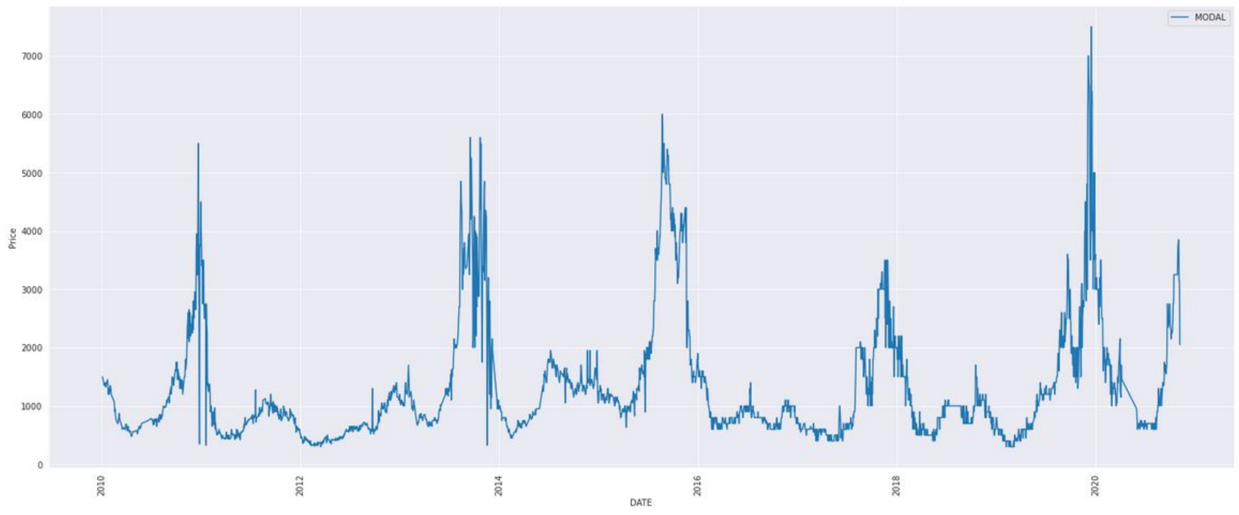




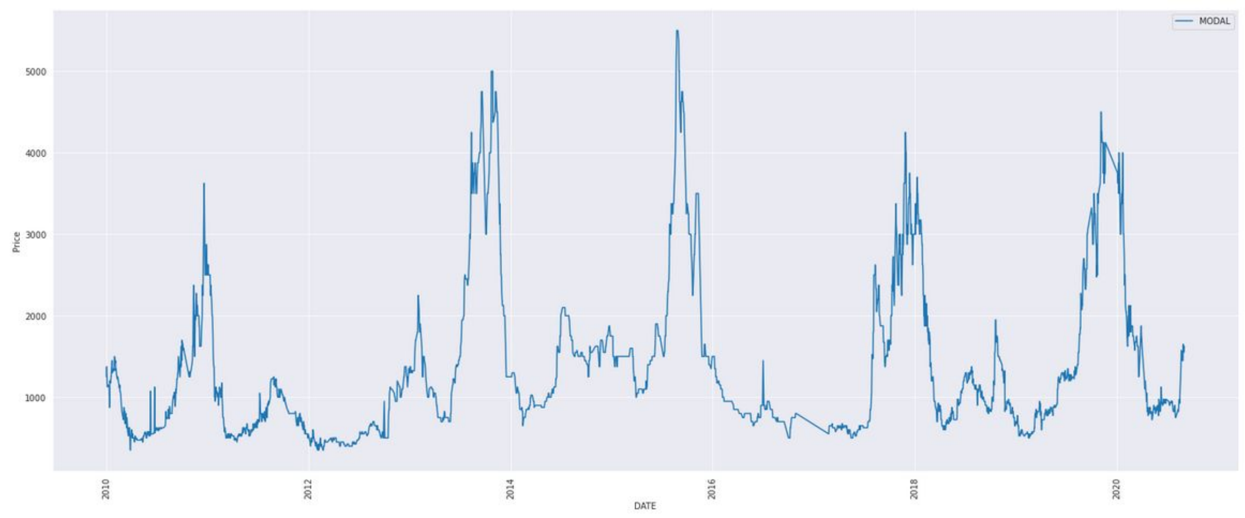
- We’ve constructed **box plots** to view how the values have ranged over the years. Along with the median values, we observe the percentiles that give sound notions about the variations as well.



- Then, we’ve made added visualizations by taking differences in the time period. This enables analyzing the suitability of these differenced series for **ARIMA** and other methods.
- Next, the series is **decomposed** using additive method as the variance of the data is not seeming to be increasing over the years. Due to this, we got a clear picture of the irregular trend present in our data. The seasonality that is produced by the decomposition is the same as expected.
- The data when visualized as histogram appeared to **log normal** and to verify this when log normalized values are plotted that seemed out to be interesting as it produced two peaks in the graph(one major and another minor).
- To check correctness in seasonality of the data of Surat we plotted the data of pune, Both plot are similar.



Pune



Surat

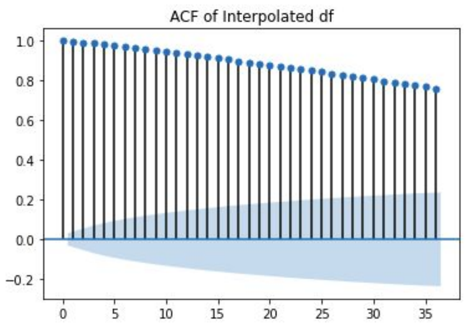
Models

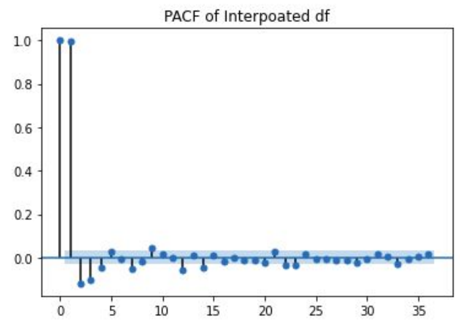
Seasonality observed

On a basic visualization of the obtained dataset, we've inferred that there is a notable seasonality of about two years in the price of onion at Surat. Despite a decent amount of noise and residuals, the overall behavioral pattern of the data depicts that around the end of each of the odd years (2013 - 2019), we notice a seasonally-driven peak in the prices. Thus, basic ARIMA models are not the right choice, so far as 8-10 years of data is concerned. Viewing the same, we've started working on models that provision the capturing of both seasonality and the trend of a stationary series in order to promise a decent prediction of the onion prices for the next period.

