DS8004 Data Mining Project Investment Fund Analytics (April 2017)

Arabi, Aliasghar (#500606766) – Najlis, Bernardo (#500744793)

***Abstract*—This project aimed to create a model that finds potential correlations between world news events and stock market movements. The data set used for news is based on the Reddit /r/worldnews channel, and DJIA daily values for stock markets, using KNIME and R with Decision Trees and SVM (Support Vector Machines) models. Initial data pre processing was done using Hadoop and Hive in a cloud cluster environment using Azure HDInsights. The models obtained did not achieve an acceptable accuracy level, and further data exploration revealed how this experiment can be attempted with other datasets to be improved. The project was completed as part of the DS8004 Data Mining Course at the Masters in Data Science Program at Ryerson University in Toronto, Ontario during the months of January through April 2017.**

***Index Terms*— Fund Analytics, Text Analytics, Sentiment Analysis, Social Media Analytics, Data Mining, Knime, SVM, Decision Trees, Classification, Reddit, DJIA, Dow Jones**

I. Case Description and Problem Presentation

INVESTMENT funds play a central role in the free market economy, as they provide capital from individual and corporate investors to collectively purchase securities while each investor retains ownership of their shares. Therefore, understanding the market and optimizing investment strategies is one of their key activities. One of the major influencers in markets is news, so news analytics is typically used by investment funds to make decisions on their fund allocations.

Funds use news analytics to predict [1]:

- Stock price movements
- Volatility
- Trade Volume

Using text and sentiment analytics, it is possible to build models that try to predict market behavior and anticipate to them. Applications and strategies for this can be:

- Absolute Return Strategies
- Relative Return Strategies
- Financial Risk Management
- Algorithmic Order Execution

This project aims to build a model around news analytics to anticipate DJIA (Dow Jones Industrial Average) trends (up or down) by looking at the correlation between world news events and this major stock market index using text analytics.

The DJIA index is an average calculation that shows how 30 large publicly owned companies based in the United States have traded during a standard trading session in the stock market. The index has compensation calculations for stock events such as stock splits or stock dividends to generate a consistent value across time. [4] Historical DJIA prices can be obtained from many online sources, this project will use the Wall Street Journal's public website [3] as the source.

For the world news data source we aim to use is the **/r/worldnews** Reddit channel [2]. Reddit is a major social news aggregation and discussion website, where members can submit content and also vote these submissions up or down to determine their importance.

The scope of our dataset both for news and stock index values will range from January 2008 through December 2016. Multiple tools were considered and we decided to use Knime, as it is an open source data analytics and integration platform. Knime provides a graphical user interface and can also be augmented with the use of R language scripts.

II. Proposed Methodology as Data Mining Project

The methodology we propose follows these steps:

1. Connect to the Reddit public provided API (Application's Programmers Interface) and download all news headlines from the /r/worldnews. As usually all online services impose data download limitations on APIs, we may resort to accessing some of the publicly available reddit dataset dumps online.
2. Find the top 25 news headlines sorted by upvotes

for each day. If a particular date has less than 25 news, we will only use those.

3. Perform text analytics techniques on the news headlines obtained. These will contain one or several of these common algorithms:
   - Tokenization
   - Stop word removal
   - Stemming
   - Sentiment Detection and classification
4. Download DJIA daily historical data from WSJ or other online sources available.
5. Label the daily news headlines based on the index value move for each day, comparing the open and close value: 0 for days when the close price was equal or less than the open price, 1 for days when the close price was higher than the open price.
6. Train multiple models with the combined outputs of 3 and 5.
7. Test and tune the model using test set.

III. Data Acquisition and Engineering

The data acquisition and engineering process is composed of several steps from the raw data sources obtained online to the usable, filtered and formatted required by our project.

A. *Methodology*

As anticipated, we found that the Reddit API limitations did not permit us to get the data volumes required. The Reddit API allows a maximum of 60 requests per minute that can return a maximum of 100 objects per call. The complete dataset for this project is around 1.7 billion objects so it would require around 231 days of continuous downloads to obtain the full dataset.

We decided to use a publicly available Reddit data dump provided online. [7] The caveat with this data dump is that contains data from all Reddit channels, so it will require more data processing than originally anticipated.

The complete dataset with all submissions for all 10 year and all subreddit channels is 74.1 GB compressed. To download this data we used an Ubuntu virtual machine in the Azure public cloud and series of bash scripts. Data is split on a monthly package in compressed .tar.bz2 files, so after each file is downloaded locally in this VM, we uploaded it to an Azure Blob Storage container.

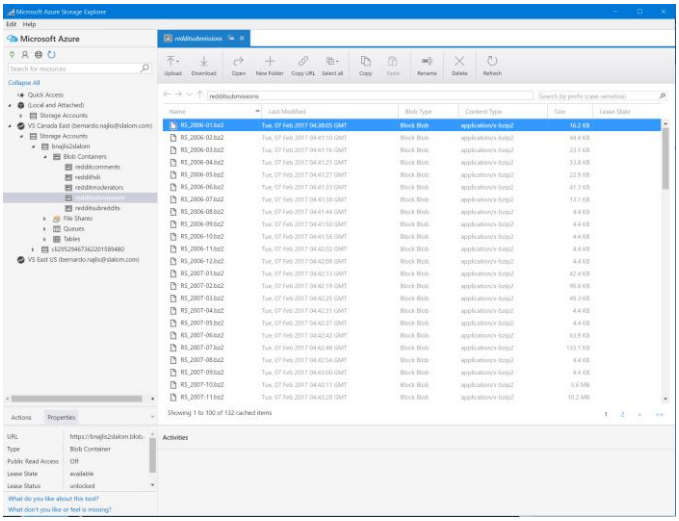


Fig. 1. Azure Blob Storage Container with all files downloaded from public online sources. This service allows to store virtually unlimited number of files to be later used with other data processing services in the Azure Cloud.

B. *Raw Format*

The WSJ DJIA data format is very simple and easily accessed, and the schema is fairly simple:

Field Name	Description
Date	Date in mm/dd/yy format (i.e: 03/21/2016 for March 21st, 2016)
Open	Value for the DJIA at market open, in points
High	Highest value for the day, in points
Low	Lowest value for the day, in points
Close	Value for the DJIA at market close, in points

Table 1. Schema for the DJIA dataset obtained from the WSJ website.

