SARCASM DETECTOR -
A TEXT CLASSIFICATION PROJECT

CONTENTS

SARCASTIC OR NOT - TEXT CLASSIFICATION

1. PROBLEM STATEMENT

Build a **classification model to identify Sarcasm.**

The Data is taken from Twitter in different languages. Given data set includes Tweets, Tweet ids and a target variable (sarcastic or non-sarcastic).

2. INTRODUCTION TO PROJECT

A. CASE STUDY EXPLANATION

This report is an analysis of Text data from Twitter. The data contains tweets from many **different languages.** We have classified whether a tweet is **sarcastic or non-sarcastic based on words used** in it. The aim of this project is to build a **Text classification model** to classify statements to sarcastic or non-sarcastic **to understand the true context** in which it appears.

B. PAIN AND GAIN ANALYSIS

The Pain and Gain analysis of this project is as tricky as identifying a real sarcastic statement. This needs very **subtlety in Perception and Understanding of the communication** usually varies from **Tone, Body language, Vocabulary and Absurdity levels** while communicating.
In **real world scenarios,** a person needs to understand the subtlety of this usage, requires **second-order interpretation** of the speaker's or writer's intentions; different parts of the brain must work together to understand sarcasm.
Using Analytics to identify Sarcasm will really change the game in understanding the true intention of a statement. This approach will reduce **Time Consumption of sentiment analysis** drastically there by it is used in many fields just like a **Spam filter**.
Applications are such as **analyzing product reviews** to know if a positive review is really positive or how many people are really supporting an idea or concept.

Examples: A French company has developed an **analytics tool** that claims to have up to 80% accuracy in identifying sarcastic comments posted online. Many organizations like **United States Secret Service** is requesting for Bids for Software to identify sarcasm in Tweets.

C. DOMAIN KNOWLEDGE

Sarcasm is "**a sharp, bitter, or cutting expression or remark; a bitter gibe or taunt**".

Sarcasm is usually used to express **disagreement, insult or ridicule** in such a way that a listener should judge the statement based on various aspects of **Tonality, Vocabulary, Ambiguity and Body language** to get the real meaning. Sarcasm is argued to be more sophisticated than lying because lying is expressed at as early as the age of three, but sarcastic expressions take place much later during development.

Sarcasm recognition and expression both require the development of understanding forms of language.
Few people may get it immediately, but **few may not get even after years** based on Intelligence especially **if sarcasm occurs without a cue or signal** (e.g., a sarcastic tone or rolling the eyes). It is considered that Sarcasm requires more Intelligence than usual, it needs creativity, various understanding forms of language.

There can be layers of sarcasm in a sentence. For Example, “Sarcasm is lowest form of wit” by Mark Twain. This popular statement is a sarcastic one representing how ‘sarcasm’ is highest form of wit. But most of the people take this statement seriously and understand it literally and miss the real context behind it.

3. EXPLORATORY DATA ANALYSIS

The Given dataset contains **91298 Observations and 3 Variables**- ID, tweet and label. The Independent variables needed for modelling should be structured, but our only predictor variable is unstructured. So, our **first goal is to convert the unstructured tweets to structured** variables (Terms). We use tm package for this text mining operations involved.

A. UNDERSTANDING THE DATA

Let's look at Summary and head of the data.

```
> summary(tweet)
      ID                tweet                      label
 Length:91298       Length:91298       non-sarcastic:39998
 Class :character   Class :character   sarcastic    :51300
 Mode  :character   Mode  :character
```

```
> head(tweet)
          ID                                                                         tweet     label
1: T000452358                                                b'oh yea that makes sense ' sarcastic
2: T000452359                  Estas enfermedad a un cargo poltico tu como pblico #jesuischarlieytal' sarcastic
3: T000452360 @alleygirl2409 until i\\'m and all the old men will finally date me #sarcasmsun mar 081243 ist sarcastic
4: T000452361        b""@sarinas it had been chanted peacefully you can't deny #hypocrisysat mar 154155 ist sarcastic
5: T000452362                     b""there's nothing like being on vacation and having to do your homework sarcastic
6: T000452363            People who are sarcastic tend to be better problem solvers "" I can solve anything sarcastic
> tail(tweet)
          ID                                                                 tweet         label
1: T000543650          b'this is me right know i still know every move from the i learned #grateful non-sarcastic
2: T000543651                                            b'rt look who visited today @royaltroongc non-sarcastic
3: T000543652                                                b'only minutes till the begins with non-sarcastic
4: T000543653                                       b'so beyond happy to be getting ios tonight ' non-sarcastic
5: T000543654 b'skyrim remastered releasing with sun and moon great birthday week for me ^^ #autumn non-sarcastic
6: T000543655                                b'such a big all probably leads to one #content #smile' non-sarcastic
```