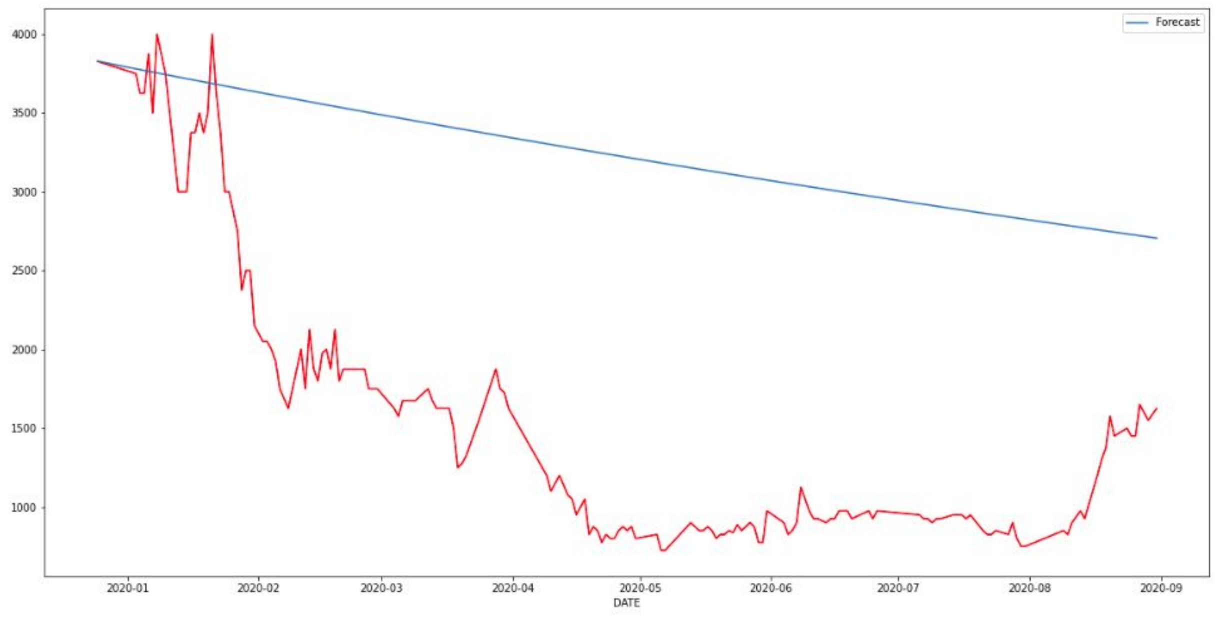
Hence, we have tried on SARIMA and FB Prophet as models, the details of which are discussed in the following sections.

SARIMA observations



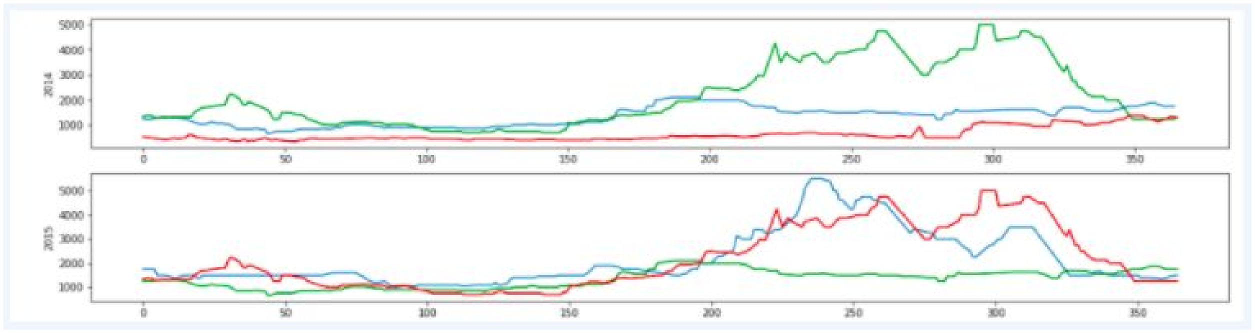


- We plotted **ACF** and **PACF** plots for the onion data that we had collected. From these graphs, we found that there are 3 lag terms (AR) and many MA terms which are significant.
- As there are many MA terms with high significance we choose to take only 2 MA terms to make the model simpler.
- We also observed that 7th and 13th lag terms are becoming significant in PACF, so we decided to keep S=7 and P=2.
- We choose d=0, the p-value returned by the adfuller test on the original series is less than 0.05 i.e the series is stationary.
- So, the final parameters are (p,d,q) = (3,0,2) and (P,D,Q,S) = (2, 0,0,7).
- The summary of this model said that S.L.14 is insignificant, so we tried to make a simpler model by taking (p,d,q) = (2,0,2) and (P,D,Q,S) = (1,0,0,7).
- All terms in the second model are significant and the model was also able to capture some trends in the test data, though the predictions are not accurate enough.
- Root mean squared error = Rs 19.5 per Kg.

