Data Mining
(CSE-362) Project
Odd Semester
2020-21

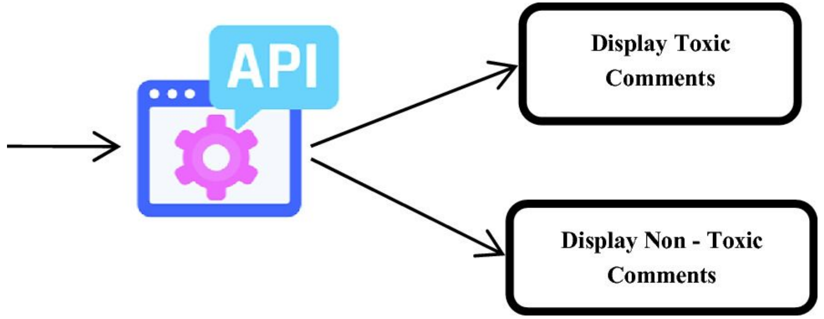
Hate Speech / Toxic Comment Detection

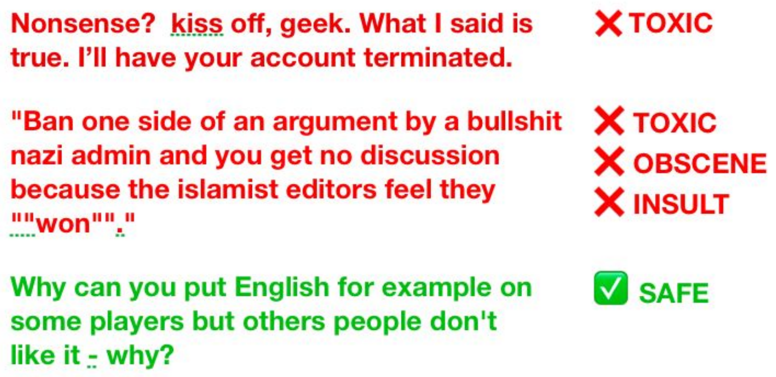
Under the guidance of
Dr. Bhaskar Biswas

Department of
Computer Science
and Engineering
IIT (BHU) Varanasi

Harshit Agrawal (18074019)
Ashish Kumar (18075068)
Sachin Srivastava (18075070)

27.09.2020





Nonsense? kiss off, geek. What I said is true. I'll have your account terminated.	✗ TOXIC
"Ban one side of an argument by a bullshit nazi admin and you get no discussion because the islamist editors feel they ""won""."	✗ TOXIC ✗ OBSCENE ✗ INSULT
Why can you put English for example on some players but others people don't like it - why?	✅ SAFE

Overview

Detection and Flagging of comments containing text which may incite violence, spread hate and induce negativity.

