

**Topic Analysis on Stock Market Prediction**

*Satoshi Taniguchi*

*Junjun Tao*

*Kavuri Lakshmi Kausalya*

**School of Graduate Professional Studies**

Data Analaytics

DAAN 888 – Design and Implementation of Analytics System

Fall, 2022

# Document Control

**Work carried out by:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email Address** | **Task description** |
| Junjun Tao | jbt5620@psu.edu | Coordinating the overall project, devising and building the topic model, and evaluating and discussing the results of the topic model. |
| Satoshi Taniguchi | szt5607@psu.edu | Coordinating the overall project, devising and building the stock price prediction models, and evaluating and discussing the results of the stock price forecasting model. |
| Kavuri Lakshmi Kausalya | lkk5341@psu.edu |  |

**Revision Sheet**

|  |  |  |
| --- | --- | --- |
| **Release No.** | **Date** | **Revision Description** |
| 1.1 | 2022-09-03 | Updated purpose and objectives of the project. Updated task lists in the future in this project. |
| 2.0 | 2022-09-18 | We updated data source, data collection, and tasks for data cleaning. |
| 3.0 | 2022-10-02 | We updated data description and a plan for data preparation. |
| 4.0 | 2022-10-16 | We updated data cleaning results and a plan for data transformation. |
| 5.0 | 2022-10-30 | We updated data transformation results and a plan for modeling. |
| 6.0 | 2022-11-13 | We updated modeling and evaluation. |
| 6.1 | 2022-11-24 | We removed the python script on page 13, added a table of each cluster on page 14, typed equations manually on pages 16 and 17, clarified the explanation about the rolling forecasting technique on page 19, providing more detailed explanation about the Null model by adding equations on page 18, and added discussion about why our models have high prediction accuracy compared to the Null model on page 21 and 22. |

**TABLE OF CONTENTS**

[Document Control 1](#_Toc120704051)

[Week 2 Predictive / Descriptive Analytics System Group-Based Assignment 1](#_Toc120704052)

[WEEK 4 Predictive/Descriptive Analytics System Group-Based Assignment 3](#_Toc120704053)

[WEEK 6 Predictive Analytics System Group-Based Assignment 5](#_Toc120704054)

[WEEK 8 Predictive/descriptive Analytics System Group-Based Assignment 8](#_Toc120704055)

[WEEK 10 Predictive/descriptive Analytics System Group-Based Assignment 10](#_Toc120704056)

[WEEK 12 Predictive / Descriptive Analytics System Group-Based Assignment 12](#_Toc120704057)

[WEEK 13 Predictive/descriptive Analytics System Group-Based Assignment 21](#_Toc120704058)

[WEEK 14 Predictive / Descriptive Analytics System Group-Based Assignment 21](#_Toc120704059)

**General Guidelines**

1. To complete all the homework assignments for this course please use this template document.
2. Each assignment has to be submitting by Sunday 11:59 PM EST.
3. Each figure should be followed by a brief description about the figure.
4. The figures can be hand drawn and scanned in some circumstances, but the hand drawn figure should be clear and legible to obtain full credits. Unclear hand drawn figures will receive partial credits. For constructing figures and diagrams it is advised to use tools.
5. Figures and tables should have appropriate captions. For documenting and referencing styles please follow the APA or MLA writing style.
6. Please make sure that you provide a reference section.
7. Any material text or figure taken from books, journals or Internet should be referenced. If you have a sentence or a figure that does not belong (authorship) to you, they need to be clearly referenced. If you fail to do so your report will be considered as a case for plagiarism. It is your duty to make sure that your report is free from any activity related to plagiarism. In case you are suspected of attempting plagiarism then you will be responsible for the cause. The penalty for plagiarism will be a “0” awarded to your report. So, it is good to keep simple, always have the principle to acknowledge people for their contributions.

Please go through the following instructions before submitting the report

**Academic Integrity**

Academic integrity — scholarship free of fraud and deception — is an important educational objective of Penn State. Academic dishonesty can lead to a failing grade or referral to the [Office of Student Conduct](http://www.sa.psu.edu/ja/).

Academic dishonesty includes, but is not limited to:

* cheating
* plagiarism
* fabrication of information or citations
* facilitating acts of academic dishonesty by others
* unauthorized prior possession of examinations
* submitting the work of another person or work previously used without informing the instructor and securing written approval
* tampering with the academic work of other students

How Academic Integrity Violations Are Handled

In cases where academic integrity is questioned, [procedure requires an instructor to notify a student](http://www.psu.edu/oue/aappm/G-9-academic-integrity.html) of suspected dishonesty before filing a charge and recommended sanction with the college. Procedures allow a student to accept or contest a charge. If a student chooses to contest a charge, the case will then be managed by the respective college or campus Academic Integrity Committee. If a disciplinary sanction also is recommended, the case will be referred to the [Office of Student Conduct](http://www.sa.psu.edu/ja/title=).

All Penn State colleges abide by this Penn State policy, but review procedures may vary by college when academic dishonesty is suspected. Information about Penn State's academic integrity policy and college review procedures is included in the information that students receive upon enrolling in a course.

Additionally, Penn State students are expected to act with civility and personal integrity; respect other students' dignity, rights, and property; and help create and maintain an environment in which all can succeed through the fruits of their own efforts. An environment of academic integrity is requisite to respect for oneself and others, and a civil community.

For More Information on Academic Integrity at Penn State

Please see the [Academic Integrity Chart](http://www.campuses.psu.edu/CAO.pdf)  for specific college contact information or visit one of the following URLs:

* Penn State Senate [Policy on Academic Integrity](http://www.psu.edu/dept/oue/aappm/G-9.html)
* [iStudy for Success!](http://istudy.psu.edu/tutorials/) — learn about plagiarism, copyright, and academic integrity through an educational module
* [Turnitin](http://tlt.its.psu.edu/turnitin) a web-based plagiarism detection and prevention system

# Week 2 Predictive / Descriptive Analytics System Group-Based Assignment

1. **Purpose of our research**

Predicting future stock prices using predictive analysis can be beneficial for obtaining returns on investments. Although various models have been developed for this purpose, most of them are based on time-series analysis of past stock price patterns and economic indicators, and do not focus on the world's interests and emotions. In this project, we attempt to analyze potentially existing topic about politics and economics in the daily news and predict future stock prices based on the results of the topic analysis. The business goal of our project is to predict the future Dow Jones Industrial Average (DJIA), which is a stock market index of 30 prominent companies listed on stock exchanges in the United States, from the results of a time series forecasting model and a topic analysis of the daily news. In our approach will use two models: the ARIMA model, which is a pure time series model, and the ARIMAX model, which combines the time series model with the prediction of the objective variable by the explanatory variables. We intend to improve the forecasting accuracy of the DJIA by incorporating the results of the daily news topic analysis as explanatory variables in the ARIMAX model. For our purposes, we have three business questions to answer;

1. Which topic in the daily news affects the DJIA value?
2. How will the value of future DJIA index change?
3. Do the results of the daily news topic analysis improve the predictive accuracy of the DJIA index?
4. **Tasks for our project**

* Business and data understanding
* Data collection
* Data cleaning and preparation
* Data transformation and variable selection
* EDA
* Modeling
  + Topic analysis

Since we collect news headlines information through the data collection process, the task we face is to solve a natural language processing (NLP) problem because we want to analysis the emtional types based on the data.

For the preprocessing of news headlines data, we choose to use GloVe (Global Vectors for Word Representation) after data cleaning process, which include removing special characteristics, transforming to lower case, remving stop words.

We will use RNN (Recurrent Neural Networks) model to do topic analysis. RNN model uses sequential data or time series data and is commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning [1]. Therefore, we choose to use RNN model to build classifier for emotions.

* + Stock prices predictions

We will build two models to do the stock prices prediction tasks: the ARIMA model which is a pure time series model, and the ARIMAX model, which combines the time series model with the prediction of the objective variable by the explanatory variables. The comparison results between these two models could tell us wheather topic analysis could improve the accuracy of stock predictions or not.

* Evaluation
* Conclusion

|  |  |  |
| --- | --- | --- |
| Serial Number | Task Description |  |
| 1 | Project Proposal |  |
| 2 | Data Collection |  |
| 3 | Data Cleaning |  |
| 4 | EDA |  |
| 5 | Data Transformation |  |
| 6 | Data Modelling |  |
| 7 | Predictive Analytics System Evaluation & Validation |  |
| 8 | Final Submission |  |

# WEEK 4 Predictive/Descriptive Analytics System Group-Based Assignment

1. **Data Sources and Data Collection**

Our objective is to predict the value of the DJIA based on the results of the topic analysis of the Daily News. Therefore, we need two datasets: the daily news and the DJIA values. We used a data collection period of 9/1/2017 to 8/31/2022, with the aim of ensuring a data set long enough for time-series analysis.