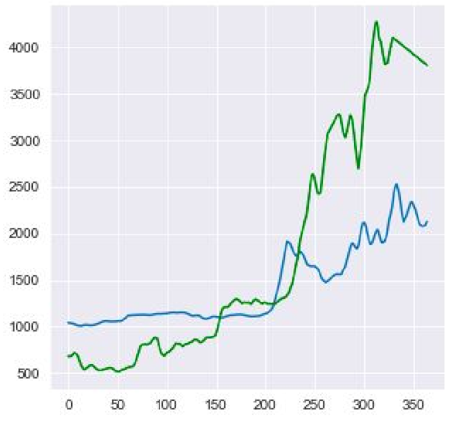


Linear Regression

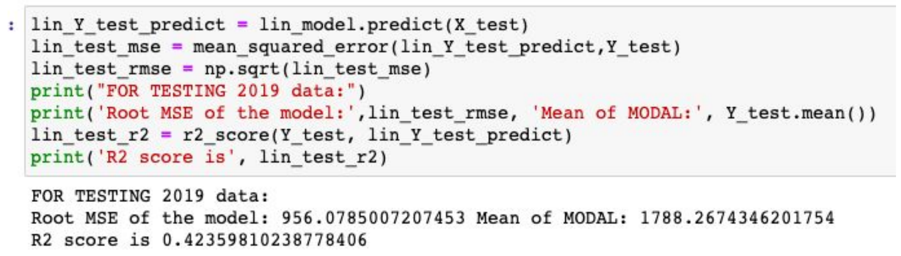
- Following figure depicts the current year prices in Blue, the last year prices in Green and the prices of two years back in Red.



- After the results obtained from SARIMA showed very little accuracy and very less propensity towards the correct trend, we chose to approach the problem using much simpler methods and try to judge how close we're able to reach. This brought in Regression into the picture.
- From the figure shown above, we observe that the blue curve can be generalised as a **middle curve** running between the green and the red curves.
- Novelty in this approach: Use 'Price 1 year back', 'Price 2 years back', etc. as **distinct parameters**. This brings in the involvement of multiple parameters, in contrast to the single parameter we possessed till now.
- Instead of merely going by the average of the prices obtained, we are implementing regression that shall give us the best weights that can make the predictions bear a decent adherence to the true values.
- The following figure shows how Regression worked in predicting onion prices. As observed, the model roughly captured the rise that has happened towards the end of the year.



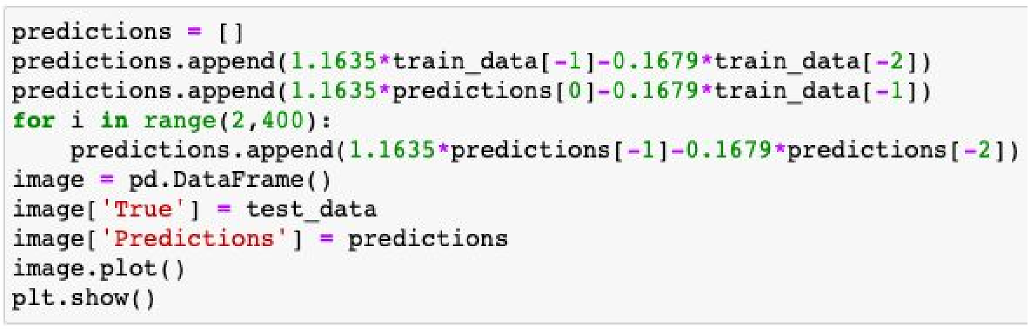
- We've obtained the following result for Linear Regression on onion data. Here RMSE turns out to be Rs. 956 per quintal, i.e. Rs. 9.56 per kg.



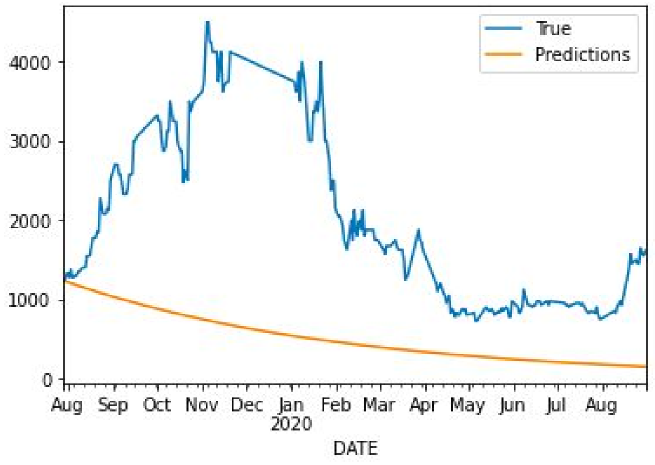
```
: lin_Y_test_predict = lin_model.predict(X_test)
  lin_test_mse = mean_squared_error(lin_Y_test_predict,Y_test)
  lin_test_rmse = np.sqrt(lin_test_mse)
  print("FOR TESTING 2019 data:")
  print('Root MSE of the model:',lin_test_rmse, 'Mean of MODAL:', Y_test.mean())
  lin_test_r2 = r2_score(Y_test, lin_Y_test_predict)
  print('R2 score is', lin_test_r2)
```

```
FOR TESTING 2019 data:
Root MSE of the model: 956.0785007207453 Mean of MODAL: 1788.2674346201754
R2 score is 0.42359810238778406
```