Introduction

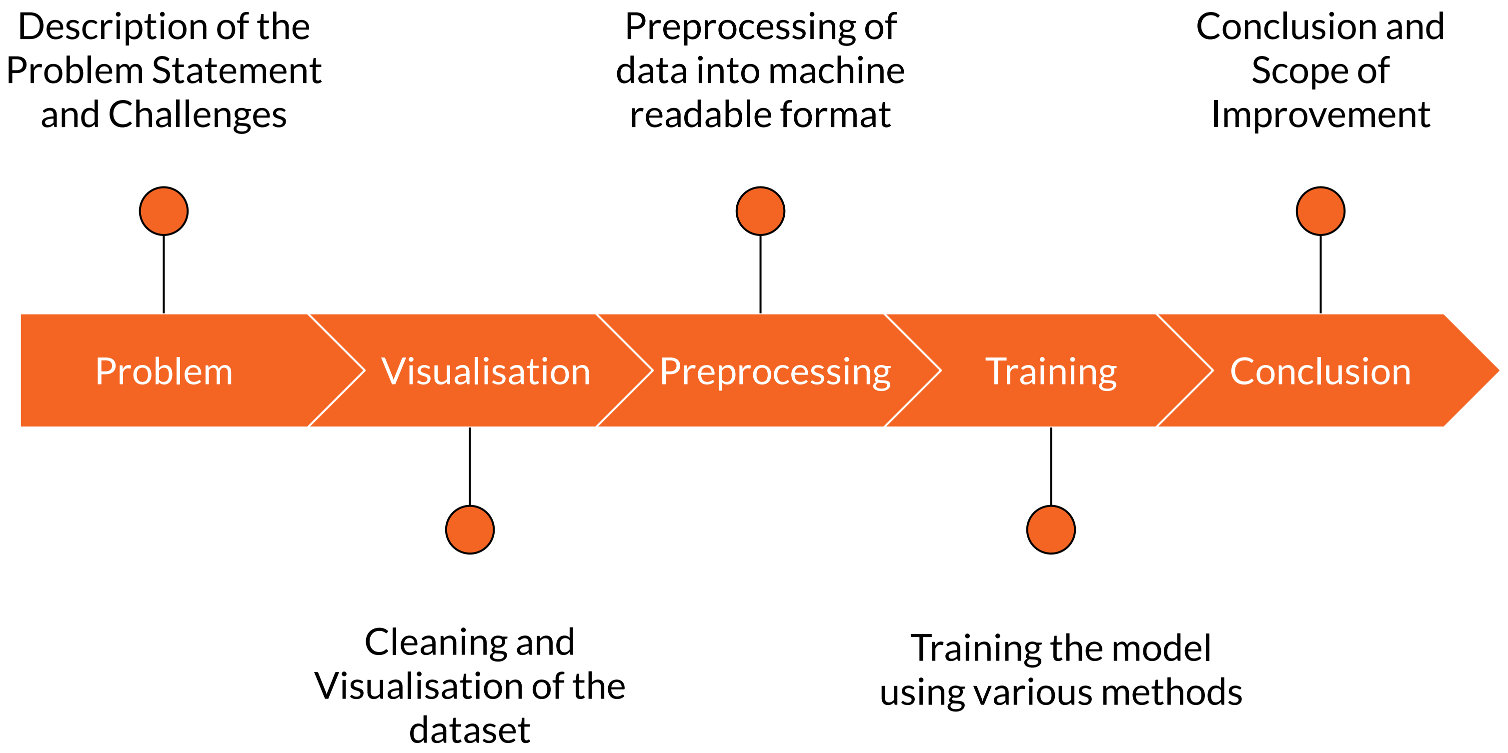
The project **Hate Speech / Toxic Comment Detection** aims at:

- Removing any textual material containing hate or toxicity.
- Online bullying has become a serious issue in recent years.
- The threat of abuse and harassment online means that many people stop expressing themselves.
- Toxic comments spread negativity, racial harassment and depression
- Removing hate speech makes easy for everyone to express their opinions freely over the internet.

Motivation & Applications

- Social networking: Message or tweet can be reported to the support team.
- Online meetings / webinars: Message can be hidden from attendees and shown only to the moderators.
- Chatbot training: The model can give large negative feedback for any insulting reply towards the user.
- Threatening messages: Directly reported to concerned authorities for immediate actions.