For the Reddit dataset, each of the obtained the data in monthly individual .tar.bz2 packages contains a large text file with all reddit submissions for that month. Each line contains one submission in JSON format, as seen in Figure 2.

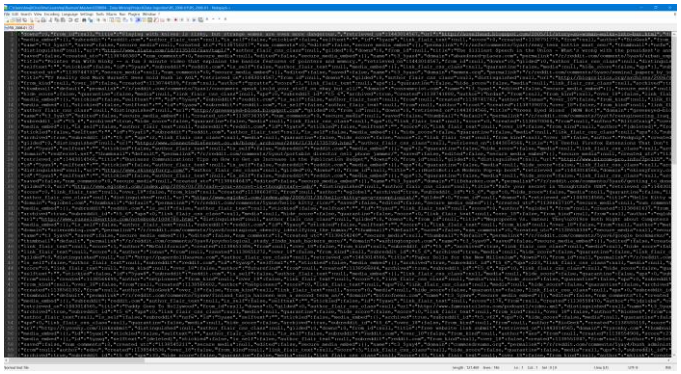
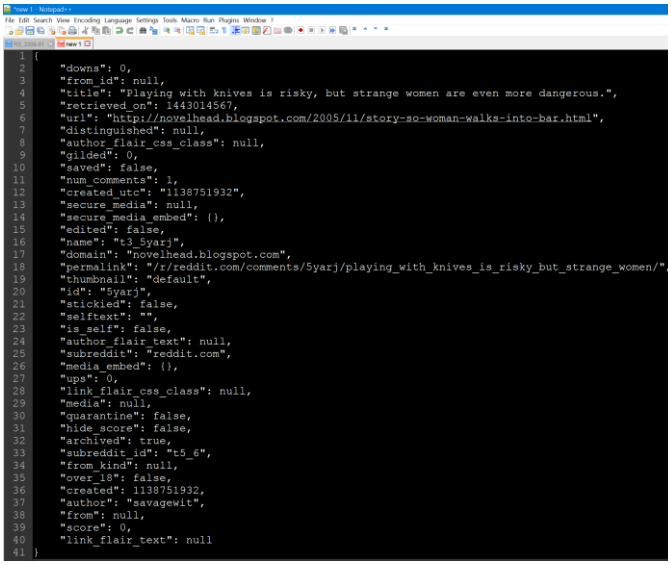


Fig. 2. Screen capture of a sample monthly Reddit submissions data dump. The file contains multiple lines with all submissions in JSON format.

By further exploring one of these submissions, we determined we only need a subset of all the fields available in each JSON document. As the data volume for the uncompressed data set is very large, we decided to use Azure HDInsight as the data processing engine to filter out data from all channels other than /r/worldnews and to keep only the fields

required for our project:

- subreddit = "worldnews"
- title (news headline)
- created_utc (date_time) and year
- score, up, down (upvotes, and downvotes)



```
{
    "downs": 0,
    "from_id": null,
    "title": "Playing with knives is risky, but strange women are even more dangerous.",
    "retrieved_on": 1443014567,
    "url": "http://novelhead.blogspot.com/2005/11/story-so-woman-walks-into-bar.html",
    "distinguished": null,
    "author_flair_css_class": null,
    "gilded": 0,
    "saved": false,
    "num_comments": 1,
    "created_utc": "1138751932",
    "secure_media": null,
    "secure_media_embed": {},
    "edited": false,
    "name": "t3_5yarj",
    "domain": "novelhead.blogspot.com",
    "permalink": "/r/reddit.com/comments/5yarj/playing_with_knives_is_risky_but_strange_women/",
    "thumbnail": "default",
    "id": "5yarj",
    "stickied": false,
    "selftext": "",
    "is_self": false,
    "author_flair_text": null,
    "subreddit": "reddit.com",
    "media_embed": {},
    "ups": 0,
    "link_flair_css_class": null,
    "media": null,
    "quarantine": false,
    "hide_score": false,
    "archived": true,
    "subreddit_id": "t5_6",
    "from_kind": null,
    "over_18": false,
    "created": 1138751932,
    "author": "savagewit",
    "from": null,
    "score": 0,
    "link_flair_text": null
}
```

Fig. 3. Screen capture of a single submission in JSON format. Our project only required a subset of all these fields.

C. *Processing in HDInsight Hadoop Cluster*

Using Hadoop and Hive to process big data volumes is a common technique that also gets simplified by the access to public cloud services like Azure. Azure HDInsight [8] is a managed Hadoop service that lets you create a fully configured cluster by using a web UI to select just a couple of options (like cluster size and nodes capacity) and is charged by the hour. For the purposes of this project, we created a small cluster during the development phase while experimenting with Hive queries.

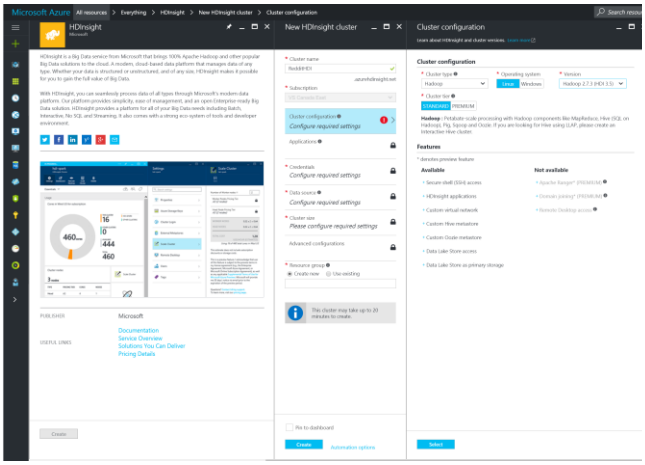


Fig. 4. Azure HDInsight web user interface for cluster creation. Options allow to create a cluster on Linux or Windows, select different Hadoop versions, integrate with other Azure services (like Blob Storage), number of and node sizes.

