**Sacramento’s Booming Businesses**

|  |  |  |
| --- | --- | --- |
| Ken Koyanagi  *California State University, Sacramento*  *kenkoyanagi@csus.edu* | Justin Tran  *California State University, Sacramento*  *justintran@csus.edu* | Madeline Lewis-Whitfield  *California State University, Sacramento*  *madelinelewis-w@csus.edu* |

**Abstract**

*In this project we have taken a dataset from the City of Sacramento to gain insightful information on Sacramento businesses. Our technique involved grouping together similar words, with a large number unique string values, from a column of data using k-means clustering to create categories, and using the clusters to forecast the growth of those categories.*

Keywords: Clustering; K-means, ARIMA, Forecasting, String vectors

**1. Introduction**

Our goal is to give insight and direction to potential business entrepreneurs or investors who are looking in the Sacramento area. We plan to find out how well businesses in general are doing in Sacramento, where businesses are opening most frequently, which business categories are doing well and to predict the future business growth in Sacramento.

The City of Sacramento provides data on business operations in the area. The data we used is the Business Operation Tax Information data that is available through their API. When we first gathered the data, we retrieved 89,724 entries with 27 columns in the data. To achieve our goal of seeing which types of businesses are growing and which are closing in the Sacramento area in addition to how well certain areas within Sacramento are doing, we will cluster together the similar businesses and create business categories through unsupervised learning methods.

**2. Preprocessing**

We first observed the ‘Business Description’ column of the CSV that was retrieved through the City of Sacramento API, since that will be the value we use for clustering. We noticed that there were many misspellings, different representations of the same thing, a lack of consistency and frequent use of special characters. Out of 89,624 rows, there were 42,642 unique business descriptions.

We first put the data through data reduction and reduced the number of columns to nine, only maintaining useful information and getting rid of specific location information such as street names and which direction the business is facing. The nine columns we kept are Account Number, Business Name, Business Description, Application Date, Business Start Date, Business Close Date, Current License Status, Location City and Location Zip Code. We were also able to reduce the data by only keeping rows where the Business City key held the value “Sacramento”. This resulted in a reduction of data to 76,240 rows.

The next step was to clean the data. We reduced the zip codes to five-digit zip codes since only some of the rows had extended nine-digit zip codes. We then removed special characters and numbers by only allowing alphabetical characters in the business description. We then removed stopwords such as ‘and’, ‘the’, ‘of’, ‘for’ and ‘a’ and custom words ‘service’ and ‘services’ since we observed that those words were occurring in different business categories and skewing our clustering. Lastly, we replaced instances of ‘A/C’ with ‘AC’ since the removal of special characters was modifying the word representation. Lastly, we reformatted the dates to an R and MongoDB friendly format.

**3. Clustering**

The next step was to cluster the businesses based on their business description and for this we used the Scikit-learn library. Since we had string values for the description, the first step was to get a vector representation. We used the fit transform function to get the inverse document frequency (IDF) of the array of business descriptions. The IDF diminishes the weight of terms that occur very frequently in the array, increases the weight of terms that occur rarely and returns a TfidVectorizer (tf-idf). The TfidVectorizer function will then get input the tf-idf and convert the collection of raw document to a matrix of the features.

*lines = numpy.array(d0c)*

*doc\_feat = TfidVectorizer().fit\_transform(lines)*

After we created a matrix from the business descriptions, we attempted to find the optimal k for our k-means cluster. We attempted the silhouette width method on a 16GB machine and with a matrix consisting of over 76,000 entries. The machine reached a state where there was 48GB of compressed memory and soon crashed. Therefore, we had to look for a method that wasn’t as memory intensive.

We turned to the elbow method in R with over 76,000 entries to test k values from 1 to 100. Unfortunately, with the amount of entries, the calculations took far too long. We then took a random sample size of 7600 entries, which is less than 10% of the data, to compute k values from 1 to 100. This still took over four hours to finish the computation, but finished producing the graph using ggplot2 in R.