Structure of Presentation

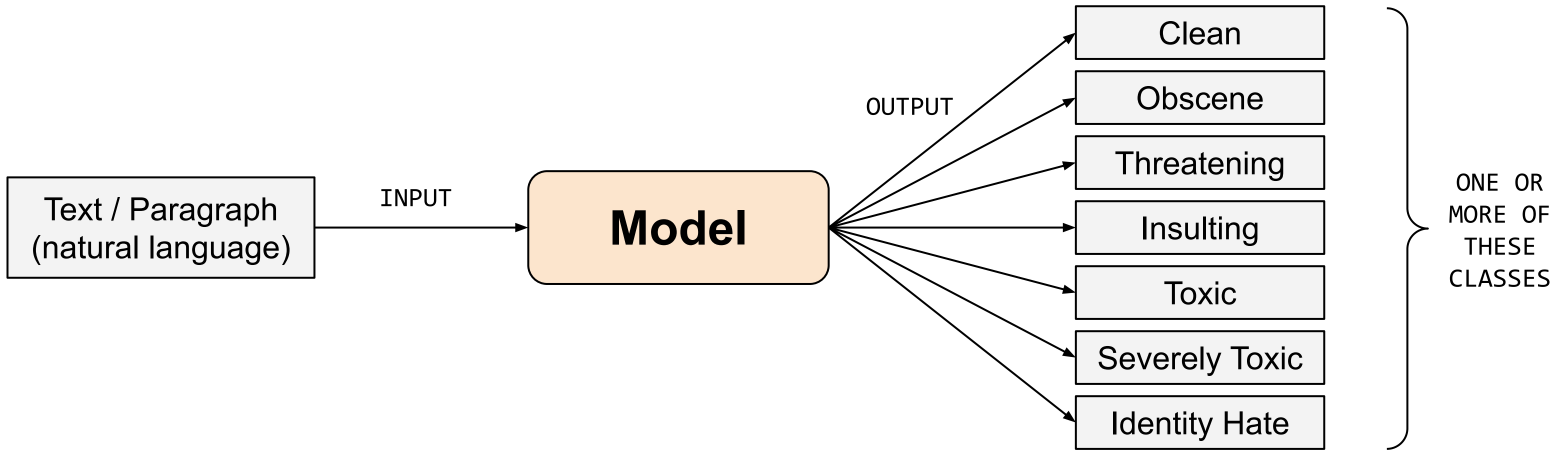

Description of the Problem Statement and Challenges
Preprocessing of data into machine readable format
Conclusion and Scope of Improvement
Problem
Visualisation
Preprocessing
Training
Conclusion
Cleaning and Visualisation of the dataset
Training the model using various methods

Problem Description

Problem Description

Statement

Given any text or paragraph containing a few lines in natural language, the objective is to classify it as belonging to one or more of the following categories: clean, obscene, threatening, insulting, toxic, severely toxic and identity hate.



Problem Description

Multi-class classification problem

Seven different classes:
Clean, obscene, threatening, insulting, toxic, severely toxic, identity hate.