The final cluster used to run the processing queries had the following characteristics:

Type	Node Size	Cores	Nodes	Specs
Head	D12 v2	8	2	2.4 GHz Intel Xeon E5-2673 v2 (Haswell) processor, with Intel TurboBoost 2.0 can go up to 3.1 GHz.
Worker	D12 v2	16	4	

Table 2. HDInsight Hadoop cluster specifications used to pre-process the raw data into a subset that can be used with data analytics tools (Knime).

D. *Processing in Hive*

Hive is a data processing service part of the Hadoop ecosystem which "translates" HQL (Hive Query Language) queries into Map Reduce programs. HQL is very similar to SQL (Structured Query Language) that our team is very familiar with. Hive can process data in two ways: by building a "virtual" schema on top of existing files in HDFS (Hadoop Distributed File System) or by importing data into "tables" managed by it. One of the advantages of HDInsight is that it can access files stored in an Azure Blob Storage container as an HDFS file system transparently.

```
CREATE DATABASE IF NOT EXISTS reddit;

USE reddit;

CREATE EXTERNAL TABLE IF NOT EXISTS reddit.submissions_raw
(
  value STRING
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
LOCATION 'wasb://subtest@bnajlis2slalom.blob.core.windows.net/';
```

Fig. 5. HQL queries to create a Hive schema and external tables to access raw data stored in Azure Blob Storage through HDFS URLs.

Generally, processing data in large JSON text, compressed data sets is very time-consuming, so we decided to pre-filter the data into ORC tables first. ORC is a highly compressed, binary columnar data store and is much more efficient than compressed .tar.bz2 JSON files for fast data processing.

```
INSERT INTO reddit.submissions_orc(submission_year, subreddit, submission_date, title, score, ups, downs)
SELECT
    CAST(from_unixtime(CAST(v.created_utc AS BIGINT), 'yyyy') AS SMALLINT) as submission_year,
    v.subreddit,
    from_unixtime(CAST(v.created_utc AS BIGINT), 'yyyy-MM-dd') as submission_date,
    v.title,
    v.score,
    v.ups,
    v.downs
FROM reddit.submissions_raw jt
    LATERAL VIEW json_tuple(jt.value, 'subreddit', 'created_utc', 'title', 'score', 'ups', 'downs') v
    AS subreddit, created_utc, title, score, ups, downs
WHERE v.subreddit = "worldnews";
```

Fig. 6. HQL queries to insert only required columns (submission_year, subreddit, submission_date, title, score, ups, downs) from "worldnews" subreddit into an ORC table.

Once the data is in the Hive managed ORC tables we export these into CSV files that can be processed with other data analytics tools (namely, Knime).

```
INSERT OVERWRITE DIRECTORY
'/user/admin/reddit_worldnews'
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (   "separatorChar" = " ",
                         "quoteChar"   = "'")
SELECT
    submission_date,
    regexp_replace(title, "\n|\t", "") as title,
    score
FROM reddit.submissions_orc;
```

Fig. 7. HQL queries to export data from ORC tables into CSV format.

As the ORC tables created were clustered into 10 buckets, we obtained a set of CSV files, that then we will have to union to obtain the complete dataset. Once the Big Data processing was complete, we downloaded the final dataset into our local laptops and shutdown the HDInsight cluster.

Field Name	Description
submission_year	Year of submission, extracted from the submission_date field
subreddit	Name of the subreddit forum, for the case of this project all will be "worldnews"
submission_date	UTC date-time for the submission
title	News headline
score	Score calculated by Reddit that correlates to the number of upvotes and downvotes for each submission
ups	Number of upvotes for the reddit submission
downs	Number of downvotes for the reddit submission

Table 3. Schema for the CSV files exported using Hive into CSV format.

E. *Ingestion into Knime*

Ingestion of data into Knime is done using the CSV Reader component. Multiple components read each file exported from Hive and combined using the "Concatenate" component. This process is the equivalent of a "UNION ALL" in SQL language: all rows are combined into one large dataset.

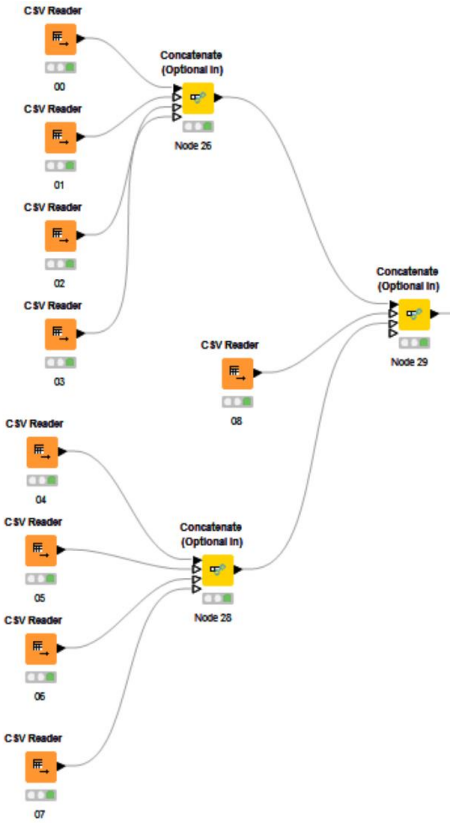


Fig. 8. Ingestion of independent CSV files extracted from Hadoop into Knime. We combined all files into one large dataset with all concatenated rows.

Additional steps were performed in the Knime workflow to further refine the data from the CSV files:

- **Sort**: order all news by date and upvote
- **RowID**: Add a numeric unique key to identify each row
- **Column Rename**: Add names to columns per schema in Table 2
- **Rank**: Assign an order number based on the score, for news for each day
- **Row Filter**: keep only top x news ranked per date
- **Sort**: Re sort rows by date and rank. This will improve the performance when joining with the DJIA dataset.
- **String** to Date/Time: Format news headlines date-time string value as the native date/time datatype from Knime

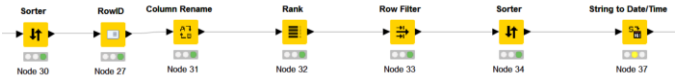


Fig. 9. Additional steps for ingestion: sorting, ranking, filtering and data type conversions.