We decided to use Reddit Economy News as the data source for the daily news. We chose to use Reddit News as our daily news data source because Reddit attracts a great deal of attention from people in the U.S., and the news given by Reddit is consistent with the interests of the general population in the U.S. In addition, an API is available to collect individual news from Reddit News, allowing us to easily extract the top news for each day. We used the "psaw" library in python to collect data with the API. We sorted the news in order of the number of comments posted on the news, and retrieved the top 50 news items. It should be noted that Reddit news only contains headlines, not the full text of the news. The data includes news posting times as time stamps and news headlines. Fig. 1 shows the first 10 rows of the obtained data.



Fig. 1 Examples of the Reddit Economy News dataset

The DJIA values were collected from The Wall Street Journal as csv files of historical data. Fig. This data includes the date, the open value, close value, highest value, and lowest value of the day. Fig. 2 shows the first 10 rows of the obtained data.



Fig. 1 Examples of the DJIA value dataset

1. **Tasks for data cleaning**

**For News Dataset**

Since the news headlines dataset will be used for topic analysis, we need to apply basic steps about data cleaning and several text preprocessing steps in NLP on it.​

* Removing missing values​
* Removing punctuations like . , ! $( ) \* % @ and unidentified special characters​
* Removing Stop Words​
* Lower casing​
* Tokenization​
* Lemmatization​
* Stemming​

**For DJIA Dataset:**

* Missing values: There are 4 types of missing values:
  + 1. Missing Completely at Random
  + 2. Missing at Random
  + 3. Not Missing at Random
  + 4. Structurally Missing Data
* Handling Missing Data:
  + 1. Removing the Data points that are missing
  + 2. Inputting Missing Values ( Mean, Median, Mode)
  + 3. Specialized Algorithms (KNN)
* Outliers:
  + 1. Visual methods ( Box plot, Scatter plot)
  + 2. Statistic methods (Standard Deviation, Box plot)
  + 3. Specialized Algorithms (Isolation Forest)

But since the data we are dealing with are stock price values and they are moving every day, so we cannot define outliers and remove it. **Otherwise we may lose important information about stock prices trending analysis.**

# WEEK 6 Predictive Analytics System Group-Based Assignment

1. **Data Description**

**For News Dataset**

* Brief Overview:



* **Column Descriptions:**

|  |  |  |
| --- | --- | --- |
| ***Column Name*** | ***Description*** | ***Original Format*** |
| id | The unique id number for each Reddit News | String |
| num\_comments | The number of comments under each Reddit News | Numeric |
| score | The score for each new evaluated by Reddit News | Numeric |
| title | The title of each Reddit News | String |
| created | The timestamp for each news creation | Float |
| date | The post date and time for each Reddit News | DateTime |

The dataset will be used for topic analysis because we want to figure out if people’s discussion will influence stock price’s development. For example, if people discuss a lot on the internet about “education” topic during the whole week, then the educational stocks’ prices may increase in the following day. If people seldom discuss about “agriculture” related topics then the stock prices regard to agriculture may decrease during that period.

**For DJIA Dataset:**

Brief Overview:



* **Column Descriptions:**

Date: The date on which the stock price is being traded.​

Open: The price of the stocks at the opening time of the stock market.​

Close: The price of the stocks at the closing time of the stock market.​

High: The highest price for the stocks traded on the day.​

Low: The lowest price for the stocks traded on the day.​

**Data Preparation Plan**

**For News Dataset**

Since the news headlines dataset will be used for topic analysis, we need to apply basic steps about data cleaning and several text preprocessing steps in NLP on it.​

Data Cleaning

* Checking and removing missing values​
* Removing punctuations like . , ! $( ) \* % @ and unidentified special characters​
* Removing Stop Words​
* Lower casing​
* Tokenization​
* Lemmatization​
* Stemming​

Data Transformation

After topic modeling, we will perform aggrigation of Score and Num\_comments in a day or in a week for each topic that will be obtained by the model. Those values will be expranatory variables that can increase acuracy of our stock prediction model​.

**For DJIA Dataset:**

Since the DJIA dataset will be used for time series analysis, we need to apply data cleaning and data transformation. In data cleaning step, since the data we are dealing with are stock price and they are moving every day, so we decided not to remove outliers. Otherwise, we may lose important information about stock prices trending analysis.​

Data Cleaning

* Checking and removing missing values​

Data Transformation

* Removing missing values​
* Selection

Time series models require stationarity of the data. The DJIA value as shown in Fig.1 , however, looks like non-stationary data. We will apply a data transformation technique to make it stationary, e.g. differencing.

In addition, we need to chose a target variable that will be predicted in our model.

Chart, line chart

Description automatically generated

Fig. 1: DJIA Close value against time

# WEEK 8 Predictive/descriptive Analytics System Group-Based Assignment

1. **Data cleaning results**

* **For News Dataset:**

The following data cleaning was performed on the Reddit news headlines. Fig. 1 shows the headlines before and after data cleaning. Special characters have been removed, and lemmatization and stemming have been correctly performed.

* + Step 1: Remove missing values
  + Step 2: Remove special characters
  + Step 3: Lemmatization and Stemming

Before:

C:\Users\dinuser\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B9B8D7DF.tmp

After:

C:\Users\dinuser\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\4A7C0C45.tmp

Fig. 1 News headlines before and after data cleaning.

* **For Stock Price Dataset:**

For the DJIA dataset, we checked for missing values and duplicates. The dataset was confirmed to contain no missing values and no duplicates. In addition, we decided not to remove outliers because the purpose of this analysis was to forecast time-series data, which contain time-series trends.

1. **Plan of Data Transformation:**
2. **For News Dataset:**

The data transformation and variable selection procedures to be implemented for this data are as follows. In order to perform K-means clustering as our topic modeling method, we need to vectorize the data. Also, to obtain better topic modeling results, we will remove words that are not important for the clustering from the vector.

* + Step 1: Remove stop words
  + Step 2: Apply Vectorization (Use TfidfVectorizer() to build vectorizer)
  + Step 3: Reduce features (Eg: Chi-squared value or Naïve Bayes methods)

1. **For Stock Price Dataset:**

Time series forecasting models assume that the data are stationary. Therefore, we conducted an Augmented Dickey-Fuller (ADF) test on the DJIA dataset to check whether the data are nonstationary. In this test, the null hypothesis is that the data is non-stationary, and we can obtain the p-value of the null hypothesis. The test result shows that the p-value is 0.40, as shown in Fig.2, so we cannot reject the null hypothesis. Therefore, we decided to compute the diffecence in order to obtain certainly stationary data.

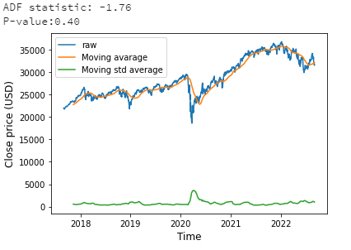


Fig.2 DJIS value time trend and the result of ADF test

* + Step 1: Compute difference
  + Step 2: Confirm that the data is stationary

# WEEK 10 Predictive/descriptive Analytics System Group-Based Assignment

1. **Results of valiable selection and data transformation**

**For stock price dataset:**

* Data Transformation – News Data
  + TF – IDF is a measure of originality of a word by comparing the number of times a word appears in a document with the number of documents the word appears in.

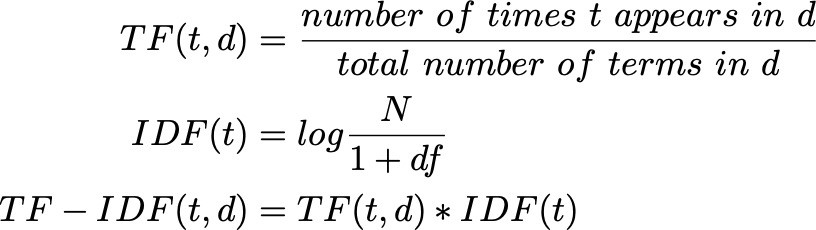


Fig1. Definition of TF-IDF

* + Here, we are using TfidfVectorizer() in Sklearn to build vectorizer and the vectorizer tokenizes the sentences and removes the stopwords. It also transforms sentences into lower case.
  + The shape of vector shows that we have 90,245 total news records and 6376 unique words after removing the stopwords.