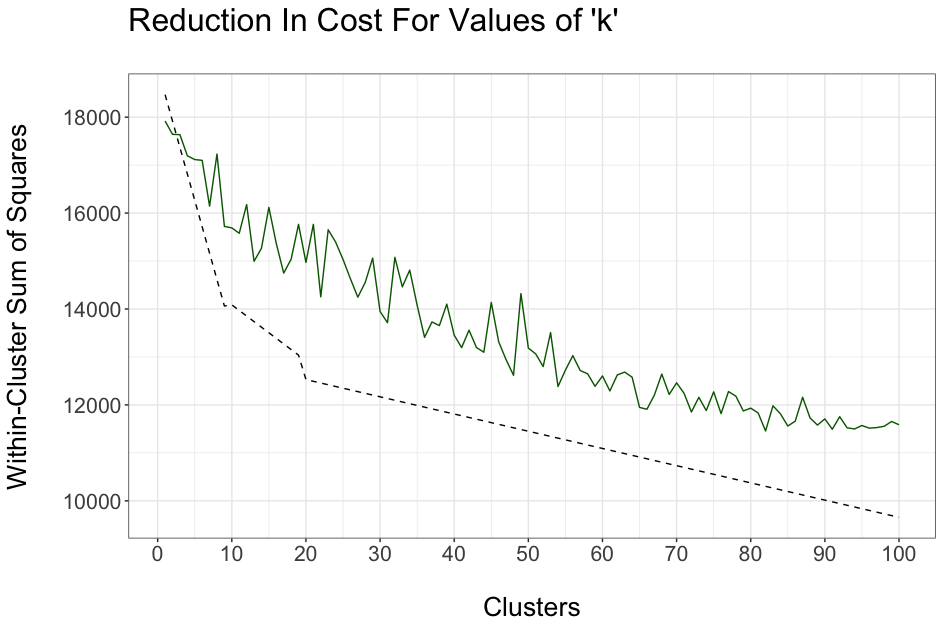


Figure 1. Elbow Curve

The curve indicates that the optimal k is somewhere near 20, however after doing a k-means cluster with k=20, we found that the cluster quality was poor. The sample taken to generate the curve wasn’t an accurate representation since there are 42,642 unique business descriptions and with a sample size of 4,600 it isn’t representing the full variance in the data.

Through manual testing and analysis and by viewing the clusters, we found that a k value of 50 was near the optimal k. Now with a k defined, we wanted our cluster script to check to see which clusters were to be deemed ‘good.’ We first had the script look into each cluster and find the top three most frequently occurring words using a collection. We checked to see if the topmost common word in the cluster occurred in its respective cluster greater than 90% relative to the size of the cluster, thus checking the intra-class correlation. Then we check to see if the cluster size was greater than twenty. If it passed these two checks, it was deemed to be a ‘good’ cluster. To further check the quality of the cluster, we could have checked the top three most common words and checked to see if they occured a high-percentage of times in other clusters to check the inter-class correlation.

Once the clusters were created and the good clusters were labeled, we wanted to label our clusters, thus creating the business category labels. We had the script create the label based on the most common words in the cluster. The script looks at the top three most commonly occurring words and if the word occurred more than 50% of the time in the cluster, it became a part of the label. For example, if the top three words were ‘ENTERTAINER, ‘ADULT’ and ‘DANCER’ with their frequency being 95%, 93% and 48% respectively, the cluster was labeled as ‘ENTERTAINER/ADULT.’

We generated the following labels, with 42 good clusters and 7 bad clusters.

Table 1. Generated Cluster Labels



**4. Data Warehouse**

We stored the data in MongoDB, which is unconventional, but we chose to use it for its flexibility and as an opportunity to learn a new technology. The documents that make up our database all follow the same structure and contain the following attributes.

Figure 2. Sample Document

|  |
| --- |
| {  "\_id": ObjectId("5911fd1d6956ef632b252c4e"),  "Account Number": 1024050,  "Business Name": "JUICY",  "Business Description": "ADULT ENTERTAINER",  "Application Date": "2016-02-05",  "Business Start Date": "2016-02-05",  "Business Close Date": "",  "Current License Status": "LICENSE EXPIRED",  "Location City": "SACRAMENTO",  "Location Zip code": 95811,  "Cluster": 40,  "Good Cluster": "GOOD",  "Representation": 0.946987951807,  "Cluster Description": "ENTERTAINER (0.946987951807)  ADULT (0.934939759036)",  "Cluster Label": "ENTERTAINER/ADULT"  } |

Using the clustering script, we added the attributes: ‘Cluster’ which represents the number of the cluster it was in; ‘Good Cluster’ which is either ‘GOOD’ or ‘BAD’; ‘Representation’ which is the representation of the most common word in its respective cluster, with 1.00 being 100% representation; ‘Cluster Description’ which contains the most common words in the cluster with a limit of three words with their representation in parentheses; and lastly the ‘Cluster Label’ is added with the name of the label.