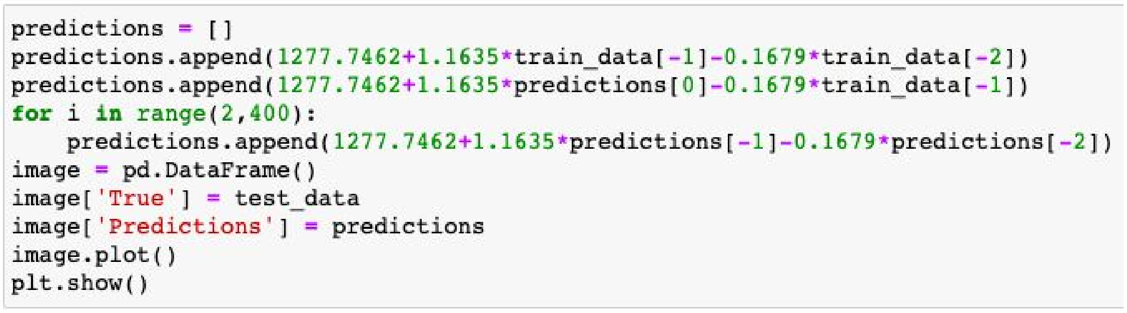
- Thus, **adding parameters like data of 1 week back, 2 weeks back**, etc. as suggested by the visualisations of seasonality made, we are likely to obtain **better results**.
- However, in case we need to predict for a month or greater duration, as we've done in the above figures, we need an approximation for the data 7 or 14 days back.
- **Problem** involved in the task of **approximating** true data of some period with the predictions:
    - This was an effort in **ARIMA**, taking 2 AR terms as per the results obtained in summary. Though the training data had been fit quite well, the following results from testing data showed the problem. Here we had taken predictions[-2] to replace the training data of 2 days back, and this stripped the predictions of the expected trend of rise and fall.



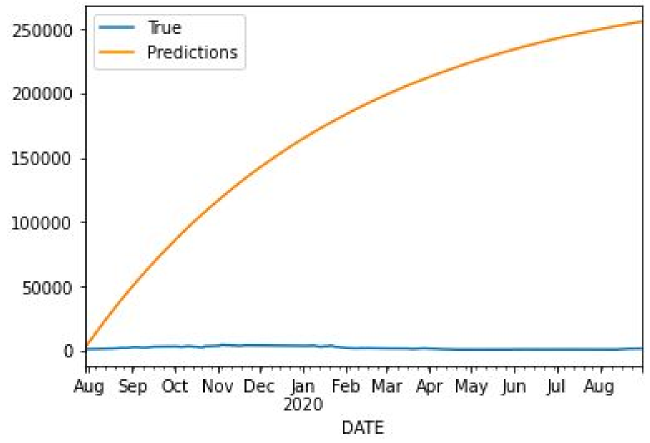
```
predictions = []
predictions.append(1.1635*train_data[-1]-0.1679*train_data[-2])
predictions.append(1.1635*predictions[0]-0.1679*train_data[-1])
for i in range(2,400):
    predictions.append(1.1635*predictions[-1]-0.1679*predictions[-2])
image = pd.DataFrame()
image['True'] = test_data
image['Predictions'] = predictions
image.plot()
plt.show()
```



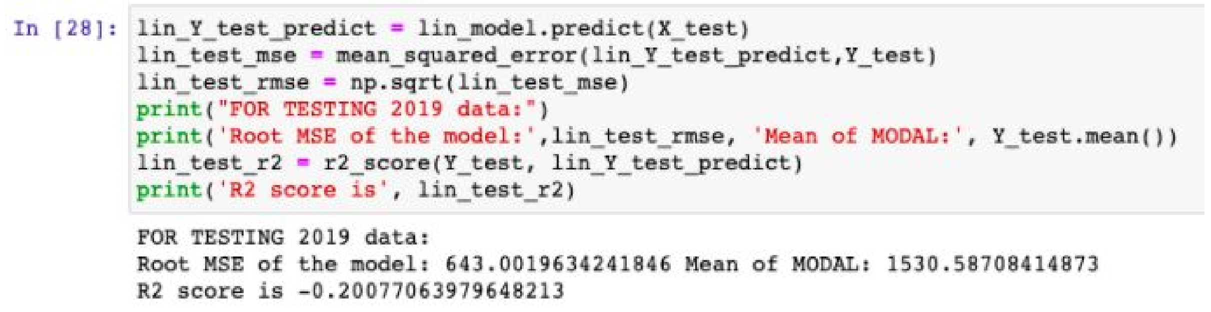
Here, the constant term had been removed. On placing the constant term back, we observe the following - a largely incorrect trend of increase in the predictions:



```
predictions = []
predictions.append(1277.7462+1.1635*train_data[-1]-0.1679*train_data[-2])
predictions.append(1277.7462+1.1635*predictions[0]-0.1679*train_data[-1])
for i in range(2,400):
    predictions.append(1277.7462+1.1635*predictions[-1]-0.1679*predictions[-2])
image = pd.DataFrame()
image['True'] = test_data
image['Predictions'] = predictions
image.plot()
plt.show()
```



- From the above observations, we concluded that **there are potential difficulties if the parameters required to predict some value are themselves predicted values.**
- This led us to conclude that Regression on the basis of values of the previous two years could give justifiable predictions for onion, however it suffered a negative R2 score in case of tomato.



```
In [28]: lin_Y_test_predict = lin_model.predict(X_test)
         lin_test_mse = mean_squared_error(lin_Y_test_predict,Y_test)
         lin_test_rmse = np.sqrt(lin_test_mse)
         print("FOR TESTING 2019 data:")
         print('Root MSE of the model:',lin_test_rmse, 'Mean of MODAL:', Y_test.mean())
         lin_test_r2 = r2_score(Y_test, lin_Y_test_predict)
         print('R2 score is', lin_test_r2)

         FOR TESTING 2019 data:
         Root MSE of the model: 643.0019634241846 Mean of MODAL: 1530.58708414873
         R2 score is -0.20077063979648213
```

- However, if the target period is restricted to 7 days, the accuracy can expectedly be increased decently.
- The next section talks about the FB Prophet Model, the results of which convinced us to cease our procedures on Regression and continue on Prophet alone.

FB Prophet

- FB prophet divides the y value into three components, growth ($g(t)$), seasonality ($s(t)$), and holidays/special-events ($h(t)$); we have ignored the holidays component since adding them didn't improve the model.