1. Features Reduction – News Data
   1. Remove infrequent words. Because they’re so rare, the association between them and other words is dominated by noise. So, we kept words whose frequencies are greater than 10.
   2. Select features based on chi-squared statistics. Since we are going to use KMeans model to perform clustering, we need to select features to reduce the data complexity and avoid overfitting. In this case, 1000 features with highest chi-squared statistics are selected.

We will also run the model with all the features to see if features selection improves the performance of model. Since the purpose of variable selection is to avoid overfitting the topic model, and too few variable selections may result in less accurate clustering, we need to compare several numbers of selected variable and choose an appropriate number after topic modeling. The whole codes could be found in the appendix.

**For stock price dataset:**

We applied differencing (d = 1) to obtain stationary data. Differencing is a method of transforming a time series dataset. It can be used to remove the series dependence on time, so-called temporal dependence. Therefore, we can obtain stationary data eliminating (or reducing) trend and seasonality by this method. Compared to Fig. 2 in the week 8 submission, the trend of differenced price looks like stationary. In addition, we also performed ADF test on differenced price, and the p-value of ADF indicates it is stationary.

Chart

Description automatically generated

Fig5. The differenced price and the result of ADF test.

1. **Modelling Plan**:
   1. Topic Modeling:

We will perform KMeans model to get major topics in the News data set. This will allow us to get statistical information about news, which can be the number of news of each topics, the number of comments of each topic, the number of score of each topic, and so on.

* 1. Stock Price prediction:

We are going to use ARIMAX model, which can consider explanatory variables in addition to ARIMA model. The first and second terms are the same as ARIMA model, and, the ARIMAX model is characterized by a third term, which can improve prediction accuracy of the target variable by external variables.

Fig.6. The equation that represents ARIMAX model.

In our modeling, we will input statisticacl information that is obtained from our topic modeling. We must note that the ARIMAX model applies linear regression to the residuals of the ARIMA model. Therefore, the external input variables have an assumption of normality. We have already prepared the code for the normalization by Box-cox transformation, but the results of that transformation can only be obtained after the topic modeling has been run and will be presented in next week's submission.

# WEEK 12 Predictive / Descriptive Analytics System Group-Based Assignment

1. **For news dataset:**

We built KMeans model to perform the topic analysis based on news dataset.

Firstly, We used the data of latest 3 months to be the testing set and the remains to be the training dataset. The unknown data then can be predicted by applying testing dataset on the model. The MSE (Mean Squared Errors) value is used to evaluate the model performance. We chose to not use SSE (Sum of Squared Errors) because SSE could be influenced by the number of inputs. So MSE is the better choice for us. To make the results become more interpretative, we use range (1, 20, 1) to be the number of clusters in KMeans model.

After training the model without features reduction, we plotted the trending line for MSE value to see model's performance with the change of number of clusters. From the figure we can see that the model performs better when there are more clusters. The top terms per cluster were exported. From the results we can have an initial understanding that there are several clusters which regard topics such as stocks, market, economy and so on.

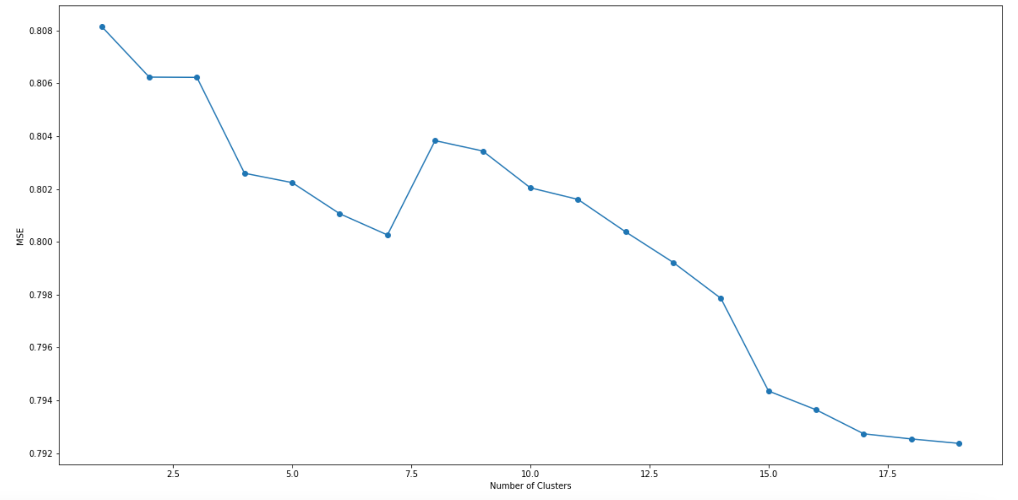


Figure 1 The number of clusters vs MSE

Then we trained the model with features reduction to see if features reduction helps to improve the model’s performance. The MSE trending line shows that with the features reduction, the model's performance becomes better than before because the MSE value under features reduction started at 0.36 when number of clusters equals to 1 while the MSE value started at 0.808 when we didn’t use features reduction. Then we chose 14 to be the final value of clusters number because the slope of MSE trending line becomes smaller after 14.

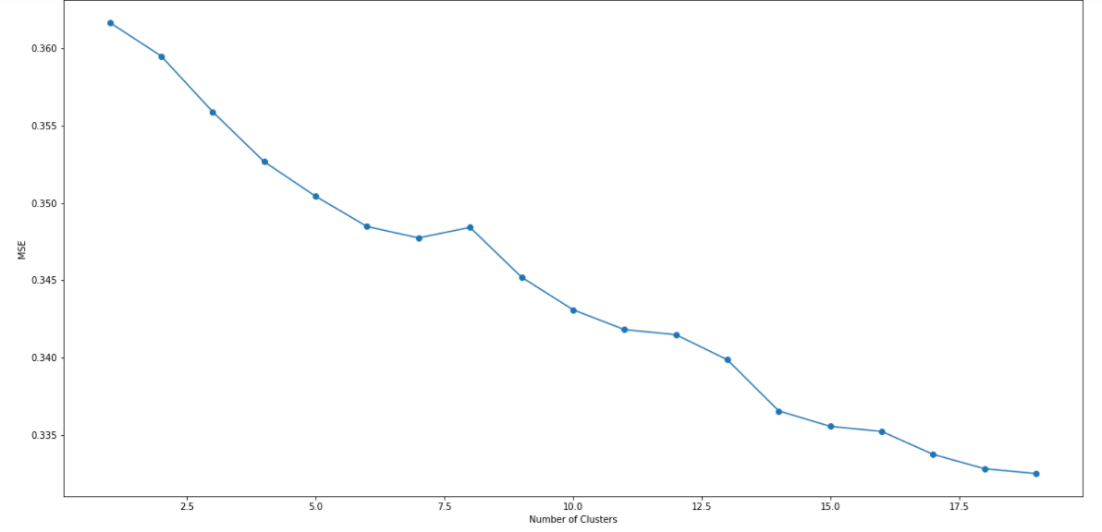


Figure 2 The number of clusters vs MSE

**Example of topics (top terms per topic):**

Table1 Examples of top 10 terms in clusters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cluster 2 | Cluster 7 | Cluster 11 | Cluster 12 | Cluster 13 |
| Stock | Price | China | Market | Economi |
| Buy | Oil | Economics | Stock | World |
| Trade | Rise | Trade | Weekli | Global |
| Best | High | Bank | Review | China |
| Tip | Home | New | Crash | Grow |
| Investor | Year | Year | Hous | Year |
| Invest | Fall | World | Global | Growth |
| Week | Gold | Job | Labor | Recess |
| Rise | Hous | Billion | Investor | Covid |
| Exchang | Surg | Bitcoin | Bond | Slow |

Topic 2: stock, buy, trade, best, tip, investor, invest, week, rise, exchang

The key words in this topic indicate that the topic is related to investment in stock and buy in or sale out operations.

Topic 7: price, oil, rise, high, home, year, fall, gold, hous, surg

The key words in this topic indicate that the topic is related to oil, gold and house prices' change.

Topic 11: china, econom, trade, bank, new, year, world, job, billion, bitcoin

The key words in this topic indicate that the topic is related to countries' economic and job opportunities. It also talks about cryptocurrencies such as Bitcoin.

Topic 13: economi, world, global, china, grow, year, growth, recess, covid, slow

The key words in this topic indicate that the topic is related to the growth and recession of economic in the world which may be influenced by the pandemic of COVID virus.