If a Business Close Date does not exist, that indicates that the business is still open or in a few cases, that the businesses are still in the application state.

The database can be queried using either the find or aggregate function. For example, if we want to find all of the businesses in the FOOD cluster, we run ‘db.clustered.find({“Cluster Label”:”FOOD”})’. To make this simpler, we have created an interface on our website that consists of dropdowns that allow the user to easily query the data.

**4.1 Questions Answered**

The user of our application is able to ask several questions and get the answers returned in the form of a table with the query results. We provide the user a dropdown list to specify their search parameters. The search parameters are industry, cluster quality, license status and business status. An example of a question a user might have is how many businesses in the beauty industry are open and have a license status of renewed. The user would specify these search parameters using the dropdowns and the following query to the mongo database would be generated by the system:

collection.find({$and:[{$or:[

{"Cluster Label" : "HAIR/STYLIST"},

{"Cluster Label" : "BEAUTY/SALON"},

{"Cluster Label" : "SALON/HAIR"}]},

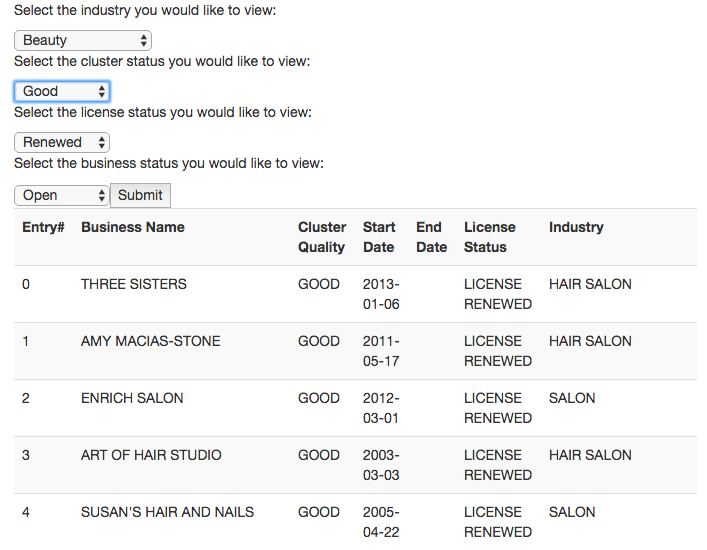
{"Good Cluster": quality},

{"Current License Status" : license},

{“Business Close Date":status}]

},{},function(e,docs)

This would return all the documents in the mongo database that have a cluster related to beauty, and the license and business status and cluster quality the user specified.