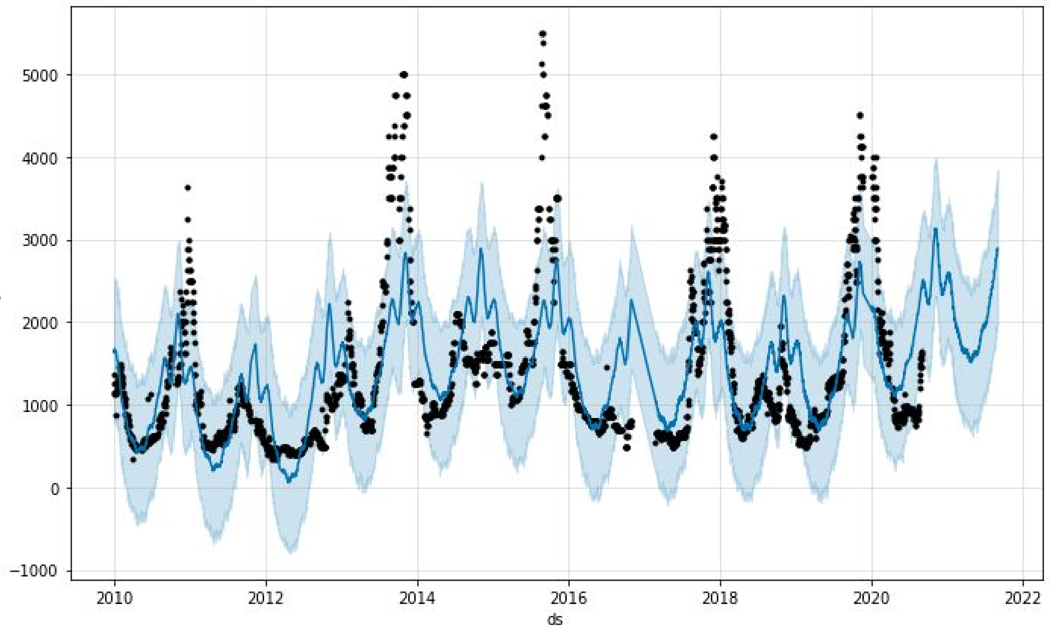
$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

$$g(t) = \frac{C}{1 + exp(-(k + a(t)^T\delta)(t - (m + a(t)^T\gamma)))}$$

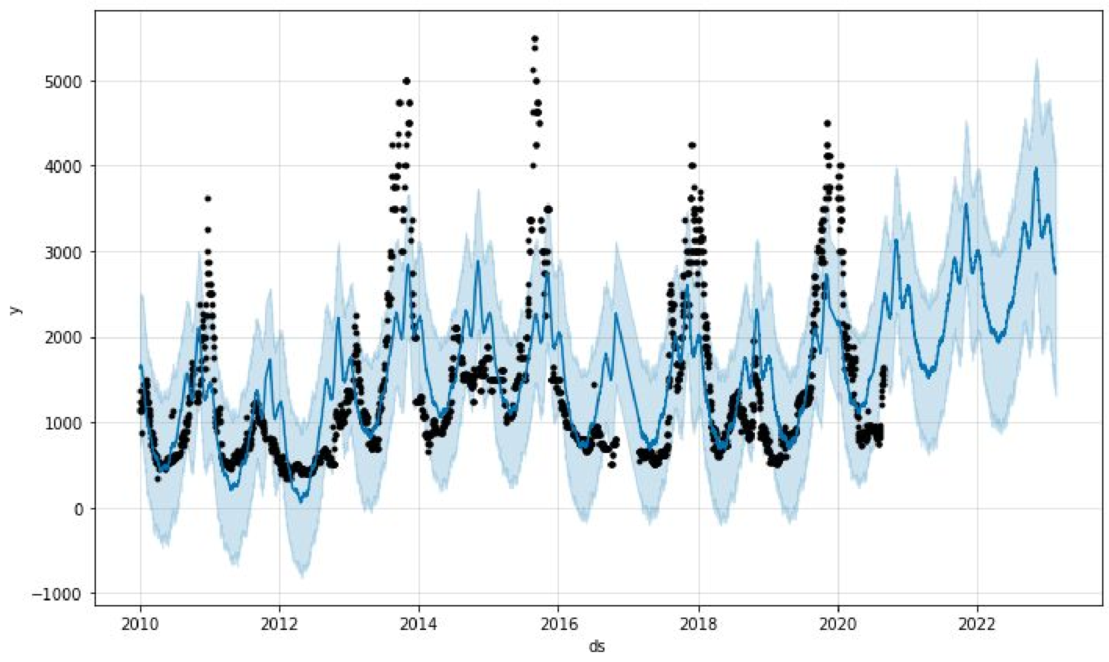
$$\gamma_j = \left(s_j - m - \sum \gamma_l\right)\left(1 - \frac{k + \sum_{l<j}\delta_l}{k + \sum_{l\le j}\delta_l}\right)$$

$$s(t) = \sum_{n=1}^{N}\left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right)\right)$$

- We tried forecasting the price for onion with Prophet for one year and obtained average-good results, the confidence intervals for the prediction were wide.



- Also trying the forecast for longer periods we observed that price gradually increased in the prediction which may be because of the growth function which is not accurate for our data as the data appears to be almost stationary.