These are several examples of the topics that may related to stock price. In the following process we used correlation value to evaluate which topics have strongest relationship with stock price.

1. **For stock price dataset:**

We built three models, ARIMA, ARIMAX, and Null, to predict stock prices and to evaluate their accuracy.

* 1. *Modeling:*
     1. ARIMA Model

As shown in equation (1), this model combines three models: an autoregressive model (AR model), a moving average model (MA model), and a integrated model (I model). It is capable of predicting future values from past time series data. This model includes the parameters p, which determines how many past values are referenced in the AR model, q, which determines how many past values are referenced in the MA model, and d, which determines the degree of differencing. As mentioned in the Data Transformation section, we confirmed that target time series data are stationary at d=1. We computed the AIC and BIC for d=1 and for p and q between 0 and 10 exhaustively to determine the p and q. As shown in Tables 2 and 3, p=2 and q=6 have the highest scores in AIC and the second highest scores in BIC, we decided to adopt p=2 and q=6. Fig.3 shows the summary of the ARIMA model. As shown in Fig.3, Since the P-value of ar.L1 and ar.L2 is 0.000, in that AR model, the data from one and two days before have a clear correlation with the stock price on that day. L1, ma.L2, and ma.L6 have a P-value of 0.000, the data from one, two, and six days before are strongly correlated with the stock price of that day in the MA model. According to the Q-Q plot, as shown in Fig.4, the blue plot roughly corresponds to the red line, which means that the residues are approximately normally distributed.

…Equation (1)

Graphical user interface, text, application

Description automatically generated with medium confidenceTable2 Combination of p and q sorted by AIC Table3 Combination of p and q sorted by BIC

Table

Description automatically generated Chart, line chart

Description automatically generated

Figure 3 Sammary of the ARIMA model Figure 4 Q-Q plot of the ARIMA model

* + 1. ARIMAX Model

As shown in equation (2), this model includes a linear regression by external variable (Xm) in addition to the ARIMA model mentioned above. We used our topic modeling outputs as this external variable. Outputs of our topic model are the cluster numbers to which the headline of each article belongs. Therefore, a data transformation is needed to input those numbers into this model. First, we aggregated the number of comments, the number of posts, and the score, which is the sum of upvotes and downvotes, for each cluster number and each day dimensions. We then removed as outsliers that are below Q1 - 1.5 IQR and above Q3 + 1.5 IQR, and applied a box-cox transformation and a min-max scaler. In order to predict future stock prices, the news data for the date to be predicted cannot be used, so the date, which is the index of the news data, is added by one and merged with the stock price data. For example, to predict the stock price on 9/2/2022, the model would be input with the news data on 9/1/2022. We have built two ARIMAX models. One model includes all the output attributes of the topic model, and the other includes the two output attributes that are most correlated with the stock price, based on their P-values.

…Equation (2)

The summary of the first model is shown in Fig. 5, where the P-values of the external explanatory variables indicate that many of the attributes are not related to stock prices with high probability. And there are too many attributes of the inputs in the linear regression terms, which may cause overfitting of the predicted results to the training data. Therefore, we re-created the ARIMAX model including only the three largest attributes among the P-values: comments\_11, comments\_7, and count\_2. We did not include the results, but in this model, the P-value of comments\_7 was so large (0.770) that we ended up building an ARIMAX model that included only comments\_11 and count\_2. In addition, the Q-Q plot in Fig. 6 suggests that the residues in this model are approximately normally distributed.

Table

Description automatically generatedChart, line chart

Description automatically generated

Figure 5 Sammary of the ARIMAX model Figure 6 Q-Q plot of the ARIMAX model

Fig. 7 shows a summary of the model including only the two external variables that were most correlated with the stock prices. The P-value of comments\_7 is 0.018, indicating that it is correlated with stock prices when α is assumed to be 0.05. The comments\_7 is the aggregated number of comments on topic 7, which is related to oil, gold and house prices' change. Therefore, its coefficient is negative, suggesting that the stock price tends to fall on the day after a day with a lot of news classified as Topic 7. Although its count\_2 does not have a clear correlation with the stock price when α is defined as 0.05, its P-value is lower than those of other attributes and might have a minor relationship with the stock price. Topic 2, as already discussed, relates to the trade between stock price and investment. In addition, the Q-Q plot in Fig. 8 indicates that the residues in this model are approximately normally distributed.

Table

Description automatically generatedChart, line chart

Description automatically generated

Figure 7 Sammary of the ARIMAX model Figure 8 Q-Q plot of the ARIMAX model

* + 1. Null Model

This model is used to evaluate the prediction accuracy of our ARIMA and ARIMAX models, which randomly predict future stock prices based on the distribution of our training data set. We computed the difference between the stock prices on two consecutive dates and calculated the standard deviation and mean of the differences. We then produced a normal distribution with the obtained standard deviation and the mean. Assuming that the distribution of the differencrs is normally distributed, we created a model that randomly predicts differences in unknown data. The probability density function of the normal distribution is expressed as in Equation (3), and the integrated probability density by x is expressed as in Equation (4). Our Null model, as shown in Equation (5), predicts a difference with the stock price on the previous day by substituting a random value between 0 and 1, r, for the inverse function of FX (x). For example, that model simply predicts the next day's stock price by randomly sampling based on the distribution of train data, while other our models predict the next day's stock price after learning the time series trend based on train data.

* 1. Testing and Model Evalation:
     1. Testing technique

We have adopted a rolling forecasting technique in the forecasting of the stock price in the test data, which suppresses the overfitting of the model to the training data. The rolling forecasting is a method that adds data up to the previous day of forecasting in the test data in addition to the original training data. As shown in Figure 9, for example, if the data for January 5 is to be forecasted, data up to January 4 is considered as the training data; if the data for January 6 is to be forecasted, data up to January 5 is included in the training data. This technique is applicable in models that make time-series forecasts, because even in test data, the data up to today is already known at the time of forecasting tomorrow's data. Chart

Description automatically generated

Figure 9 An example of a roling forecasting

Evaluation with Root mean square error (RMSE):

First, we calculated the Root mean square error (RMSE) as a metric to evaluate the prediction accuracy of our model. The RMSE on the train data is shown in Table 4 and the RMSE on the test data in Table 5. Table 4 shows that the RMSE values are US$160 lower for all of our models compared to the null model. This fact indicates that our models predict with better accuracy than models that simply forecast randomly from the actual distribution of stock prices. Among our models, the ARIMAX model with all features has the lowest RMSE, but the difference between the models is quite small. The prediction results for the test data have the same tendency as the prediction results for the training data, as shown in Table 5, our model predicts better than the Null model, and there is no significant difference in prediction accuracy between the ARIMA and ARIMAX models. In addition, the RMSE of the test data is about US$50 higher than the RMSE of the train data. Generally, if there is a difference in forecast accuracy between train data and test data, it may be due to overfitting of the model to the train data. However, we believe that in our model, this is simply due to the large fluctuations in stock prices in the test data, making it difficult to forecast. This is because our ARIMA model completely avoids overfitting to the train data set by the rolling forecast, which always predicts by data up to the day before the forecast data, due to its rolling forecast.

|  |  |
| --- | --- |
| Model | RMSE (US$) |
| ARIMA | 378.5 |
| ARIMAX (All features) | 367.9 |
| ARIMAX (Top two features) | 374.0 |
| Null | 511.0 |

Table4 RMSE values of train data Table5 RMSE values of test data

|  |  |
| --- | --- |
| Model | RMSE (US$) |
| ARIMA | 329.4 |
| ARIMAX (All features) | 327.1 |
| ARIMAX (Top two features) | 328.9 |
| Null | 478.4 |

* + 1. Evaluation with correlation coefficients (r)

Figures 10-13 show the trends of the actual value of the stock price and the price predicted by each of our model. In these plots, it is difficult to identify differences in prediction accuracy between our models. However, only in the results of the Null model in Figure 13 the differences between the actual and predicted values are relatively large.

Shape, arrow

Description automatically generated Shape, arrow

Description automatically generated

Figure 10 Actual and predicted price in ARIMA Figure 11 Actual and predicted price in ARIMAX1

Shape, arrow

Description automatically generatedShape, arrow

Description automatically generated

Figure 12 Actual and predicted price in ARIMAX2 Figure 13 Actual and predicted price in Null model

We also calculated correlation coefficients (r) to quantitatively analyze the difference between this predicted and actual value. Figures 14-17 show scatter plots with actual values on the x-axis and predicted values on the y-axis, each graph including its linear regression equation and correlation coefficient (r) in the legend. The r-values of our ARIMA and ARIMAX models are approximately 0.94, with no significant differences among the models. On the other hand, the r-value of the null model is 0.89, which is significantly lower than the other models.