One final step in order to get the data ready to do any type of analytics is to combine it with the DJIA WSJ dataset. The following steps were done:

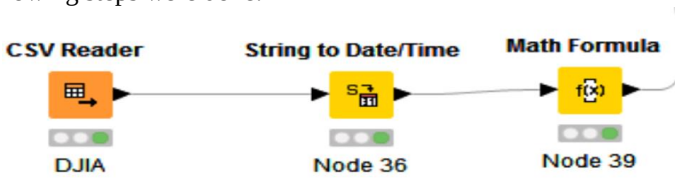


Fig. 10. Loading and preprocessing DJIA data from WSJ.

- CSV Reader: read the CSV raw data with schema from Table 1.
- String to Date/Time: Convert the text provided data into native Knime date-time datatype.
- Math Formula: Label the market results for the date, using the formula from Fig. 11

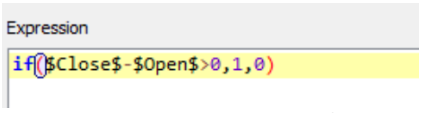


Fig. 11. Formula used to label each trading date for days with gains and losses.

After these was done, both datasets were joined based on the date field.

IV. Descriptive Analytics on the Data

A. *DJIA*

The Dow Jones Index values (as seen in the below chart) account for a total of 2,265 days (from 2008-01-02 to 2016-12-30) ranging from 6,547.05 points to 19,974.62 points with the lowest close price recorded on 2009-03-09 and the highest on 2016-12-20. It is worth knowing that the index values are scored in points that compose of 30 different stocks, and its composition gets changed and over time by S&P, who created and owns the index.

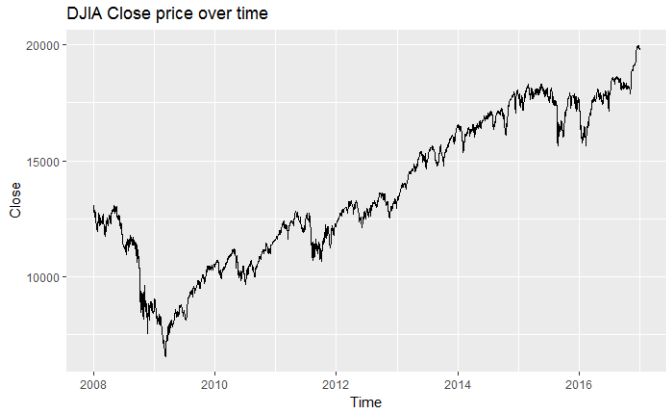


Fig. 12. DJIA Close price over time.

We also looked at the close price spread per year (figure 13) and per month (figure 14). The close price spreads are almost similar for all years but 2008 and 2009 when stock market crashed.

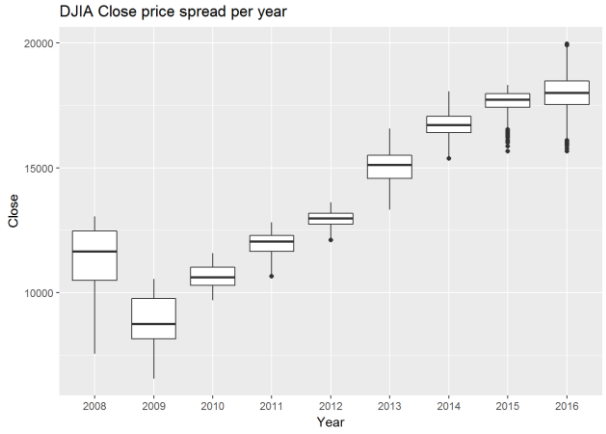


Fig. 13. DJIA Close price spread per year

Same observation holds for spreads of the close price for different months with no major discrepancy across different months.

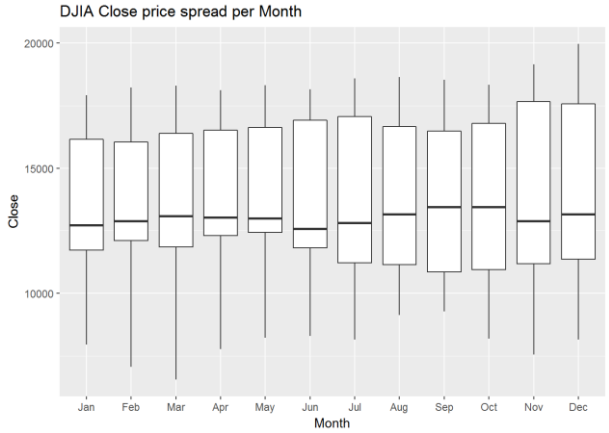


Fig. 14. DJIA Close price spread per month

The following chart in Figure 15 shows the proportion of up and down days by year and by month. It is clear that up and down days are about 50% split in both year and month.

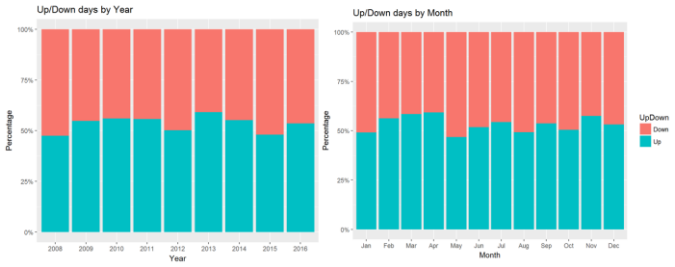


Fig. 15. Up/Down days per year(left) and per month(right)

B. *Reddit /r/worldnews*

The complete Reddit /r/worldnews dataset has 2,018,344 headlines. We decided to limit our dataset to a maximum of 10 headlines per day based the upvoting score. Headlines range from 2008-01-25 to 2016-12-31, with scores (upvotes) varying from 0 to 93,832. The chart below shows upvotes spread per year with year 2016 with the highest number of outliers which can be explained by higher number of news headline in the channel on Reddit.