➕

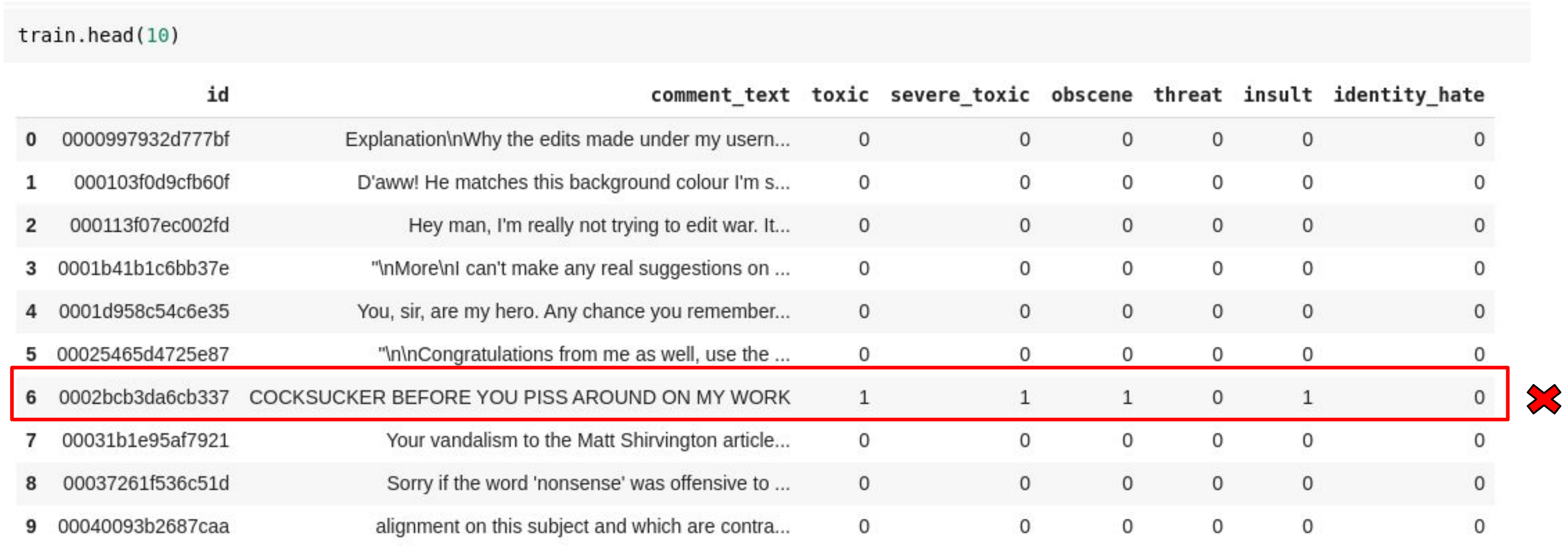
Multi-label classification problem

Any text or paragraph can be abusive in multiple ways: it can be both threatening & insulting to someone

- Model will output the **probability** of text to belong to each category.
- Based on certain **threshold** (which can be tuned as hyperparameter), the text can be classified to be belonging to a category / set of categories.