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Figure 14 Actual vs predicted price in ARIMA Figure 15 Actual vs predicted price in ARIMAX1

Chart, scatter chart

Description automatically generated Chart, scatter chart

Description automatically generated

Figure 16 Actual vs predicted price in ARIMAX2 Figure 17 Actual vs predicted price in Null model

* 1. Discussion

In the evaluation discussed above, our ARIMA and ARIMAX models have significantly higher prediction accuracy compared to the Null model. As mentioned in 2.1.3, the Null model predicts values by random sampling based on the distribution of training data, while our ARIMA and ARIMAX models predict values based on results learned from past trends using the autoregression and moving average models. Therefore, the high prediction accuracy in the ARIMA and ARIMAX models indicates that there is a time-series dependence in their target variables. In the AR model, since we are using p=2, our target variable includes the dependent relationship between an observation and two lagged observations. Thus, the good prediction accuracy in the ARIMA and ARIMAX models indicates that there is a time-series dependence in their target variables. In the MA model with q=6, there is the dependency between an observation and a residual error from a moving average model applied to six lagged observations. The time-series dependence of these target variables improved the prediction accuracy of the ARIMA and ARIMAX models, which performed better than the Null model that simply predicts randomly from a distribution of train data.

On the other hand, there is no distinct difference in the accuracy of the ARIMA and ARIMAX models. This fact means that external variables based on our topic modeling results, which were input in addition to the time-series dependence, had no effect on improving the accuracy of the forecast. However, we cannot conclude that economic news trends do not affect stock prices in our project. This is because our topic modeling results are unstable, since changing the number of clusters to be classified or the random seed can significantly change the clusters obtained. For example, adopting supervised learning, using pre-prepared labels that are likely to be related to DJIA stock prices for topic modeling, might improve the performance of topic extraction and, consequently, improve the accuracy of stock price prediction. In addition, the current model considers news trends from one day prior to the stock price to be predicted, but it may take multiple days for economic news trends to be reflected in the stock price. To solve this problem, it would be useful to compare multiple patterns of time lags between news data and the stock prices to be forecasted. We believe that these strategies are future work to improve the accuracy of stock price forecasts by topic modeling.

# WEEK 13 Predictive/descriptive Analytics System Group-Based Assignment

**Purpose:**

To complete all the modules of the Predictive/Descriptive Analytics System

**Tasks:**

1. Submit the final project report of your team’s Predictive /Descriptive Analytics system by the end of week 13. In this week the team is supposed to complete the visualization of the Analytics system.
2. Project report feedback and more instructions about the final submission of the project report, datasets, scripts etc. will be provided by the instructor over the e-mail.

# WEEK 14 Predictive / Descriptive Analytics System Group-Based Assignment

**Purpose:**

To provide a demonstration of your team’s Predictive /Descriptive Analytics System

**Tasks:**

1. The team should prepare to demonstrate the designed Predictive/Descriptive Analytics System in this capstone course.
2. The demonstration should be presented in class. The power point presentation should be uploaded through the box/Canvas.
3. Each team should submit the data sets and the scripts into their respective box folder.
4. Every team will have an opportunity to go through the demonstration of the Predictive/Descriptive Analytics system designed by different teams.
5. Team members should submit the peer evaluation form latest by Thursday 11:59 PM EST in week 14.
6. More instructions will be provided by the instructor over the e-mail.

**Reference**

[1] Eshan Singh, Nursulu Kuzhagaliyeva, S. Mani Sarathy,

Chapter 9 - Using deep learning to diagnose preignition in turbocharged spark-ignited engines, Editor(s): Jihad Badra, Pinaki Pal, Yuanjiang Pei, Sibendu Som, Artificial Intelligence and Data Driven Optimization of Internal Combustion Engines, Elsevier, 2022, Pages 213-237

**Appendix:**

# %%

import numpy as np

import pandas as pd

import nltk

import seaborn as sns

import re

import matplotlib.pyplot as plt

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from sklearn import preprocessing

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from nltk.stem.lancaster import LancasterStemmer

from sklearn.cluster import KMeans

# %%

#from imblearn.combine import SMOTEENN, SMOTETomek

#from imblearn.pipeline import make\_pipeline

from nltk.tokenize import WordPunctTokenizer

#from nltk.stem import LancasterStemmer

# %%

df\_origin=pd.read\_csv('Reddit\_news\_2.csv') #Text data

df\_origin

# %%

df=df\_origin[['id', 'title', 'date']]

# %%

df

# %%

df=df.dropna()

df

# %%

X=df.drop(['id', 'date'],axis=1)

# %%

messages=X.copy()

messages.reset\_index(inplace=True)

# %%

#Word Tokenization

from nltk.tokenize import word\_tokenize

from nltk.corpus import wordnet

import nltk

#nltk.download('wordnet')

#nltk.download("wordnet",'wordnet')

#nltk.data.path.append('.wordnet/nltk\_data/')

#nltk.download('omw-1.4')

corpus2 = []

#nltk.download('stopwords')

for i in range(0, len(messages)):

review = re.sub('[^a-zA-Z]', ' ', messages['title'][i]) #Remove Special Characters

review = ''.join(review)

corpus2.append(review)

df['title\_2']=corpus2

df['title\_2'] = df['title\_2'].astype(str).str.lower()

df['title\_lemma']=corpus2

lemma = nltk.wordnet.WordNetLemmatizer()

for i in range(0, len(df)):

df['title\_lemma'][i] = lemma.lemmatize(df['title\_2'][i])

# %%

from nltk.corpus import stopwords

from nltk.tokenize import RegexpTokenizer

regexp = RegexpTokenizer('\w+')

df['title\_3']=df['title\_lemma'].apply(regexp.tokenize)

stopwords = nltk.corpus.stopwords.words("english")

my\_stopwords = ['news','say','said','says','think','thought','like','make','made','result']

stopwords.extend(my\_stopwords)

df['title\_4'] = df['title\_3'].apply(lambda x:[word for word in x if word not in stopwords])

ps = PorterStemmer()

df['stemmed'] = df['title\_4'].apply(lambda x: [ps.stem(y) for y in x])

# %%

print(df['title'][:20])

print(df['stemmed'][:20])

# %%

minda = min(df['date'])

maxda = max(df['date'])

mindat = minda[0:10]

maxdat = maxda[0:10]

from datetime import datetime

from dateutil.relativedelta import relativedelta

x = datetime.strptime(mindat, '%Y-%m-%d')

y = datetime.strptime(maxdat, '%Y-%m-%d')

if \_\_name\_\_ == "\_\_main\_\_":

flag = str(y - relativedelta(months=+3))

print(y)

print(flag)

# %%

train = pd.DataFrame(df[df.date < flag])

test = pd.DataFrame(df[df.date > flag])

train0 = train.drop(['date'], axis=1)

test0 = test.drop(['date'], axis=1)

# %%

train0['stemmed\_string'] = train0['stemmed'].apply(lambda x: ' '.join([item for item in x if len(item)>2]))

test0['stemmed\_string'] = test0['stemmed'].apply(lambda x: ' '.join([item for item in x if len(item)>2]))

all\_words\_train = ' '.join([word for word in train0['stemmed\_string']])

all\_words\_test = ' '.join([word for word in test0['stemmed\_string']])

# %%

from nltk.probability import FreqDist

#nltk.download('punkt')

all\_token\_train = nltk.tokenize.word\_tokenize(all\_words\_train)

all\_token\_test = nltk.tokenize.word\_tokenize(all\_words\_test)

fdist = FreqDist(all\_token\_train)

fdist

# %%

#Tokenization and Vectorizer

from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer

vectorizer1 = TfidfVectorizer(token\_pattern=r"(?u)\b\w+\b", stop\_words='english')

vectorizer2 = TfidfVectorizer(token\_pattern=r"(?u)\b\w+\b", stop\_words='english')

comment\_train = vectorizer1.fit\_transform(train0['stemmed\_string']).todense()

comment\_test = vectorizer2.fit\_transform(test0['stemmed\_string']).todense()

vocab1 = np.array(vectorizer1.get\_feature\_names())

vocab2 = np.array(vectorizer2.get\_feature\_names())