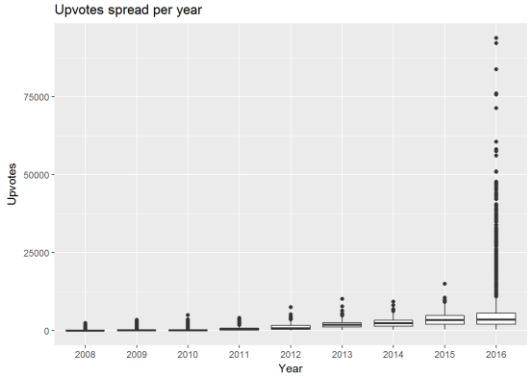


Fig. 16. Upvotes spread per year

The highest scored headline on 2016-11-26 is: “Fidel Castro is dead at 90.”

Also, here is a sample 0-scored headlines:

- "Avalanche Kills TV Star Christopher Allport"
- "Immunizations"
- "WHO to recommend ways to reduce harm of alcohol
- "Nicolas Sarkozy and Carla Bruni marry"

The word cloud of the unigrams in headlines (figue17) visually shows the most frequent terms. Please note that for this visualization (done in R) we did not do any stemming or stop word removal to show all the terms in their complete forms.

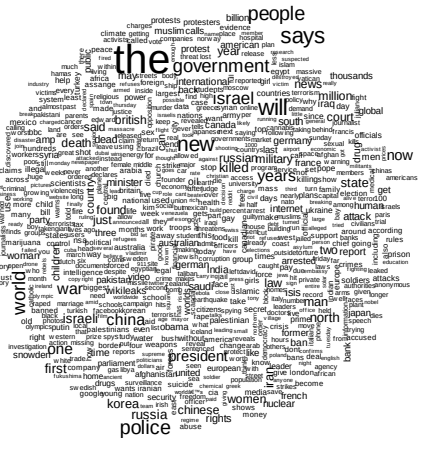


Fig. 17. News headlines unigram word cloud

From the word cloud one can see that most frequent terms are name of different countries such as: “Russia”, “Israel”, “Iran”, and “China” to name a few as well as other terms such as: “war”, “kill”, “protest”, “people”, and “police”.

Finally, after performing dictionary based sentiment analysis scoring of the headlines, we looked at the distribution of sentiment score of the headlines per weekdays (figure 18) and we did not observe any significantly different distribution over the days.

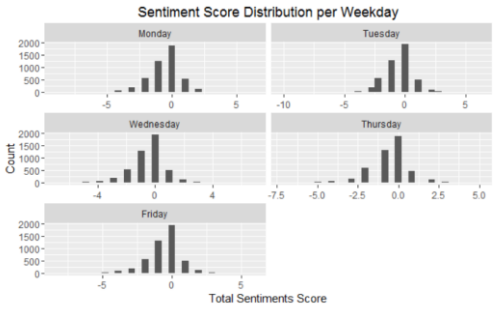


Fig. 18. Sentiment score distribution per weekday

What we found interesting about the sentiment score of the news headline is that they are slightly skewed to the left (negative scores) meaning that there is a slightly more number of negatively scored headlines than positively scored headlines.

V. Predictive Models Description and Results

A. *Data preprocessing and text analytics in KNIME*

As part of the preprocessing, we removed punctuations, stop words, words less than a certain number of characters (configurable in the knime workflow, shown in Figure 20), performed stemming and finally converted all terms to lower case. We then created bag of words of unigrams, and bi-grams.

As the number of features generated by both unigrams and bigrams bag of words is considerably large, we decided to evaluate a separate method: using just headline’s positive/negative sentiment scores with their upvotes as features as the only features, in an attempt to reduce the number of dimensions. The reason why we did not perform any traditional dimensionality reduction techniques such as SVD or PCA is because we think the presence of all unigram/bigram as features are crucial, and omitting any of them will affect the performance of the model. Figure 19 below shows the KNIME workflow for data preprocessing:

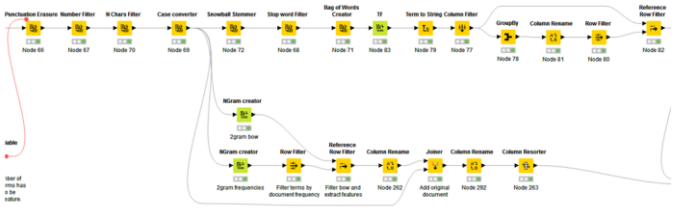


Fig. 19. Sentiment score distribution per weekday

We also implemented term-frequency scores in a way that for a unigram/bigram term to be selected as feature it should have occurred in minimum number of documents (Figure 20).

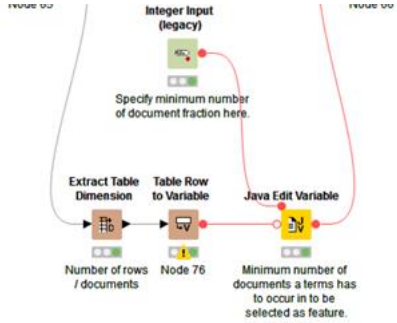


Fig. 20. Feature selection condition

For sentiment analysis preprocessing, we used a dictionary based approach and leveraged the “R Snippet” component to assign a positive and negative score to each headline (Figure 21). Due to R memory management limitations, we split the complete dataset into smaller sets and then calculate the sentiment scores. And at the end, everything is concatenated into a one single dataset.

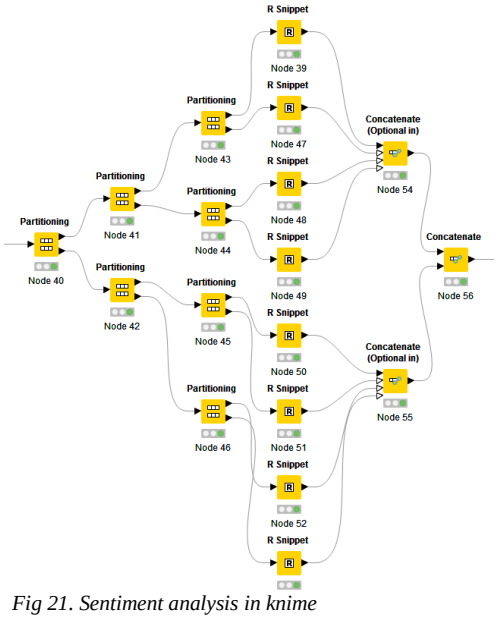


Fig 21. Sentiment analysis in knime

B. *Decision Trees and SVM*

For class prediction we used two traditional machine learning techniques for two-class classification problems: decision trees and SVM. As seen in figure 22, to prepare for training the data, we created a document vector with news as rows and unigrams/bigrams as features. Each table cell may contain either 0 or 1 with 1 indicating that that specific term exists in the headline. For the training/test split set creation, 70% of the available data is selected randomly as the training and the remaining 30% as test set. Decision trees and SVM models are trained using the training set and then the test set is used to make the predictions using the trained model.