Dataset

- Dataset is made publicly available by Conversation AI.



```
train.head(10)
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

Labels

- The dataset has been labeled on a crowdsourced annotation platform.
- Different users may have different opinions about a comment (judgement bias).
- Some offensive ‘ASCII arts’ which may or may not be labelled as toxic by users depending on how it was rendered.
- Some extremely long comments got labelled to toxic without any significant reason.
- *“Give us your address and we'll come and kill you”* : Not marked as toxic!

Challenges

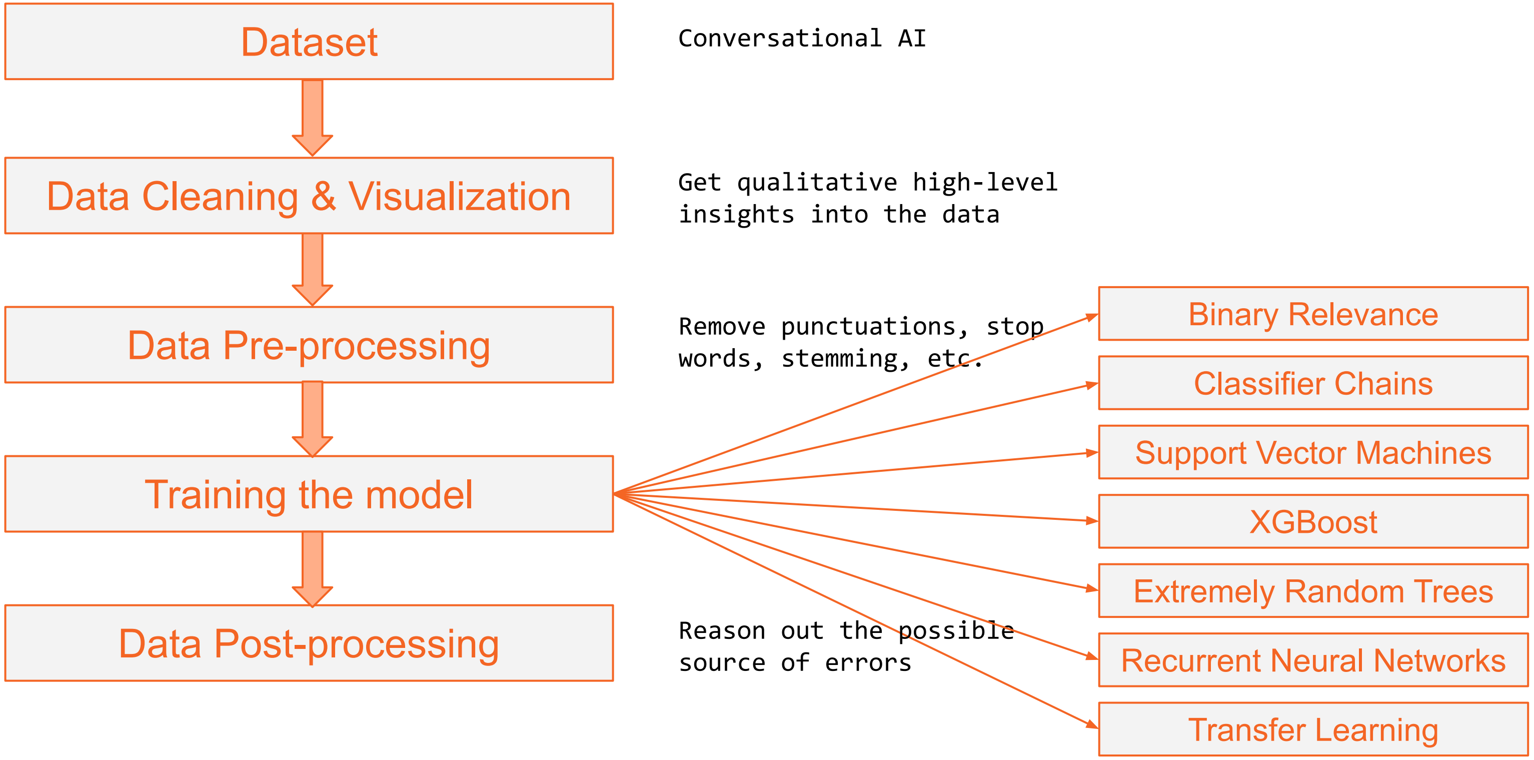
Noisy Data: Real world dataset from Internet users. Full of slangs, contractions, non-english or meaningless words and spams.

Unbalanced Dataset: Large number of comments are clean. Unclean samples are relatively rare and with six classes of toxicity problem is further aggravated.

Computational Capability: Natural Language Processing models like RNN requires huge processing powers and powerful GPUs for efficient training.

Memory Issues: More than 1 lakh training samples - difficult to store and process in internal memory at a time.

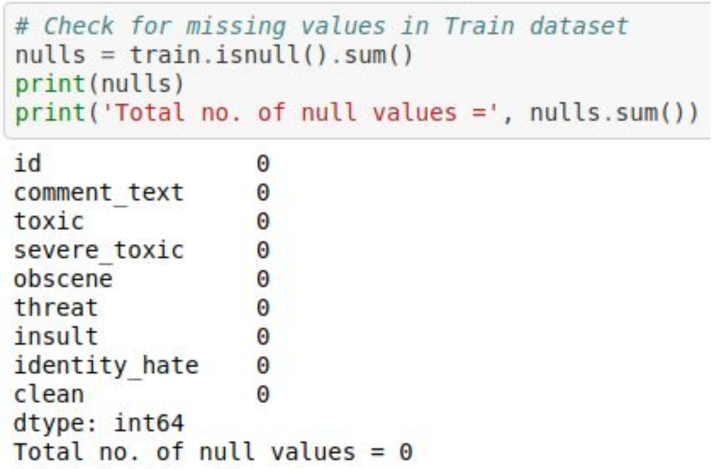
Solution Approach


Dataset
Data Cleaning & Visualization
Data Pre-processing
Training the model
Data Post-processing
Conversational AI
Get qualitative high-level insights into the data
Remove punctuations, stop words, stemming, etc.
Reason out the possible source of errors
Binary Relevance
Classifier Chains
Support Vector Machines
XGBoost
Extremely Random Trees
Recurrent Neural Networks
Transfer Learning

Data Cleaning & Visualisation

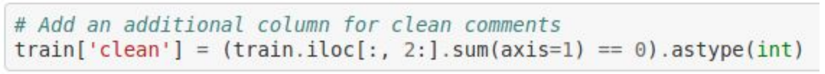
Data Cleaning & Visualization

- Checking for missing or null values.
- Adding extra column for ‘clean’ comments.

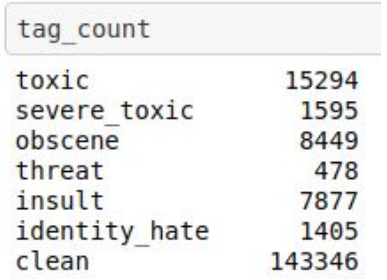