**5. Forecasting Growth**

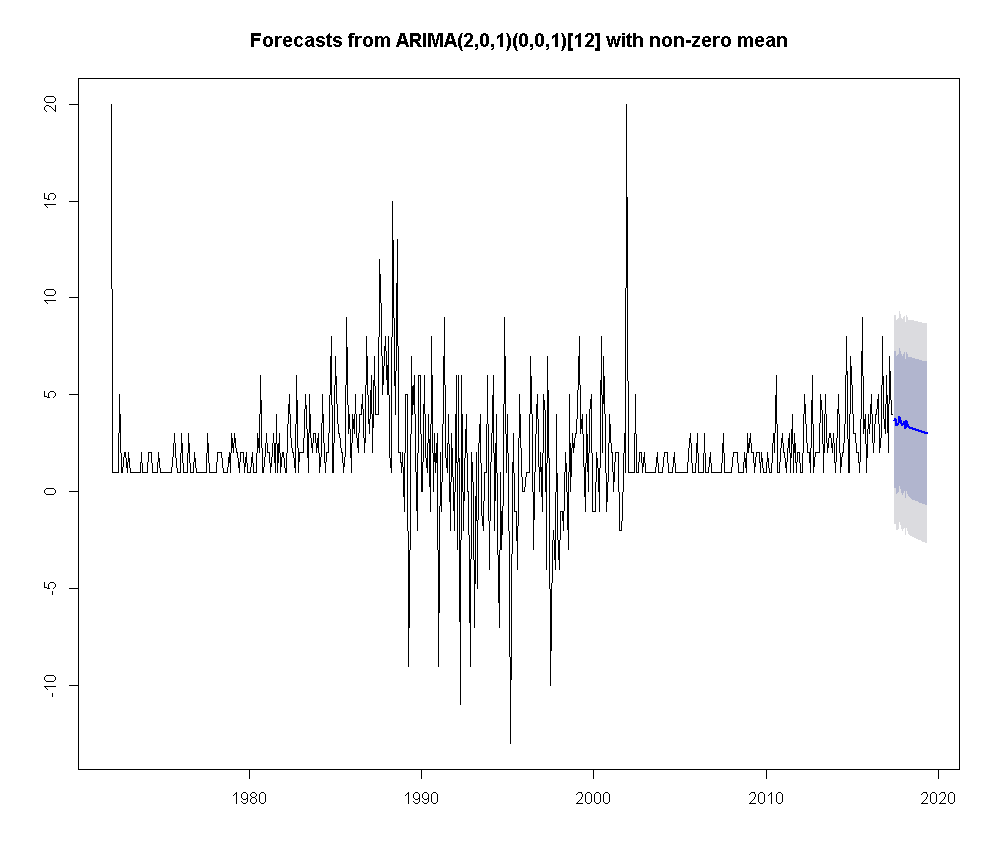
Data forecasting is the process of making predictions based on past and present data. We conducted our data forecasting through an Autoregressive Integrated Moving Average Model (ARIMA Model).

ARIMA is the data forecasting technique that analyzes a time series – a set of data that is indexed by time to generate a prediction. It is commonly used for short term predictions and requires a minimum of 40 historical data points. The ARIMA method allows for the construction of a model that implements both the Autoregressive (AR) and Moving Average (MA) model. An AR model will create a forecast that is dependent on a linear combination of past values, while the MA model will create a forecast that is dependent on past error points, which are outliers.

For us to utilize the ARIMA model, we needed to create a time series for each good cluster label to forecast their future growth. Using a Python script, we were able to produce a time series for each category that contained the total amount of businesses minus the amount of businesses that closed for each month.

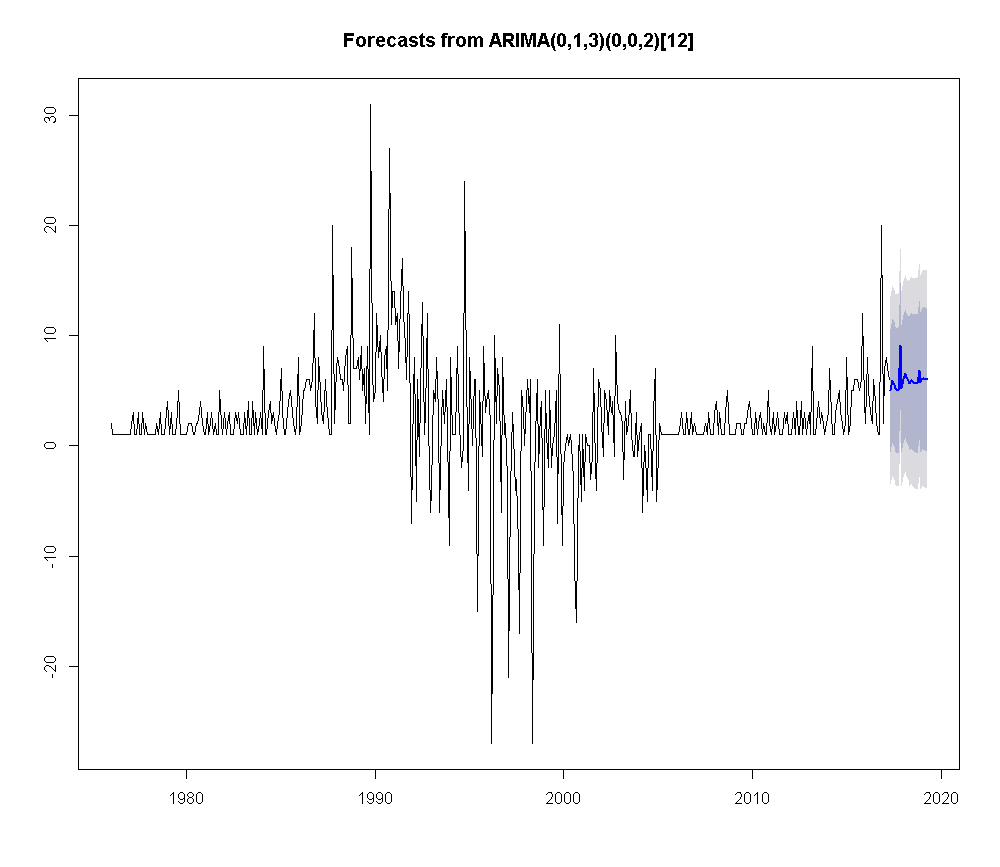
Using R, we used the Forecast library to produce and illustrate the data prediction of each cluster label, which produced forecasts for the next two years. Figure 3 illustrates the amount of Restaurants businesses growth. The y-axis represents the number of restaurants and the x-axis is the time in years. Data points that are below zero indicate that there were more businesses that closed than opened during that time.