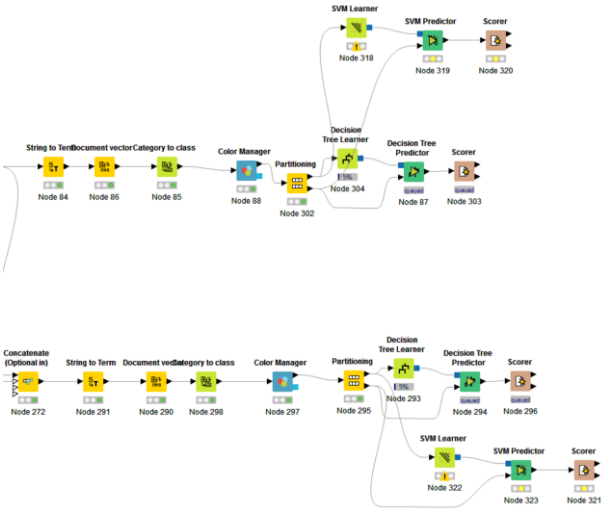


Fig 23. Document vector and decision tree / SVM model in Knime

Figure 24 shows a snapshot for a typical machine learning workflow in Knime (decision tree as example): training/test set partitioning, model training, prediction and finally scoring the model with a confusion matrix.

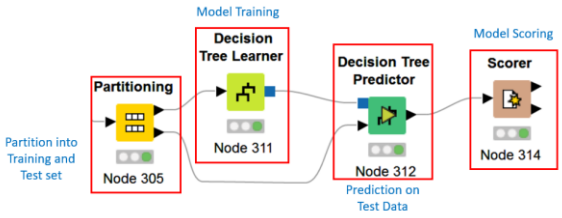


Fig 24. Typical machine learning workflow in Knime

First, the "Partitioning" node is configured to split the data into training and test sets (Figure 25) and the "Decision Tree Learner" node will be trained using just the training set. Its result is fed into a "Decision Tree Predicter" node which will predict classes of the test set using the previously learned model.

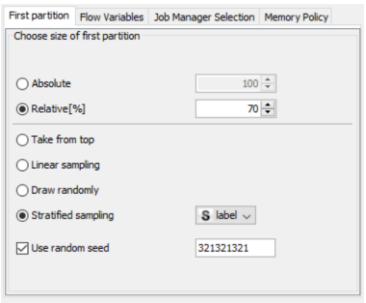


Fig. 25. Partitioning data into training and test sets

Finally, the "Scorer" node is added as the workflow's final step to obtain the confusion matrix and other statistics. (Figure 26).

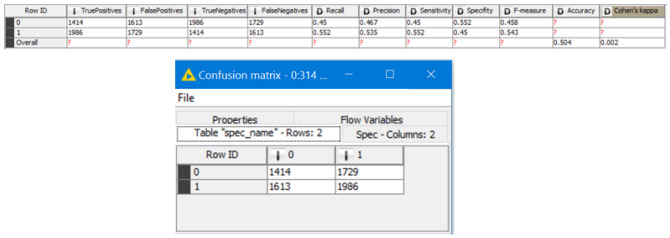


Fig 26. Confusion matrix and prediction statistics

We used Top 1, Top 3, Top 5, and Top 10 upvote headlines to train different models and summarized their accuracies in Table 4. Based on the achieved accuracy of all the models we observed very minor improvements using SVM as opposite to decision trees, but in general the accuracy of all the models is not good; across all models the maximum accuracy value obtained was 0.517 and the average 0.50235.

Decision Trees	Uni-gram	Bi-gram	Sentiment Analysis
Top 1	0.503	0.492	0.487
Top 3	0.493	0.5	0.506
Top 5	0.503	0.509	0.503
Top 10	0.502	0.501	0.504

SVM	Uni-gram	Bi-gram
Top 1	0.517	0.495
Top 3	0.511	0.507
Top 5	0.509	0.501
Top 10	0.507	0.497

Table 4. model comparison - Accuracy

VI. Conclusions, Lessons Learned and Next Steps

A. *Conclusions*

The results achieved are not promising, and discouraged us from further attempting to improve or tweak the modelling, as we believe there is a fundamental failure with the premise of the project, as the news dataset chosen is not the appropriate to prove or disprove this type of experiment.

Further exploratory data analysis was performed post-facto, to analyze the results of the project and find an explanation to the low accuracy numbers. Figure 26 shows the class distribution for the top frequent unigrams. The classes are very well balanced, and there is no plausible correlation between

unigrams (words used) and market moves up or down. This also reaffirms the perception that the dataset was not the correct one in order to solve this problem.