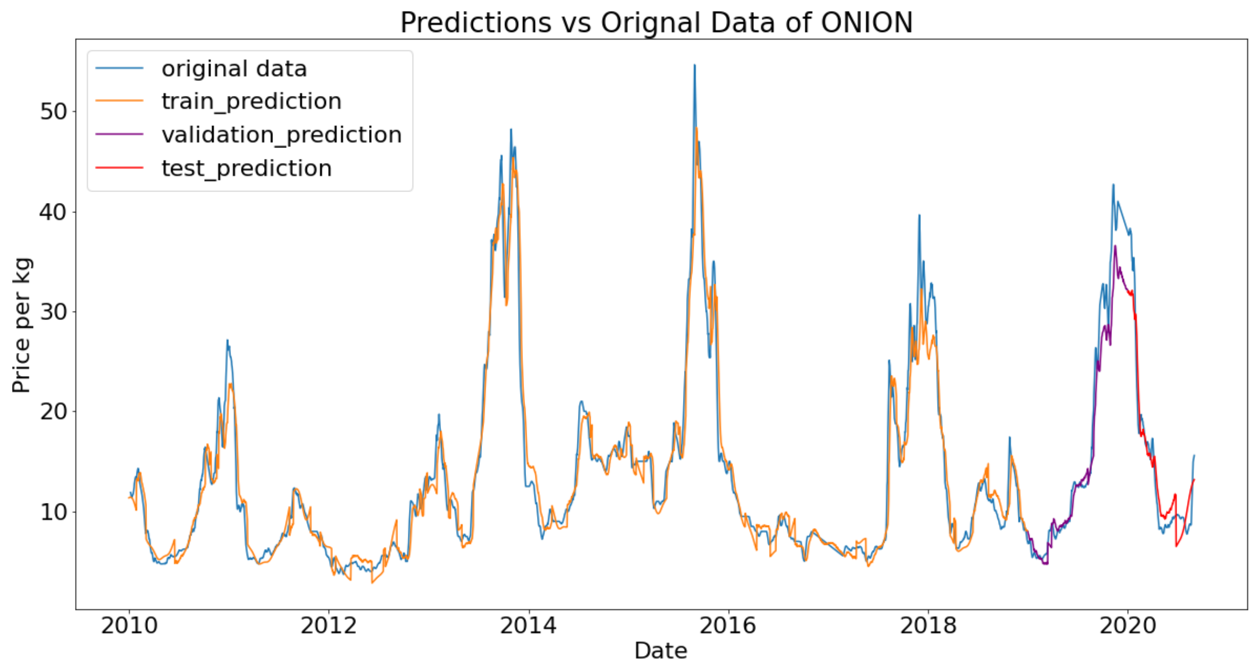


- We also observed that the price of onion has a periodicity of 2 years and Prophet considers 1-year seasonality which might have adversely affected the prediction.
- We've thus planned to test whether the model performs better if we divide the data into two partitions (so that seasonality is 2 years), carry out subsequent averaging among the two partitions, and perform predictions in the next phase of the project.
- Some other observations made from Prophet is that there are unexpected spikes in data that are not predicted by the Prophet (or spikes that the Prophet didn't expect). Looking specifically into those years (end of 2010 and 2013), we've found unexpectedly high rainfall which had probably damaged the crops that might have affected the prices. So we decided to add weather (rainfall) as a parameter for our model in the upcoming steps.
- As said earlier, we tried to train two models, one for even years and other for odd years. But the results were not attractive. One of the core reasons for this is, seasonality is not exactly of 2 years, there are some deviations from the 2years mark due to which there is an incontinuity produced in data when it is segregated by parity of years. So we tried something different then.
- We observed a strong correlation between last week's price to the current day's price, this observation is also visible in PACF plots that we made in the SARIMA section.
- So we applied a custom model layer on top of the Prophet's prediction.
- This core idea of the custom model is to use the prophet's prediction as it is if the last week's prediction is accurate enough else use a linear combination of last week's original value, last week's prediction, yhat_upper/lower and last week's yhat_upper/lower.
- The coefficients involved in the custom model are found by doing truncated exhaustive search by applying meaningful conditions on them.
- As a result, the time complexity of ~10^10 reduced to ~10^8.

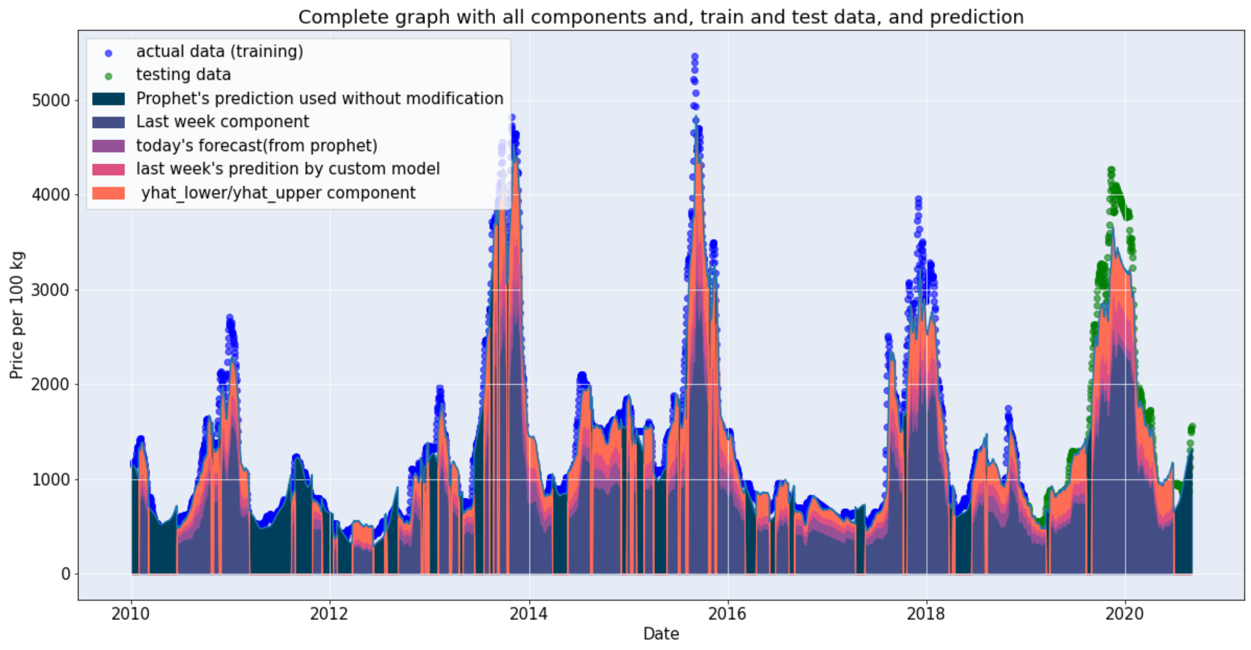
- We did train the model on 2010-2018 data, cross validation was done on 2019 data to find coefficients and testing was done on 2020(till september) data to calculate the final RMSE of the model.
- As expected, the coefficient of last week's data was higher than other coeffs as there is strong correlation.
- The model Produced good results for 8 out of 9 vegetables and also generalized very well on them.
- The Test RMSE of different vegetables are as follows:

VEGETABLE	TEST RMSE (Rs per Kg)	RANGE (min-max)(Rs per Kg)
Onion	2.54	3.7142 - 54.6428
Bottle Gourd	2.3	3.5000 - 32.8061
Potato	2.02	3.3750 - 23.9054
Cabbage	1.2	2.1607 - 23.2142
Cauliflower	2.99	3.8214 - 32.7295
Carrot	3.2	16.5808 - 51.9841
Tomato	5.04	3.0171 - 53.3928
Pointed Gourd	7.61	6.7142 - 95.3571
Peas	10.44	8.1071 - 152.3214

+++



- The below Figure displays the different components involved in the custom model and their proportions of ONION predictions.



Pseudo code for custom model:

```
# Do prediction week by week
if (prediction of last week is accurate):
current week = Use Prophet()
else:
```

```
current week = custom layer() => {

# i : day numbering starting from 0 for entire dataframe
# prophet[i] : Prophet prediction of ith day
# df : original data

prediction[i] = (c1 * prophet[i][yhat] +
            # prophet's predict for current day
             c2 * prediction[i-7] +
            #  similar to AR + MA terms
             c3 * df[i-7])
            # similar to AR term

# add expected deviation term which are functions based on yhat_lower and yhat_upper

if (last week's prediction is close to yhat_upper):
   prediction[i] += c4 * prophet[i][close] +
               c5 * prophet[i-7][close]

else:
   prediction[i] += c4 * prophet[i][close] +
               c5 * prophet[i-7][close]

# yhat_lower and yhat_upper are 85% confidence intervals
```

Second Phase of the Project - the creation of Vegetable Basket

- **Inputs**:
    - Prices of all vegetables on a day
    - Required quantity of each nutrient
    - Quantity of nutrient per kg of each vegetable
- **Output**: Quantity of each vegetable in the basket

- **Objective Function**: Minimum Price of the basket, such that all nutrients are covered as per the requirement

Vegetable Basket Choice Algorithm - The Dynamic Programming Approach

- **Approach** - Dynamic Programming
- **Recurrence Relation**:

$$\text{Price}\,(x1, x2, x3, \ldots, xm) = \min\left((\text{Price}\,(x1 - N)_{11})+P_1, (\text{Price}\,(x1 - N)_{21})+P_2, (\text{Price}\,(x1 - N)_{31})+P_3, \ldots, (\text{Price}\,(x1 - N)_{n1})+P_n\right)$$

- Here, n = number of vegetables, m = number of nutrients, Pi is the price of the ith vegetable.
- **To find:** Price (R1, R2, R3, …, Rm)
- **Complexity**: $O(n*|R_j|^m)$, both time and space

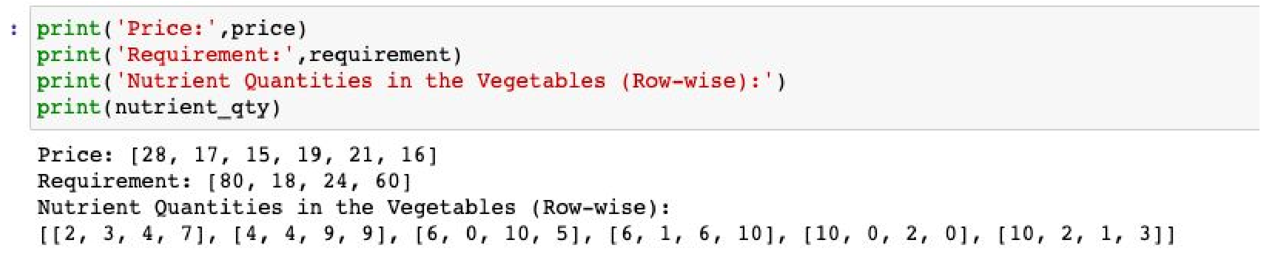
Vegetable Basket Choice Algorithm - The Dynamic Programming Approach with selected vegetables

- **Approach** - Dynamic Programming
- For each nutrient j, we may construct **Sj** = list of vegetables in the decreasing order of **(Qty. of nutrient in a vegetable/Price of that vegetable)**.
- We choose a set H of only those vegetables which is the first member of at least one Sj.
- **Recurrence Relation**:
  We use the same recurrence relation, using only the vegetables selected in the set H of size k.
- **Complexity**:
  - $O(k*|Rj|^m)$, both time and space
  - Useful when k << n.
- May give sub-optimal results if the set misses some vegetable which lies in the optimum basket.