By observing above head and tail of the data, we can infer that tweets are in languages **other than English.** By researching, it is found that there are many other languages like **Dutch, Spanish and Danish etc.** So, while preprocessing, I have **<u>included stop words of all the languages installed in R</u>**. Tweets contain many meaningless characters such as Punctuation marks, Numbers, unwanted spaces, single alphabets. We have to clean all these irrelevant characters from tweets. This is taken care in Pre Processing step.

The **label feature specifying the sentiment of the statement as sarcastic and non-sarcastic** is a character by default. We have converted it to factor while analysis.

B. UNDERSTANDING VISUALIZATIONS

For getting Our First level of Insights, we are going to discuss Visualizations here in this module. We have constructed three Word clouds- One for entire data set, one for each of two data sets with either Sarcastic or Non-sarcastic class in them. And a Bar plot is constructed to represent most frequent terms of the whole corpus which have **frequency more than 2000.**

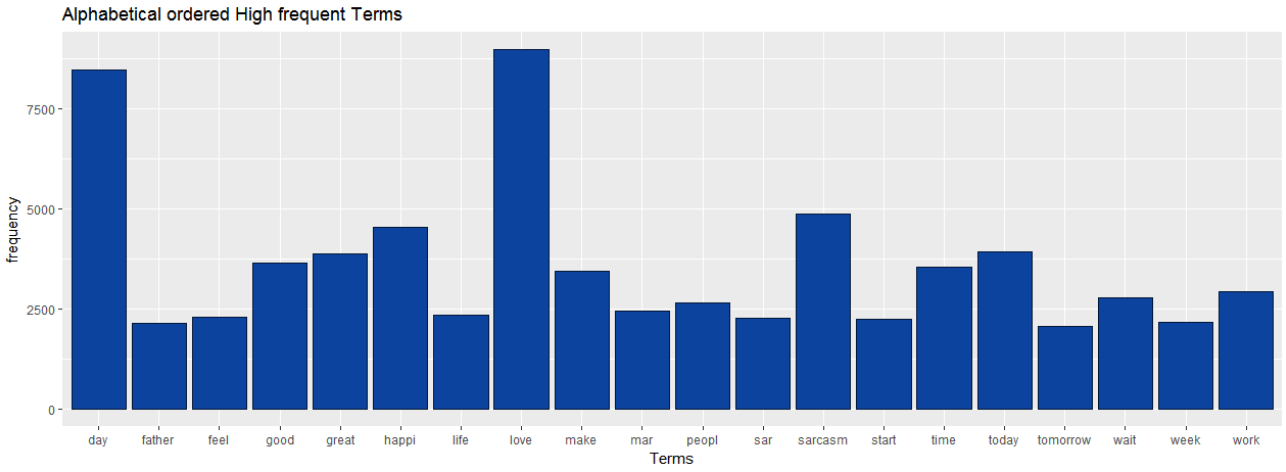
We are going to analysis these four visualizations and extract most frequent words which used in both Sarcastic and Non sarcastic tweets. It is very important to know that if a word occurs most frequently in both of the data set, then it is evident that those words doesn't contribute to our model in classifying Sarcastic tweets from non-sarcastic. We are going to eliminate all those frequent words which appear in both data sets for fair distribution of the data.

Word cloud for entire data set:

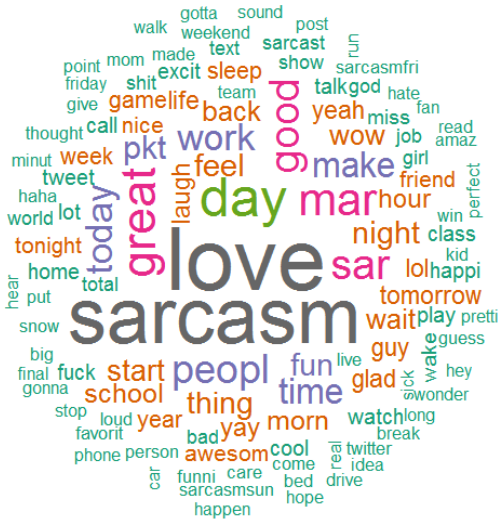
We can clearly see **'love', 'day', 'happi' (happy), and 'sarcasm' and 'today'** are top 5 most frequent words in the Corpus for whole data set.