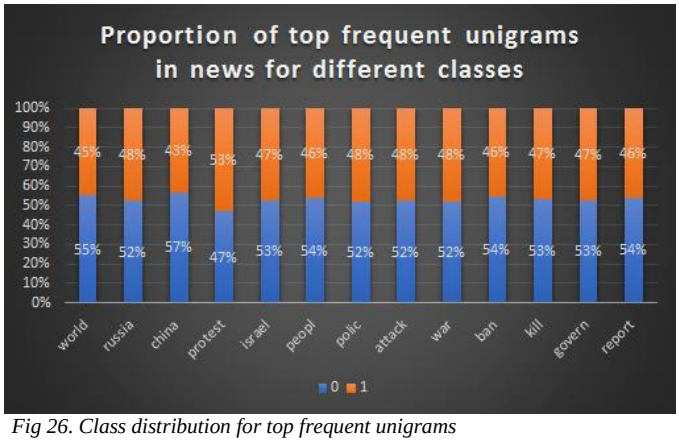


Fig 26. Class distribution for top frequent unigrams

Over the next three main points we discuss the reasons why we believe the results obtained are not as expected:

1. *Markets efficiency and wrong data*: The DJIA market data used was not at the correct level of granularity (daily open and daily close) to be able to observe potential impacts of the news headlines in the market. Markets react to headline news in time periods much shorter than days.
   Also, "world news" is a very broad subject in order to try to look for a correlation with stock markets, as it was shown by a very basic exploratory data analysis of the most upvoted and some random 0-scored news headlines.
2. *More data does not equal to better results*: Using stock market information from 2008 to predict market movements in 2016 is just too broad: markets behavior go through phases and economic realities change through time.
3. *Realistic expectations about data analytics projects*: One of the most critical assets required for a data analytics project to be successful is access to the correct dataset, which this project failed to guarantee. Even though further improvements and modelling refinements would have been possible, time restrictions made it impossible to continue with the research, as every iteration or changes in the model take copious amounts of time.

B. Lessons Learned

Both team members put a great amount of effort in learning new tools and techniques to complete this project. Here is a summary of the personal learnings shared across the team:

- **Data Mining and Analytics**

1. *Data Science Problem Framing*: We proposed a hypothesis (markets are affected by world news) and found a way to use publicly available data to create a series of experiments that use machine learning and data mining in order to prove / disprove it.
2. *Bag of Words vs. N-grams*: Used both extracted set of features in order to test different possible outcomes of models.
3. *Pseudo TF-IDF*: By using term frequency for both unigrams and bigrams, we added additional features to help improve the final model precision.
4. *Sentiment analysis*: We implemented a dictionary based approach using R to score all headlines and use this in an attempt to avoid traditional dimensionality reduction.
5. *Decision Trees and SVM:* Experimented with the differences and limitations of these two machine learning models.
6. *Validated the premise that "All data science work is 80% to 90% cleaning and preparation and just about 10 to 20% modelling and machine learning related".* Just by doing a visual inspection of the Knime model created, which is composed of 103 nodes, only 15 nodes (14.5% of the total) are for modelling; the rest of the 88 nodes (85.5% of the total) are used for data preparation cleaning, processing, manipulation and feature extraction.

- **Technical Tools**

1. *Azure HDInsight*: All big data processing was done in Hadoop clusters created in the cloud, using Hive to just keep the subset of reddit posts used in the project. Cloud-based computing provided very easy access for data processing at a very accessible cost.
2. *Knime and Knime-R integration:* Using knime allowed the team to use more data processing and modelling techniques and experiment alternatives than if this project were to be done with a data science language (R or Python). Having modules with pre-created logic that are assembled and combined also allows to create a single package that contains all data transformations and permits reproducibility and visual understanding, with not much further documentation required.
3. *Ggplot2 for exploratory data analysis and descriptive analytics*: One of the most popular graphic packages in R, allowed the team to create a series of charts based to explore the original data set.
4. *Memory management limitations in R*. We were surprised by the multiple number of issues generated by the GNU version of R, that required to split the complete dataset to run simple scripts, due to its memory management limitations. Further exploration may lead to using Revolution Analytics R to try to overcome these, but were not performed due to time limitations.

C. Next Steps

Due to the time limitations imposed by the nature of this course-related experiment, the team could not further improve or continue the work on several ideas or approaches they feel would be valuable to try out.

1. *Change data granularity:* As explained in the conclusions section, using a different dataset should be attempted to finally prove / disprove the initial premise and hypothesis of this project. Using streaming or hourly / by minute market data is necessary to capture the quick moving nature of stock markets to news. Further experiments should be designed using more geographically localized news (as DJIA is mostly composed on United States' companies), or by industrial sector or specific companies, to try to correlate also more specific stock instruments like sector ETFs or individual companies' news.
2. *Use Deep Neural Networks:* This machine learning method is very promising in the presence of large data sets like the one used in this project. Integration with Knime is very incipient but it can still be achieved by using the R or Python code snippets.
3. *Use Hidden Markov Models:* Another machine learning type of model that can be used for further improvements, as it is specific for time series datasets like stock market price changes.
4. *Create models on a per-year basis:* As explained in the conclusions section, markets change their overall behavior and cycle through time. By creating models that are time-bound, higher values of accuracy should be achieved.

REFERENCES

[1] "News Analytics", Wikipedia: https://en.m.wikipedia.org/wiki/News_analytics

[2] "Reddit – World News": https://www.reddit.com/r/worldnews/

[3] "Wall Street Journal - Dow Jones Industrial Average": http://quotes.wsj.com/index/DJIA

[4] "Dow Jones Industrial Average", Wikipedia: https://en.m.wikipedia.org/wiki/Dow_Jones_Industrial_Average

[5] Knime: http://www.knime.org

[6] "Reddit API Documentation": http://www.reddit.com/dev/api/ "PushShift.io – Reddit Data Dump" http://files.pushshift.io/reddit/ "Azure HDInsight": https://azure.microsoft.com/en-us/services/hdinsight

[7] "Sizes for Cloud Services" – Microsoft Azure Public Website: https://docs.microsoft.com/en-us/azure/cloud-services/cloud-services-sizes-specs#dv2-series

Aliasghar Arabi (Masters of Data Science and Analytics, Ryerson University 2018)
His background is as Industrial Engineer from Iran and MBA from South Korea. Currently, he works at Economical Insurance as a Business Analytics Professional. Here, he is responsible to develop statistical models based on socio-demographic, financial and customer data in support of improvement to products, operational efficiency, and the customer experience.

Bernardo Najlis (Masters of Data Science and Analytics, Ryerson University 2018) was born in Buenos Aires, Argentina in 1977. Received a BS in Systems Analysis from Universidad CAECE in Buenos Aires, Argentina in 2007, and is candidate for Masters of Data Science and Analytics at Ryerson University in Toronto, Canada.
He currently is a Solutions Architect in the Information Management & Analytics team at Slalom, a worldwide consulting firm, working in Big Data and Advanced Analytics projects.