# %%

df\_vector\_train = pd.DataFrame(comment\_train, columns= vocab1)

df\_vector\_test = pd.DataFrame(comment\_test, columns= vocab2)

df\_vector\_train = pd.DataFrame(df\_vector\_train, columns= df\_vector\_test.columns)

df\_vector\_train = df\_vector\_train.dropna(how='all', axis = 1)

df\_vector\_test = pd.DataFrame(df\_vector\_test, columns= df\_vector\_train.columns)

df\_vector\_test = df\_vector\_test.dropna(how='all', axis = 1)

# %%

df\_vector\_train

# %%

MSE = []

I = range(1,20,1)

MYSEED = 1001

for i in I:

model = KMeans(n\_clusters=i, init='k-means++', max\_iter=100, n\_init=1,random\_state=MYSEED)

model.fit(df\_vector\_train)

MSE.append(model.inertia\_/len(df\_vector\_train))

# %%

plt.figure(figsize=(20,10))

plt.plot(I, MSE,marker='o')

plt.xlabel("Number of Clusters")

plt.ylabel("MSE")

plt.show()

# %%

true\_k = 17

MYSEED = 1001

model = KMeans(n\_clusters=true\_k, init='k-means++', max\_iter=100, n\_init=1,random\_state=MYSEED)

model.fit(df\_vector\_train)

# %%

print("Top terms per cluster:")

order\_centroids = model.cluster\_centers\_.argsort()[:, ::-1]

terms = df\_vector\_train.columns

for i in range(true\_k):

print("Cluster %d:" % i),

for ind in order\_centroids[i, :10]:

print(' %s' % terms[ind]),

print

# %%

prediction\_test = model.predict(df\_vector\_test)

prediction\_train = model.predict(df\_vector\_train)

# %%

df\_result\_train = pd.DataFrame(df[df.date < flag])

df\_result\_train['result'] = prediction\_train

df\_result\_test = pd.DataFrame(df[df.date > flag])

df\_result\_test['result'] = prediction\_test

# %%

X = df\_vector\_test

y = df\_result\_test['result']

X\_train = df\_vector\_train

y\_train = df\_result\_train['result']

# %%

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

# 1000 features with highest chi-squared statistics are selected

chi2\_selector = SelectKBest(chi2, k=1000)

X\_kbest = chi2\_selector.fit\_transform(X\_train, y\_train)

print(X\_kbest)

print('Original number of features:', X\_train.shape)

print('Reduced number of features:', X\_kbest.shape)

reduced\_columns = chi2\_selector.get\_feature\_names\_out(df\_vector\_train.columns)

# %%

chi2\_selector.get\_feature\_names\_out(df\_vector\_train.columns)

# %%

df\_reducedvector\_train = pd.DataFrame(X\_train, columns= reduced\_columns)

df\_reducedvector\_train

# %%

MSE2 = []

I = range(1,20,1)

MYSEED = 2002

for i in I:

model2 = KMeans(n\_clusters=i, init='k-means++', max\_iter=100, n\_init=1,random\_state=MYSEED)

model2.fit(df\_reducedvector\_train)

MSE2.append(model2.inertia\_/len(df\_reducedvector\_train))

# %%

plt.figure(figsize=(20,10))

plt.plot(I, MSE2,marker='o')

plt.xlabel("Number of Clusters")

plt.ylabel("MSE")

plt.show()

# %%

true\_k = 14

MYSEED = 2002

model2 = KMeans(n\_clusters=true\_k, init='k-means++', max\_iter=100, n\_init=1,random\_state=MYSEED)

model2.fit(df\_reducedvector\_train)

# %%

print("Top terms per cluster:")

order\_centroids = model2.cluster\_centers\_.argsort()[:, ::-1]

terms = df\_reducedvector\_train.columns

for i in range(true\_k):

print("Cluster %d:" % i),

for ind in order\_centroids[i, :10]:

print(' %s' % terms[ind]),

print

# %%

df\_reducedvector\_test = pd.DataFrame(df\_vector\_test, columns= reduced\_columns)

prediction2\_train = model2.predict(df\_reducedvector\_train)

prediction2\_test = model2.predict(df\_reducedvector\_test)

df\_result\_train['reducedresult'] = prediction2\_train

df\_result\_test['reducedresult'] = prediction2\_test

# %%

df\_origin\_train = pd.DataFrame(df\_origin[df\_origin.date < flag])

df\_origin\_test = pd.DataFrame(df\_origin[df\_origin.date > flag])

# %%

df\_result\_train['comments']= df\_origin\_train['num\_comments']

df\_result\_train['score']= df\_origin\_train['score']

df\_result\_train['count']= 1

df\_result\_train['date'] = df\_origin\_train['date'].str[:10]

df\_result\_test['comments']= df\_origin\_test['num\_comments']

df\_result\_test['score']= df\_origin\_test['score']

df\_result\_test['count']= 1

df\_result\_test['date'] = df\_origin\_test['date'].str[:10]

# %%

df\_result\_train[65:]

# %%

df\_aggregated\_train = df\_result\_train.groupby(['date','reducedresult'], as\_index=False).agg({"comments": "sum"})

df\_aggregated\_test = df\_result\_test.groupby(['date','reducedresult'], as\_index=False).agg({"comments": "sum"})

df\_aggregated\_train['score'] = df\_result\_train.groupby(['date','reducedresult'], as\_index=False).agg({"score": "sum"})['score']

df\_aggregated\_test['score'] = df\_result\_test.groupby(['date','reducedresult'], as\_index=False).agg({"score": "sum"})['score']

df\_aggregated\_train['count'] = df\_result\_train.groupby(['date','reducedresult'], as\_index=False).agg({"count": "sum"})['count']

df\_aggregated\_test['count'] = df\_result\_test.groupby(['date','reducedresult'], as\_index=False).agg({"count": "sum"})['count']

df\_aggregated\_train['reducedresult'] = df\_aggregated\_train['reducedresult'].apply(str)

df\_aggregated\_test['reducedresult'] = df\_aggregated\_test['reducedresult'].apply(str)

# %%

df\_aggregated\_train

# %%

df\_aggregated\_train.to\_csv('ClusteringResult\_train.csv', sep=',', header=True, index=False)

df\_aggregated\_test.to\_csv('ClusteringResult\_test.csv', sep=',', header=True, index=False)

# %%

df\_aggregated\_train = pd.read\_csv('ClusteringResult\_train.csv', delimiter=',', header=0)

df\_aggregated\_test = pd.read\_csv('ClusteringResult\_test.csv', delimiter=',', header=0)

df\_aggregated\_train['reducedresult'] = df\_aggregated\_train['reducedresult'].apply(str)

df\_aggregated\_test['reducedresult'] = df\_aggregated\_test['reducedresult'].apply(str)

# %%

df\_output\_train = df\_aggregated\_train.pivot(index='date',columns='reducedresult', values=['comments','score','count'])

df\_output\_train = df\_output\_train.fillna(0)

df\_output\_train.columns = ["\_".join((j,k)) for j,k in df\_output\_train.columns]

df\_output\_test = df\_aggregated\_test.pivot(index='date',columns='reducedresult', values=['comments','score','count'])

df\_output\_test = df\_output\_test.fillna(0)

df\_output\_test.columns = ["\_".join((j,k)) for j,k in df\_output\_test.columns]

df\_output\_train = df\_output\_train.loc[:, (df\_output\_train != df\_output\_train.iloc[0]).any()]

df\_output\_test = df\_output\_test.loc[:, (df\_output\_test != df\_output\_test.iloc[0]).any()]

df\_output\_train = pd.DataFrame(df\_output\_train, columns= df\_output\_test.columns)

df\_output\_test = pd.DataFrame(df\_output\_test, columns= df\_output\_train.columns)

# %%

df\_output\_train[20:]

# %%

fig = plt.figure(figsize = (15,20))

ax = fig.gca()

df\_output\_test.hist(ax = ax,bins = 30)

# %%

from feature\_engine import transformation as vt

df\_transformed\_train = df\_output\_train

df\_transformed\_train = df\_transformed\_train.add(0.0001)

df\_transformed\_test = df\_output\_test

df\_transformed\_test = df\_transformed\_test.add(0.0001)

tf = vt.BoxCoxTransformer()

tf.fit(df\_transformed\_train)

df\_transformed\_train = tf.transform(df\_transformed\_train)

df\_transformed\_test = tf.transform(df\_transformed\_test)

# %%

fig = plt.figure(figsize = (15,20))

ax = fig.gca()

df\_transformed\_train.hist(ax = ax,bins = 10)

# %%

for col in df\_transformed\_train.columns:

Q1 = df\_transformed\_train[col].quantile(0.25)