Figure 3. Restaurant Growth Forecast



We observe that the Restaurant industry is predicted to decline back to zero in the next few years based on the graph. In contrast, Figure 4 shows that Consulting businesses are predicted to grow in the Sacramento area.

Figure 4. Consulting Growth Forecast



Lastly, we wanted to take a look at all businesses in Sacramento.

Figure 5. All Businesses in Sacramento

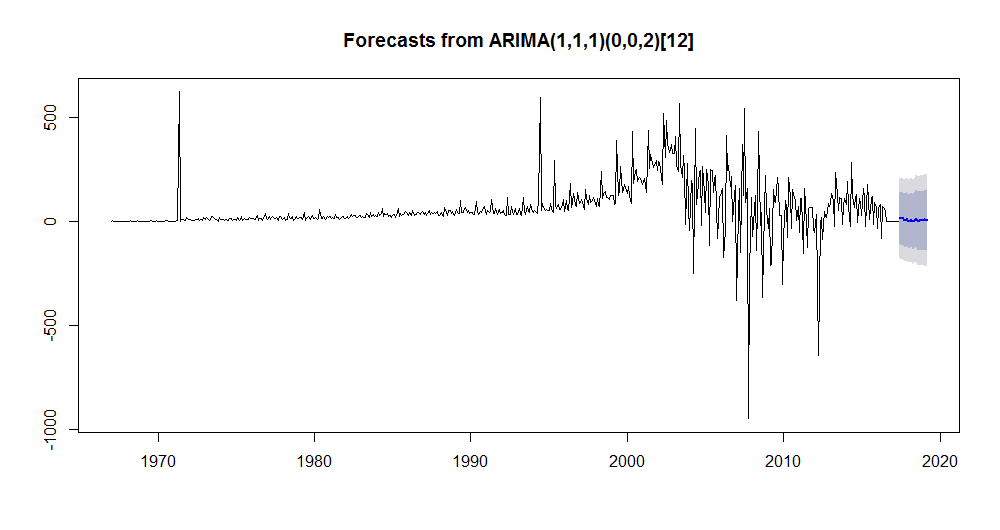


Figure 5 indicates that businesses in general will stabilize in Sacramento somewhere near zero after experiencing heavy spikes from the early 1990’s to the early 2000’s.

**6. Heatmap**

We also wanted to see where new businesses were opening up in the Sacramento area in 2017. We took the businesses that started in 2017 and used geopy, a Python library that takes the businesses’ five-digit zip code and gives us the longitude and latitude, and created a heat map. Since an area code can be quite large and grouping the businesses into a single point causes misrepresentations in the map, we increased the radius of the zones to help account for the inaccuracy.

We generated the list of points using Python and used the Google Maps API to plot the points and create the heatmap. In Figure 6, we can see that most of the new businesses opened in the Midtown area, which is not surprising if you are familiar with the Sacramento area.

Figure 6. Heatmap of New Businesses in 2017



The next step would be to use our ARIMA model to forecast which areas in Sacramento are predicted to have an overall increase in businesses, and to observe the predictions for certain business categories.

**7. Results**

We were able to cluster our data and generate business description categories with no prior knowledge to the types of major businesses in the Sacramento area. We then were able to use those generated categories to forecast how the business types are predicted to perform in the next two years. The graphs indicate that businesses in general will start to settle near zero in the next few years after experiencing large spikes in the past few decades. We also found where most businesses opened in 2017 and discovered that Midtown is, as of now, the most popular area for businesses to open.