APPENDIX A – KNIME WORKFLOWS

Fig A-1: Initial data loading and pre-processing to create a complete dataset.

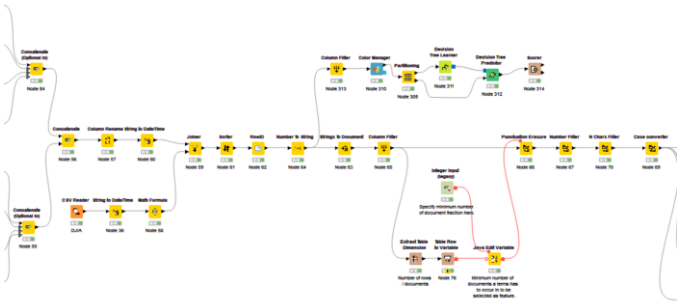


Fig A-2:Text analytics and modelling with sentiment analysis scores.

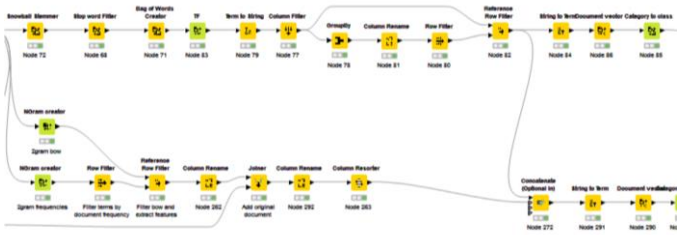
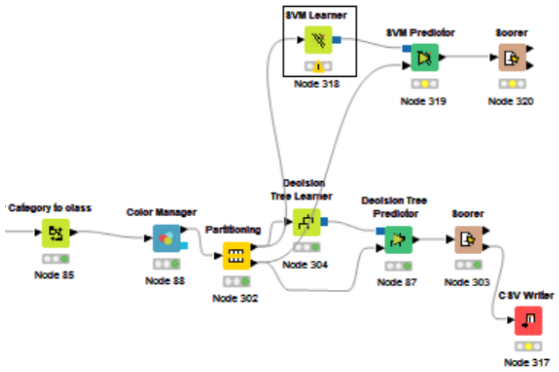


Fig A-3: Further text analytics, bag of words and preprocessing for final SVM and decision trees model creation.



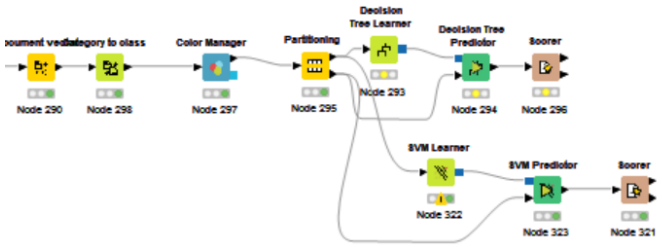


Fig A-4: Machine learning models for Decision Trees and SVM

APPENDIX B – DICTIONARY BASED SENTIMENT ANALYSIS IN R

```
#loading positive words
pos.dic   <-   scan('positive-words.txt',   what   =
'character', comment.char = ';')

#loading negative words
neg.dic   <-   scan('negative-words.txt',   what   =
'character', comment.char = ';')

pos.score.sentiment = function(sentences, pos.words)
{
  require(plyr);
  require(stringr)

  # we got a vector of sentences. plyr will handle a
list
  # or a vector as an "l" for us
  # we want a simple array ("a") of scores back, so
we use
  # "l" + "a" + "ply" = "laply":
  scores   =   laply(sentences,   function(sentence,
pos.words) {

    # clean up sentences with R's regex-driven global
substitute, gsub():
    sentence = gsub('[[:punct:]]', '', sentence)
    sentence = gsub('[[:cntrl:]]', '', sentence)
    sentence = gsub('\\d+', '', sentence)
    # and convert to lower case:
    sentence = tolower(sentence)

    # split into words. str_split is in the stringr
package
    word.list = str_split(sentence, '\\s+')
    # sometimes a list() is one level of hierarchy
too much
    words = unlist(word.list)

    # compare our words to the dictionaries of
positive
```

```
    pos.matches = match(words, pos.words)

    # match() returns the position of the matched term
or NA
    # we just want a TRUE/FALSE:
    pos.matches = !is.na(pos.matches)

    # and conveniently enough, TRUE/FALSE will be
treated as 1/0 by sum():
    score = sum(pos.matches)
    return(score)
  }, pos.words)

  scores.df        =        data.frame(score=scores,
text=sentences)
  return(scores.df)
}


neg.score.sentiment = function(sentences, neg.words)
{
  require(plyr);
  require(stringr)

  # we got a vector of sentences. plyr will handle a
list
  # or a vector as an "l" for us
  # we want a simple array ("a") of scores back, so
we use
  # "l" + "a" + "ply" = "laply":
  scores   =   laply(sentences,   function(sentence,
neg.words) {

    # clean up sentences with R's regex-driven global
substitute, gsub():
    sentence = gsub('[[:punct:]]', '', sentence)
    sentence = gsub('[[:cntrl:]]', '', sentence)
    sentence = gsub('\\d+', '', sentence)
    # and convert to lower case:
    sentence = tolower(sentence)

    # split into words. str_split is in the stringr
package
    word.list = str_split(sentence, '\\s+')
    # sometimes a list() is one level of hierarchy
too much
    words = unlist(word.list)

    # compare our words to the dictionaries of
negative
    neg.matches = match(words, neg.words)

    # match() returns the position of the matched term
or NA
    # we just want a TRUE/FALSE:
    neg.matches = !is.na(neg.matches)

    # and conveniently enough, TRUE/FALSE will be
treated as 1/0 by sum():
    score = sum(neg.matches)
    return(score)
  }, neg.words)

  scores.df        =        data.frame(score=scores,
text=sentences)
  return(scores.df)
}

aircanadap                                          <-
pos.score.sentiment(aircanada_all$text, pos.dic)
aircanadan                                          <-
neg.score.sentiment(aircanada_all$text, neg.dic)
```