We further ensure this hypothesis from a bar plot.

Bar plot representing most frequent words in Whole data set:



Let's visualize two word clouds which express the frequency of Terms for Sarcastic and Non-Sarcastic tweets:



Wordcloud for Sarcastic tweets

Wordcloud for Non-Sarcastic tweets

By observing two word clouds of Sarcastic and Non-sarcastic tweets, we have enough information to produce our first level of Insights.

FIRST LEVEL OF INSIGHTS OR KNOWN INSIGHTS

1. Sarcastic and non-sarcastic tweets have **fairly clear distinction** in between them.
2. There are only few words which are common in both the tweets which can confuse our model from classifying.
3. Words like **'love', 'day', 'today', 'tonight', 'danc'(dance), 'final'** are most common frequent words in both kinds.
4. There are few words which are frequent yet clearer in their appearance which are very helpful for the model in classifying.
5. Words like 'sarcasm', 'mar', 'pkt', 'sleep', 'make', 'yay', 'awesome', 'lol', 'bless' are frequent in Sarcastic tweets.
6. Words like 'father', 'thank', 'weekend', 'amp', 'smile', 'birthday' are frequent in Non-sarcastic tweets
7. We can find **sarcastic words are more of Exclamatory, intense and unusual** in their use.
   Ex: 'gamelife', 'awesome', 'yay', 'wonder', 'laugh' etc..,
8. Whereas n**on-sarcastic words are more social and domestic**. Ex: 'father', 'kid', 'summer', 'forward', 'meet' etc..,

4. PRE PROCESSING AND CLEANING- TEXT MINING

Our Unstructured Data should be converted to Structured Data in order to do modelling.
We are using Text mining's '**tm**' package to do this. A corpus is prepared to **preprocess and clean the data and do tokenization** on it.
A user defined function ***vec2clean.corp*** is created to do preprocessing and cleaning of the corpus. This function takes in a vector with all the tweets in it and convert it to a corpus and cleans it.
The following steps are done on the corpus:

A. CASE FOLDING

The first preprocessing step is Case folding. Here, we are converting all the letters in the **Corpus to lowercase** using R's base function *tolower*.

B. REMOVING NUMBERS

In this step, we are **freeing corpus from numbers**. Here, we use tm's *removeNumbers* function.

C. REMOVE STOPWORDS

This is very interesting preprocessing step. This step is about eliminating words that doesn't make any meaning. Usually, stopwords of English would be enough, but in our case, **we have tweets in different language**. We have tweets in English, Dutch, Danish, Spanish, French etc..,

To deal with this, a vector is created to have all the **stopwords of different languages**. And Stopwords('SMART') is also include which contains most of the common words in English other than the standard stopwords('en') provide.

Coming to user defined stopwords, we have few frequent words which can be considered. We are going to deal this in feature engineering with more analysis on them.

D. USER DEFINED FUNCTIONS

While checking data for initial insights, there are **few tweets which contain apostrophes** in them. If we remove them with other punctuation, it is merging two different words to one and causing ambiguity. Also, we observed there are **few tweets containing words which cause control flow problems** (such as break). So, we should take care to avoid this situation. I have written a **user defined function** named ***AposToSpace***, which takes corpus as argument and checks for **'single', 'double' quotes and control flow words 'break'** and replace them to a space and returns clean corpus.

E. REMOVING PUNCTUATION MARKS

We have use tm's ***removePunctuation*** function to **remove all punctuation marks** such as comma, full stop, parenthesis, various brackets etc.., from the corpus.

F. STEMMING

For grammatical reasons, document contains different **inflectional forms like tense forms and derivational forms**, we are performing **stemming to reduce** all those words **to their root word**. We are using tm's ***stemDocument*** function to do this. Stemming greatly help in reducing total number of terms and increase weighting.

G. STRIPING WHITESPACES

The above performed preprocessing steps **left our corpus with many leading and trailing whitespaces** within documents. We are cleaning all of them in one go using tm's ***stripWhitespace*** function. With this step our basic preprocessing is completed.

CORPUS TO DOCUMENT TERM MATRIX:

All the above discussed preprocessing and cleaning steps are **wrapped in a user defined function** for simplicity and usage across different code segments i.e, ***vec2clean.corp*** . The resulting corpus from *vec2clean.corp* is preprocessed and cleaned.