Q3 = df\_transformed\_train[col].quantile(0.75)

IQR = Q3 - Q1

upper\_b = Q3 + 1.5\*IQR

lower\_b = Q1 - 1.5\*IQR

median = df\_transformed\_train[col].median(axis = 0)

for i in range(0, len(df\_transformed\_train)):

if df\_transformed\_train[col][i] > upper\_b or df\_transformed\_train[col][i] < lower\_b:

df\_transformed\_train[col][i] = median

# %%

fig = plt.figure(figsize = (15,20))

ax = fig.gca()

df\_transformed\_train.hist(ax = ax,bins = 10)

# %%

df\_transformed\_train = df\_transformed\_train.loc[:, (df\_transformed\_train != df\_transformed\_train.iloc[0]).any()]

df\_transformed\_test = pd.DataFrame(df\_transformed\_test, columns= df\_transformed\_train.columns)

# %%

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(df\_transformed\_train)

np\_input\_train = scaler.transform(df\_transformed\_train)

np\_input\_test = scaler.transform(df\_transformed\_test)

# %%

df\_input\_train = df\_transformed\_train

counter = 0

for col in df\_transformed\_train.columns:

df\_input\_train[col] = np\_input\_train[:,counter]

counter = counter +1

df\_input\_test = df\_transformed\_test

counter = 0

for col in df\_transformed\_test.columns:

df\_input\_test[col] = np\_input\_test[:,counter]

counter = counter +1

# %%

fig = plt.figure(figsize = (15,20))

ax = fig.gca()

df\_input\_train.hist(ax = ax,bins = 10)

# %%

#Model for stock price prediction

import pandas as pd

import matplotlib.pyplot as plt

from datetime import datetime

from statsmodels.tsa.seasonal import seasonal\_decompose

import statsmodels.api as sm

from statsmodels.tsa.stattools import adfuller

import warnings

warnings.filterwarnings("ignore")

import datetime

# %%

price\_df = pd.read\_csv('HistoricalPrices.csv', index\_col = 'Date', parse\_dates = True)

price\_df.index = pd.to\_datetime(price\_df.index)

price\_df = price\_df.rename(columns={' Close': 'Close' })

price\_df = price\_df.sort\_index(axis = 0)

# %%

df\_input\_train = df\_input\_train.set\_index(pd.to\_datetime(df\_input\_train.index) + datetime.timedelta(days=1))

df\_input\_test = df\_input\_test.set\_index(pd.to\_datetime(df\_input\_test.index) + datetime.timedelta(days=1))

# %%

price\_df\_train = price\_df.join(df\_input\_train)

price\_df\_train = price\_df\_train.dropna()

exval\_df\_train = price\_df\_train.drop([' Open',' High',' Low','Close'],axis = 1)

endog\_df\_train = price\_df\_train['Close']

price\_df\_test = price\_df.join(df\_input\_test)

price\_df\_test = price\_df\_test.dropna()

exval\_df\_test = price\_df\_test.drop([' Open',' High',' Low','Close'],axis = 1)

endog\_df\_test = price\_df\_test['Close']

# %%

endog\_df\_train = pd.DataFrame(price\_df\_train['Close'])

endog\_df\_test = pd.DataFrame(price\_df\_test['Close'])

# %%

endog\_df\_train

# %%

ma = endog\_df\_train.Close.rolling(50).mean()

ma\_std = endog\_df\_train.Close.rolling(50).std()

plt.plot(endog\_df\_train.Close, label= 'raw')

plt.plot(ma,label = 'Moving avarage')

plt.plot(ma\_std, label = 'Moving std average')

plt.ylabel('Close price (USD)', fontsize = 12)

plt.xlabel('Time', fontsize = 12)

plt.legend(loc='best')

test\_result = adfuller(endog\_df\_train.Close,maxlag=9)

print('ADF statistic: ' + str("{:.2f}".format(test\_result[0])) +'\n'

'P-value:' + str("{:.2f}".format(test\_result[1])))

# %%

endog\_df\_train\_dif = endog\_df\_train

endog\_df\_train\_dif['dif']= endog\_df\_train\_dif.Close.diff().dropna()

#endog\_df = endog\_df[1:]

endog\_df\_train = endog\_df\_train.drop('dif',axis = 1)

# %%

ma\_diff\_train = endog\_df\_train\_dif.dif.rolling(50).mean()

ma\_std\_diff\_train = endog\_df\_train\_dif.dif.rolling(50).std()

endog\_df\_train\_dif = endog\_df\_train\_dif[1:]

plt.plot(endog\_df\_train\_dif.dif, label= 'raw')

plt.plot(ma\_diff\_train,label = 'Moving avarage')

plt.plot(ma\_std\_diff\_train, label = 'Moving std average')

plt.ylabel('Close difference price (USD)', fontsize = 12)

plt.xlabel('Time', fontsize = 12)

plt.legend(loc='best')

test\_result = adfuller(endog\_df\_train\_dif.dif)

print('ADF statistic: ' + str("{:.10f}".format(test\_result[0])) +'\n'

'P-value:' + str("{:.10f}".format(test\_result[1])))

# %%

sm.graphics.tsa.plot\_acf(endog\_df\_train\_dif .dif, lags=30)

plt.xlabel('lags', fontsize=12)

plt.ylabel('acf', fontsize=1)

sm.graphics.tsa.plot\_pacf(endog\_df\_train\_dif .dif, lags=30)

plt.xlabel('lags', fontsize=12)

plt.ylabel('p-acf', fontsize=12)

# %%

from statsmodels.tsa.arima.model import ARIMA

order\_aic\_bic=[]

# Loop over p values from 0-11

for p in range(11):

# Loop over q values from 0-11

for q in range(11):

# create and fit ARMA(p,q) model

model = ARIMA(endog\_df\_train.Close, order=(p,1,q)) #because adf test showed that d=1

results = model.fit()

# Append order and results tuple

order\_aic\_bic.append((p,q,results.aic, results.bic))

# Construct DataFrame from order\_aic\_bic

order\_df = pd.DataFrame(order\_aic\_bic,

columns=['p','q','AIC','BIC'])

# %%

order\_df.sort\_values('AIC')

# %%

order\_df.sort\_values('BIC')

# %%

from statsmodels.tsa.arima.model import ARIMA

arima = ARIMA(endog\_df\_train.Close, order=(2,1,6))

arima\_fit=arima.fit()

print(arima\_fit.summary())

arima\_fit.plot\_diagnostics(figsize=(15, 12))

plt.show()

# %%

import datetime

history = [x for x in endog\_df\_train.Close]

predictions1 = list()

for t in endog\_df\_test.index:

model = ARIMA(history, order=(2,1,6))

model\_fit = model.fit()

output = model\_fit.forecast(dynamic=True)

yhat = output[0]

predictions1.append(yhat)

obs = endog\_df\_test.Close[t]

history.append(obs)

# %%

from matplotlib.pyplot import figure

figure(figsize=(8, 5), dpi=80)

plt.plot(endog\_df\_test.index, endog\_df\_test.Close, label = 'Actual', marker='D')

plt.plot(endog\_df\_test.index, predictions1, label = 'Predicted', color='red', marker='D')

plt.ylabel('Close price (USD)', fontsize = 12)

plt.xlabel('Date', fontsize = 12)

plt.legend(loc='best')

plt.show()

# %%

from statsmodels.tsa.arima.model import ARIMA

arimax = ARIMA(endog = endog\_df\_train.Close, exog = exval\_df\_train, order=(2,1,6))

arimax\_fit=arimax.fit()

print(arimax\_fit.summary())

arimax\_fit.plot\_diagnostics(figsize=(15, 12))

plt.show()

# %%

history1 = endog\_df\_train

history2 = exval\_df\_train

predictions2 = list()

for t in endog\_df\_test.index:

model = ARIMA(endog = history1, exog = history2, order=(2,1,6))

model\_fit = model.fit()

output = model\_fit.forecast(steps= 1, exog = exval\_df\_test[t:t],dynamic=False)

output = output.to\_frame()

yhat = output.iloc[0,0]

predictions2.append(yhat)

obs1 = endog\_df\_test[t:t]

history1 = history1.append(obs1)

obs2 = exval\_df\_test[t:t]

history2 = history2.append(obs2)

# %%

from matplotlib.pyplot import figure

figure(figsize=(8, 5), dpi=80)

plt.plot(endog\_df\_test.index, endog\_df\_test.values, label = 'Actual', marker='D')

plt.plot(endog\_df\_test.index, predictions2, label = 'Predicted', color='red', marker='D')

plt.ylabel('Close price (USD)', fontsize = 12)

plt.xlabel('Date', fontsize = 12)

plt.legend(loc='best')

plt.show()