```
# Check for missing values in Train dataset
nulls = train.isnull().sum()
print(nulls)
print('Total no. of null values =', nulls.sum())
```

```
id               0
comment_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
clean            0
dtype: int64
Total no. of null values = 0
```

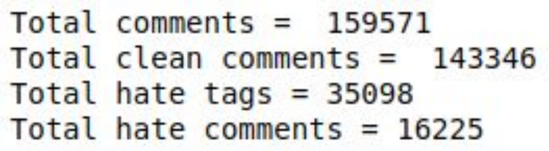


```
# Add an additional column for clean comments
train['clean'] = (train.iloc[:, 2:].sum(axis=1) == 0).astype(int)
```



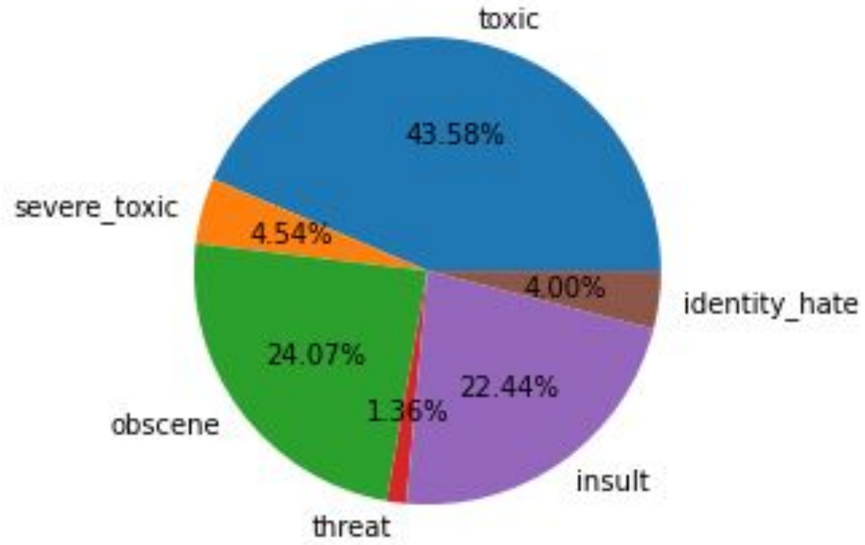
```
tag_count

toxic             15294
severe_toxic       1595
obscene            8449
threat              478
insult             7877
identity_hate      1405
clean            143346
```

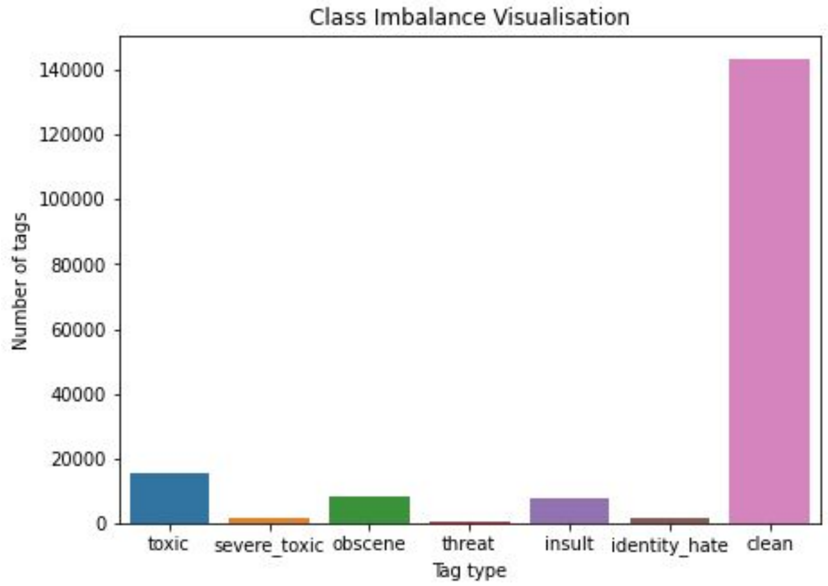