A **Document Term Matrix** (DTM) is created from the corpus. **Term Frequency** is considered as weighting to create Document term matrix to keep DTM simple. DTM has **91298** documents and **30555** terms with **100% sparsity**.

Sparsity is reduced and we made **30555 terms** to more relevant **958 terms**. We try to get a **balance between number of terms and vector size which R can allocate** while processing. This Document term matrix is then converted to Data frame for Feature engineering.

Data frame has **91298** observations and **959** features (including target class) which has **Document frequency** of varying from **74 to 8976**. We have successfully converted the raw **Unstructured Data to Structured Data**.

5. FEATURE ENGINEERING AND FEATURE SELECTION

This is most crucial step in any Data analysis. This proves the creativity of the Data scientist working.

A. REMOVING COMMON TERMS

As a part of our First level of Insights, while visualizing word clouds individually for sarcastic and non-sarcastic tweets, we have observed an interesting trend in it. There are **few words** which are more **frequently appearing in both classes**. It is sane to remove these terms from model building due to **their lack of contribution in classification by appearing in both classes**. ***findFreqTerms*** function is used to check top 100 frequent terms in both classes and extract common words. These terms are saved in '*common*' object. Words like **'tonight', 'day', 'dance', 'love', 'today' and 'final'** are found to be **common in both classes.**

B. CORRELATIONS AND ASSOCIATIONS

Checking correlation between the predictors is a must in Analysis. We have used tm's ***findAssocs*** **and pearson's correlation matrix** to detect **correlation and association.**

A 783 X 783 **correlation matrix** is built representing **positive and negative correlations** between terms. We have taken **50% as correlation limit** and filtered out highly correlated terms from our structured data. Words like "mar, pkt", "lot, laugh", "lot, loud", "laugh, loud", "iam, affirm" are **highly correlated pairs.** A term is taken from each correlation pair and made a vector called **corr.terms.**

'corr.terms' are combined with 'common' words and together removed from the data. This step has provided us with significant raise in accuracy while modelling (almost **4% percent of accuracy**).

After removing highly correlated terms and common terms, we are left with **948** terms.

FINAL DATA SET

After doing Feature Engineering on the Data, we have taken **two sets of data**. One containing **Numeric predictors** and other with **Factor predictors,** both along with target variable 'label'.

master contains **numeric predictors and target variable.**
master.factor contains **categorical predictors and target variable.** We did binning on numeric predictors and set **1 for the presence** of term and **0 for the absence.**

6. SAMPLING AND; CREATING TRAINING AND TESTING SETS

As **data is really huge** with **91298 X 948** dimensions (including target class), **Sampling is inevitable** to build our models. Our data is **gifted with having 44:56 ratio in target class**. We are using **stratified sampling** to **preserve this ratio throughout sampling and splitting**. '**caTools**' package is used to implement stratified sampling.

We have created a user defined **function *sample2train.test*** for **sampling the master data set** and **creating Training and testing sets.** *sample2train.test* will take 4 arguments in which 2 are optional. *sample2train.test* will take master data set and a number for seed to set random sample.

Remaining Optional parameters are sample ratio and training split ratio which are set to default of 0.055 to provide a **sample of more than 5022 observations** and 0.8 to have **80:20 as training and testing ratio.**

sample2train.test is designed in such a way, that it returns a list of Train and Test sets. We further assign them to train and test object and save them for modelling.

7. PREDICTIVE MODELLING

We are taking **10 samples of master data set and train our models** using *set.seed* function. We are considering **Naïve Bayes as our Base model** which is very significant in Text classification because of its **assumption of considering all variables equally important and independent. Random forest model** is our **ensemble model** in this analysis. As Random forest can handle numeric and factor data, we are providing both for a given sample and validate the performance.

All the models are trained with 5022 X 948 sample and split to 4018 X 948 Training and 1004 X 948 Testing sets.

A. NAÏVE BAYES MODELLING

Using '**e1071**' package, we are training our **Naïve Bayes classifier with factor data**. Being a Bayesian classifier with an assumption of having equally important and independent features, we are expecting a very good accuracy with Naïve Bayes. We are training the model with 10 samples changing seed.

Naïve Bayes model with default parameters trained on **10 samples have given quite good accuracy** with a **median 77.14** and **78.59 as highest accuracy**. Being a base model, Naïve Bayes gave very good accuracy with this huge data.