Vegetable Basket Choice Algorithm - Heuristics-based Time-efficient method

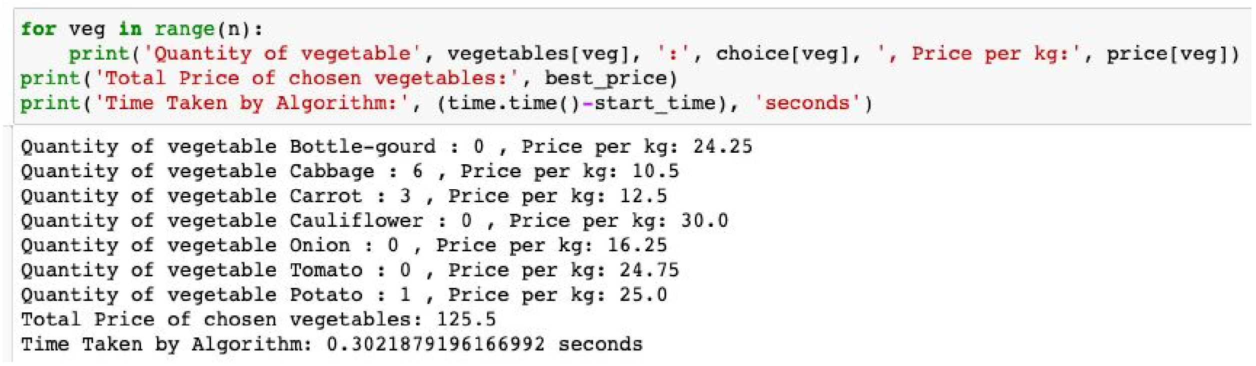
- **Approach** - Dynamic Programming

- For each nutrient j, we may construct **Sj** = list of vegetables in the decreasing order of **(Qty. of nutrient in a vegetable/Price of that vegetable)**.
- We randomly choose a permutation of m nutrients, and do the following steps for all nutrients in that order:
  - For each nutrient j, include the quantity of its first member in Sj which is required to fulfil its remaining requirement
  - Sort the vegetables in the decreasing order of price
  - For each vegetable in the above order, reduce the maximum quantity of that vegetable, preserving the criteria of requirement.
  - Sub-optimal approach, Complexity: O (nm) time and space
- Probability of obtaining a more near-optimal result can be determined by trying out the steps for multiple permutations
- The number of permutations r can be chosen in the range [1, m!]
- **Complexity**:
  - O (nmr) time
  - O (nm) space
- This yields a polynomial time solution to our problem
- Thus, we'll use the DP solution if n and m are small, and resort to this for large values of n and m



```
: print('Price:',price)
  print('Requirement:',requirement)
  print('Nutrient Quantities in the Vegetables (Row-wise):')
  print(nutrient_qty)
```

```
Price: [28, 17, 15, 19, 21, 16]
Requirement: [80, 18, 24, 60]
Nutrient Quantities in the Vegetables (Row-wise):
[[2, 3, 4, 7], [4, 4, 9, 9], [6, 0, 10, 5], [6, 1, 6, 10], [10, 0, 2, 0], [10, 2, 1, 3]]
```



```
for veg in range(n):
    print('Quantity of vegetable', vegetables[veg], ':', choice[veg], ', Price per kg:', price[veg])
print('Total Price of chosen vegetables:', best_price)
print('Time Taken by Algorithm:', (time.time()-start_time), 'seconds')
```

```
Quantity of vegetable Bottle-gourd : 0 , Price per kg: 24.25
Quantity of vegetable Cabbage : 6 , Price per kg: 10.5
Quantity of vegetable Carrot : 3 , Price per kg: 12.5
Quantity of vegetable Cauliflower : 0 , Price per kg: 30.0
Quantity of vegetable Onion : 0 , Price per kg: 16.25
Quantity of vegetable Tomato : 0 , Price per kg: 24.75
Quantity of vegetable Potato : 1 , Price per kg: 25.0
Total Price of chosen vegetables: 125.5
Time Taken by Algorithm: 0.3021879196166992 seconds
```

This is an implementation of the first approach. We know that it always gives the most optimum solution and we observe that in short inputs, it consumes about 0.3 seconds, which suggests that it can be conveniently used for short inputs, and the outputs can be fetched in reasonably short time.

Hence, it brings us to the successful completion of the last steps of our project.

Work Division

Sl. No.	Task	Members Involved
1	Data Collection	All (each member had 2 vegetables)
2	Code to select market by maximizing available data	Manu, Saptarshi
3	Data Visualizations using various plots	All
4	Data Cleaning and Outlier Analysis	Prince, Siram Nikhil
5	Weather Data Analysis	Anmol, Prince
6	ARIMA and SARIMA	Siram Nikhil, Saptarshi
7	Linear Regression	Prince, Saptarshi
8	FB Prophet + Custom Model	Manu, Siram Nikhil
9	Analysis of Models and necessary visualizations for comparison	All
10	Vegetable Basket Creation Algorithm	Anmol, Saptarshi
11	Presentation and Report	All

The above table shows the members who have spent maximum time in the respective tasks. In view of the entire project, the overall task was decently shared by all, and there has been handsome coordination in resolving the umpteen bottlenecks that kept on arising from time to time in all tasks, major or minor.

Challenges Faced

- The normal decomposition methods we've used are portraying a complex time series. Despite identifying the trend and seasonality, there's a significant presence of residual noise that remains to be taken care of.

- Time complexity has been faced as a problem in incorporating higher seasonality values. On careful analysis, we've found that SARIMA model fitting time grows geometrically with linear increase in the seasonality value. Since the expected seasonality is 730 dates (730 data points), the process couldn't terminate, despite working comfortably for smaller values.
- Large and arbitrary irregularities in the prices' trends have made the analysis quite complex, due to which we had to make many significant changes in our procedures.
- The Irregularity in the period of seasonality is the main cause for the normal fb prophet model to fail and make the data discontinuous when it is divided based on the parity of years to analyze separately.

Conclusion

- The project significantly depicts how standard techniques of Data Analysis can help in reaching valuable results that are significant to the common people.
- Apart from the libraries available, some intuitive techniques have been found to be highly beneficial in improving the quality of results.
- The final deliverables are thereby ready for the final deployment, as it can predict the prices for a duration, and thereafter convey the best vegetable basket to the consumers.