# %%

exval\_df\_train\_reduced = exval\_df\_train[["comments\_11","count\_2"]]

exval\_df\_test\_reduced = exval\_df\_test[["comments\_11","count\_2"]]

# %%

from statsmodels.tsa.arima.model import ARIMA

arimax1 = ARIMA(endog = endog\_df\_train.Close, exog = exval\_df\_train\_reduced, order=(2,1,6))

arimax1\_fit=arimax1.fit()

print(arimax1\_fit.summary())

arimax1\_fit.plot\_diagnostics(figsize=(15, 12))

plt.show()

# %%

history1 = endog\_df\_train

history2 = exval\_df\_train\_reduced

predictions3 = list()

for t in endog\_df\_test.index:

model = ARIMA(endog = history1, exog = history2, order=(2,1,6))

model\_fit = model.fit()

output = model\_fit.forecast(steps= 1, exog = exval\_df\_test\_reduced[t:t],dynamic=False)

output = output.to\_frame()

yhat = output.iloc[0,0]

predictions3.append(yhat)

obs1 = endog\_df\_test[t:t]

history1 = history1.append(obs1)

obs2 = exval\_df\_test\_reduced[t:t]

history2 = history2.append(obs2)

# %%

from matplotlib.pyplot import figure

figure(figsize=(8, 5), dpi=80)

plt.plot(endog\_df\_test.index, endog\_df\_test.values, label = 'Actual', marker='D')

plt.plot(endog\_df\_test.index, predictions3, label = 'Predicted', color='red', marker='D')

plt.ylabel('Close price (USD)', fontsize = 12)

plt.xlabel('Date', fontsize = 12)

plt.legend(loc='best')

plt.show()

# %%

from sklearn.metrics import mean\_squared\_error

import math

forecast1 = arima\_fit.get\_prediction(start= 10, dynamic=False)

pred1 = forecast1.conf\_int()

pred1['mean']=forecast1.predicted\_mean

pred1 = pred1.join(price\_df\_train)

pred1 = pred1.dropna()

forecast2 = arimax\_fit.get\_prediction(start= 10, dynamic=False)

pred2 = forecast2.conf\_int()

pred2['mean']=forecast2.predicted\_mean

pred2 = pred2.join(price\_df\_train)

pred2 = pred2.dropna()

forecast3 = arimax1\_fit.get\_prediction(start= 10, dynamic=False)

pred3 = forecast3.conf\_int()

pred3['mean']=forecast3.predicted\_mean

pred3 = pred3.join(price\_df\_train)

pred3 = pred3.dropna()

print('ARIMA model MSE\_Train:{}'.format(mean\_squared\_error(pred1['Close'], pred1['mean'])))

print('ARIMA model MSE\_Train:{}'.format(mean\_squared\_error(pred2['Close'], pred2['mean'])))

print('ARIMAX model top 2 features MSE\_Train:{}'.format(mean\_squared\_error(pred3['Close'], pred3['mean'])))

print('ARIMA model RMSE\_Train:{}'.format(math.sqrt(mean\_squared\_error(pred1['Close'], pred1['mean']))))

print('ARIMA model RMSE\_Train:{}'.format(math.sqrt(mean\_squared\_error(pred2['Close'], pred2['mean']))))

print('ARIMAX model top 2 features RMSE\_Train:{}'.format(math.sqrt(mean\_squared\_error(pred3['Close'], pred3['mean']))))

# %%

import math

endog\_df\_test['dif']= endog\_df\_test.Close.diff()

nulldist\_df = endog\_df\_test.dropna()

mu = sum(nulldist\_df['dif'])/len(nulldist\_df['dif'])

sig = math.sqrt(sum((nulldist\_df['dif']-mu)\*\*2)/len(nulldist\_df['dif']))

np.random.seed(1001)

nulldist\_df['predicted\_dif'] = np.random.normal(mu, sig, len(nulldist\_df['dif']))

nulldist\_df['predicted\_Close'] = 1.01

for i in range(0, len(nulldist\_df['dif'])-1):

nulldist\_df['predicted\_Close'][i+1] = nulldist\_df['Close'][i]+nulldist\_df['predicted\_dif'][i+1]

nulldist\_df = nulldist\_df[1:]

# %%

from sklearn.metrics import mean\_squared\_error

print('ARIMA model MSE\_Test:{}'.format(mean\_squared\_error(endog\_df\_test.Close, predictions1)))

print('ARIMAX model MSE\_Test:{}'.format(mean\_squared\_error(endog\_df\_test.Close, predictions2)))

print('ARIMAX model top 2 features MSE\_Test:{}'.format(mean\_squared\_error(endog\_df\_test.Close, predictions3)))

print('Null model MSE\_Test:{}'.format(mean\_squared\_error(nulldist\_df['Close'], nulldist\_df['predicted\_Close'])))

# %%

print('ARIMA model RMSE\_Test:{}'.format(math.sqrt(mean\_squared\_error(endog\_df\_test.Close, predictions1))))

print('ARIMAX model RMSE\_Test:{}'.format(math.sqrt(mean\_squared\_error(endog\_df\_test.Close, predictions2))))

print('ARIMAX model top 2 features RMSE\_Test:{}'.format(math.sqrt(mean\_squared\_error(endog\_df\_test.Close, predictions3))))

print('Null model RMSE\_Test:{}'.format(math.sqrt(mean\_squared\_error(nulldist\_df['Close'], nulldist\_df['predicted\_Close']))))

# %%

from matplotlib.pyplot import figure

figure(figsize=(8, 5), dpi=80)

plt.plot(nulldist\_df.index, nulldist\_df['Close'], label = 'Actual', marker='D')

plt.plot(nulldist\_df.index, nulldist\_df['predicted\_Close'], label = 'Predicted',color='red', marker='D')

plt.ylabel('Close price (USD)', fontsize = 12)

plt.xlabel('Date', fontsize = 12)

plt.legend(loc='best')

plt.show()

# %%

import scipy

slope, intercept, r, p, stderr = scipy.stats.linregress(endog\_df\_test.Close, predictions1)

line = f'Regression: y={intercept:.2f}+{slope:.2f}x, r={r:.3f}'

plt.plot(endog\_df\_test.Close, predictions1, label = 'Predicted',color='blue', marker='s',linewidth=0)

plt.plot(endog\_df\_test.Close, intercept + slope \*endog\_df\_test.Close,label=line)

plt.ylabel('Predicted', fontsize = 12)

plt.xlabel('Actual', fontsize = 12)

plt.legend(loc='best')

plt.show()

# %%

slope, intercept, r, p, stderr = scipy.stats.linregress(endog\_df\_test.Close, predictions2)

line = f'Regression: y={intercept:.2f}+{slope:.2f}x, r={r:.3f}'

plt.plot(endog\_df\_test.Close, predictions2, label = 'Predicted',color='blue', marker='s',linewidth=0)

plt.plot(endog\_df\_test.Close, intercept + slope \*endog\_df\_test.Close,label=line)

plt.ylabel('Predicted', fontsize = 12)

plt.xlabel('Actual', fontsize = 12)

plt.legend(loc='best')

plt.show()

# %%

slope, intercept, r, p, stderr = scipy.stats.linregress(endog\_df\_test.Close, predictions3)

line = f'Regression: y={intercept:.2f}+{slope:.2f}x, r={r:.3f}'

plt.plot(endog\_df\_test.Close, predictions3, label = 'Predicted',color='blue', marker='s',linewidth=0)

plt.plot(endog\_df\_test.Close, intercept + slope \*endog\_df\_test.Close,label=line)

plt.ylabel('Predicted', fontsize = 12)

plt.xlabel('Actual', fontsize = 12)

plt.legend(loc='best')

plt.show()

# %%

slope, intercept, r, p, stderr = scipy.stats.linregress(nulldist\_df['Close'], nulldist\_df['predicted\_Close'])

line = f'Regression: y={intercept:.2f}+{slope:.2f}x, r={r:.3f}'

plt.plot(nulldist\_df['Close'], nulldist\_df['predicted\_Close'], label = 'Predicted',color='blue', marker='s',linewidth=0)

plt.plot(nulldist\_df['Close'], intercept + slope \*nulldist\_df['Close'],label=line)

plt.ylabel('Predicted', fontsize = 12)

plt.xlabel('Actual', fontsize = 12)

plt.legend(loc='best')

plt.show()

# %%