B. RANDOM FOREST MODELLING

Being an ensemble model, random forest is popular in providing very good model. We have **trained our Random forest model with Numeric data and Factor data**. Random forest model also gave good accuracies. We have used **h2o package** to build random forest models as it is really quick in training a model using h2o.

With **Numeric data**, our ensemble model provided us a **median accuracy of 74.65** and with **highest of 76.59**. With **Factor data**, our ensemble model provided a **median accuracy of 76.30** and **highest of 76.69**.

It is very interesting to know that **factor data is more informative in our Classification than numeric data**.

CHOOSING MODEL

We have modelled **GBM and Logistic regression** along with above models **as an experiment. Principal component analysis is done**, but they proved to be futile while modelling and hard to do PCA on huge data with R. After Evaluating all these with Naïve Bayes model and Random forest model with Numeric and Factor data with 10 samples of data, we are more inclined to choose our base model, **Naïve Bayes as our model to freeze** for our analysis.

8. IMPROVING MODEL PERFORMANCE

We have chosen our Naïve Bayes as our Model for this analysis. We are **tuning our Naïve Bayes** to get more reliable and robust model for Sarcastic or Not Text classification.

Being a classifier based on **conditional probabilities**, Naïve Bayes has a parameter called **Laplace estimator**. Generally while modelling, there are features who have **zero probability to a class which may cause disturbance while evaluating**. We have set this parameter to 1 (Laplace = 1). **Laplace estimator will make sure there are no zero probabilities throughout the model and set a minimum of 1.**

This **Laplace estimator tuning has increased accuracy** further and **made our model more appropriate and robust** for our problem. Naïve Bayes model with Laplace estimator has evaluated with a **median of 77.89** and a **maximum accuracy of 79.28**.

"Naïve Bayes model with Laplace estimator as 1 resulted us a more Robust model."

TABLE PRESENTING AUC FOR VARIOUS MODELS.

Random Seed	NB default	Numeric RF	Factor RF	NB laplace	GBM	LR
123	**78.59**	**76.59**	**76.69**	**79.28**	**73.41**	69.22
300	76.79	72.81	76.39	**79.28**	72.81	70.72
111	77.89	74.7	76.2	78.09	72.91	70.12
222	77.89	74.6	76.59	78.09	70.12	67.89
456	76.39	74.7	76.49	77.89	73.41	69.62
10	77.69	**76.59**	76.1	77.89	74.4	73.8
99	77.49	75.5	74.6	77.49	73.21	**70.92**
1215	75.3	71.31	76.1	77.09	71.71	68.33
201	74.6	73.01	76.39	76.59	70.72	69.22
256	76.2	73.71	75.1	76.49	70.92	67.93

9. ERROR METRICS

Considering error metrics for this problem "Sarcastic or Not" is easy to put on. As there are situations where a **sarcastic disapproval classified as non-sarcastic boasting statement may give false security** and fame to a topic or product. In the same way, **when a non-sarcastic negative comment or tweet is considered as sarcastic and decided as positive statement,** this too, takes away business. So, we **consider both False Positive and False Negative** and take them as whole as **Misclassification error and minimize it.**

We are **aiming to freeze the model which is giving least misclassification error.** So, the model and seed with highest accuracy is considered to be our model of Deployment.

Note: Sarcastic statement may not always be a negative comment. It can also be humor and positive appreciation with exaggeration

Hence, **Naïve Bayes model with Laplace estimator 1 and seed 123 is freeze to appropriate model for this Text classification problem.**

PACKAGES USED:

1. '**data.table**' for faster data manipulations
2. '**tm**' package for text mining
3. '**caTools**' for Stratified sampling and Training and Testing splits
4. '**caret**' for building confusion matrices
5. '**e1071**' for building Naïve Bayes classifier
6. '**h2o**' for faster implementation of Random forest
7. '**wordcloud**' for visualizing word clouds
8. '**ggplot2**' for bar plot visualizations.

10. CONCLUSION

"**Sarcastic or Not**" is a **Text classification problem** combined **with both Text mining and Data mining** is really an **interesting problem to work on.** Being a subtlest phenomenon, sarcasm is hard even in real world situation where a person should be more intelligent in different forms of language and vocals.
This is a real Data Science challenge with a bright future scope.

In this case study, we have converted **Unstructured Raw Data to clean preprocessed and Structured Data** with a lot of information to work on. We have **built correlation matrix and compared most frequent terms** between the target classes to help our model with more clearly classified information and finally built models and **tuned them to perfection.**