We were able to take dirty data with no consistency and a large number of unique business descriptions and group together the similar businesses using k-means clustering. Looking at the clusters that were generated with our eyes and looking at the representation numbers of the clusters, with 42 good clusters being generated, we’re decently pleased with the quality of the clustering.

Since our data was too large to process in a reasonable amount of time, we had to find a good k value through trial and error, but the number we found seemed to work out. We also faced a few challenges and had to implement a few workarounds for mapping the correct data row to the cluster, but in the end, the clustering worked on string values.

The ARIMA model was a good representation of the categories growth prediction that was based off our cluster and time series data created with our Python script that calculated the growth or decline of businesses in each month in a year.

**8. Conclusion**

The preprocessing of the data was a crucial part of the project and we believe that the state should standardize the way employees enter business descriptions after seeing the number of misspellings and unnecessary use of special characters. All of us contributed to create the rules to clean and reduce our data. We then worked on clustering the descriptions which involved a lot of testing, we even experimented with Affinity Propagation clustering and R. We eventually found success with using scikit-learn for k-means clustering in Python.

We then worked on creating the ARIMA models using the Forecast library in R which involved manipulation of the clustered data and outputting a dataset in the form the library expected. We then began working on a website that hosts our findings including our heatmap and connects to our database allowing users to make discoveries from our cleaned and clustered data.

**9. Source Code**

Our source code is available on Github at https://github.com/KKoyanagi/SacBusinesses.

**10. References**

[1] "8 ARIMA models", *Otexts.org*, 2017. [Online]. Available: https://www.otexts.org/fpp/8. [Accessed: 16- May- 2017].

[2] "An Introduction to Intraclass Correlation that Resolves Some Common Confusions". [Online]. Available: http://www.faculty.umb.edu/peter\_taylor/09b.pdf. [Accessed: 16- May- 2017].

[3] "Clustering text documents using k-means — scikit-learn 0.18.1 documentation", *Scikit-learn.org*, 2017. [Online]. Available: http://scikit-learn.org/stable/auto\_examples/text/document\_clustering.html. [Accessed: 16- May- 2017].

[4] "K Means Clustering in Python", *Stamford Research*, 2015. [Online]. Available: http://stamfordresearch.com/k-means-clustering-in-python/. [Accessed: 16- May- 2017].

[5] "K Means Clustering with Tf-idf Weights", *Jonathanzong.com*, 2013. [Online]. Available: http://jonathanzong.com/blog/2013/02/02/k-means-clustering-with-tfidf-weights. [Accessed: 16- May- 2017].

[6] "A Comparison of Document Clustering Techniques", *Semantics Scholar*. [Online]. Available: https://pdfs.semanticscholar.org/c110/0f525044b2b926f7bd7f407ce7b0157bcfd8.pdf. [Accessed: 16- May- 2017].

[7] "Discovering Word Senses from Text", *ACM*. [Online]. Available: http://delivery.acm.org/10.1145/780000/775138/p613-pantel.pdf?ip=130.86.96.180&id=775138&acc=ACTIVE%20SERVICE&key=F26C2ADAC1542D74%2EF8B29279A7E50DFA%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=763608840&CFTOKEN=96742647&\_\_acm\_\_=1494970815\_25b389e33cc10537daa7744b96bf0f36. [Accessed: 16- May- 2017].

[8] L. Jing, M. Ng, J. Xu and J. Huang, "Subspace Clustering of Text Documents with Feature Weighting K-Means Algorithm", *SpringerLink*. [Online]. Available: https://link.springer.com/chapter/10.1007%2F11430919\_94?LI=true. [Accessed: 16- May- 2017].

[9] J. Contreras, R. Espinola, F. Nogales and A. Conejo, "ARIMA models to predict next-day electricity prices - IEEE Xplore Document", *Ieeexplore.ieee.org*, 2003. [Online]. Available: http://ieeexplore.ieee.org/document/1216141/#full-text-section. [Accessed: 16- May- 2017].

[10] J. Morrison, "AUTOREGRESSIVE INTEGRATED MOVING AVERAGE MODELS (ARIMA)", *Forecastingsolutions.com*. [Online]. Available: http://Forecastingsolutions.com/arima.html. [Accessed: 16- May- 2017].