```
Total comments =  159571
Total clean comments =  143346
Total hate tags = 35098
Total hate comments = 16225
```

Data Cleaning & Visualization

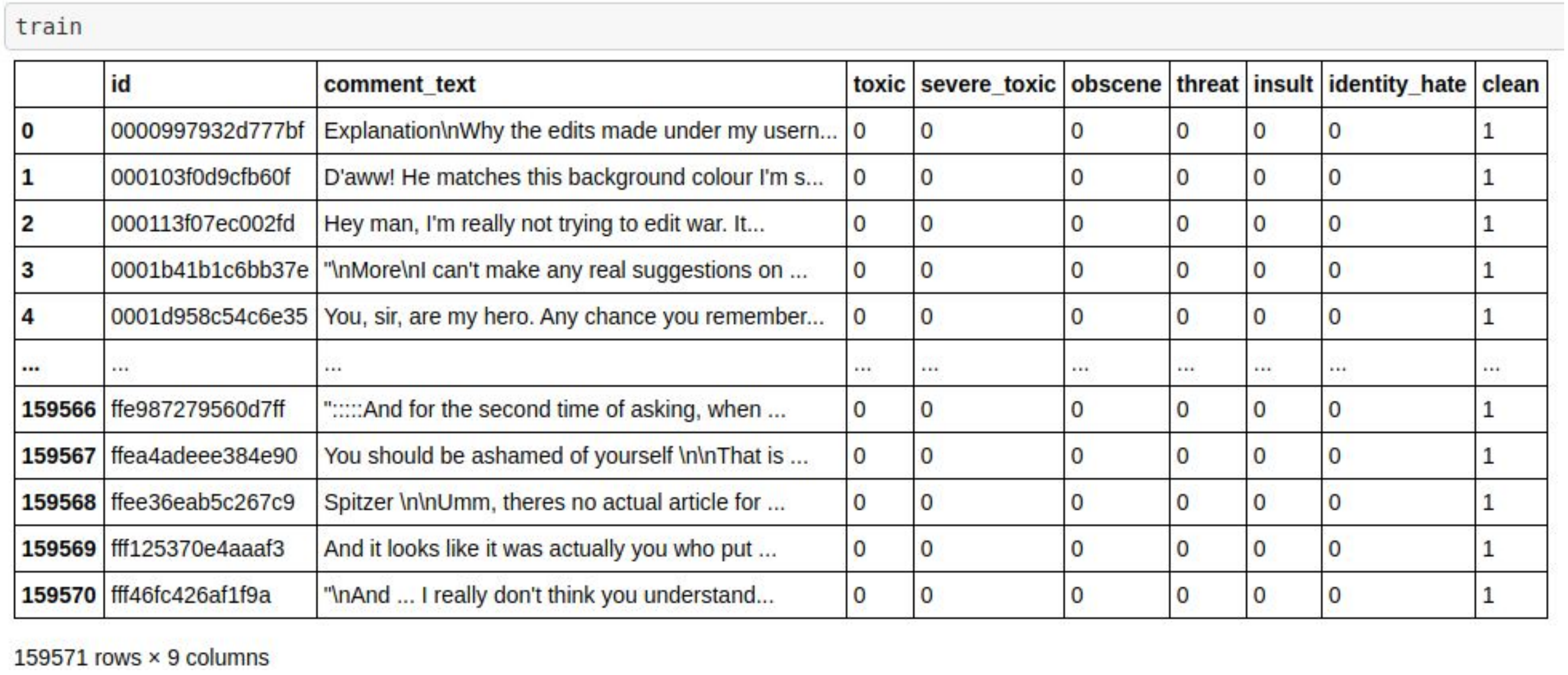


Class Imbalance: The toxicity is not spread evenly across all the classes. Out of all the hate tags, 'toxic' tags are 43.58%, whereas 'threat' tags are 1.36%.



Clean comments are ~140k out of the ~160k total comments.

Data Cleaning & Visualization



```
train
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate	clean
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	1
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	1
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	1
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	1
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0	1
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0	1
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0	1
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0	1
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0	1

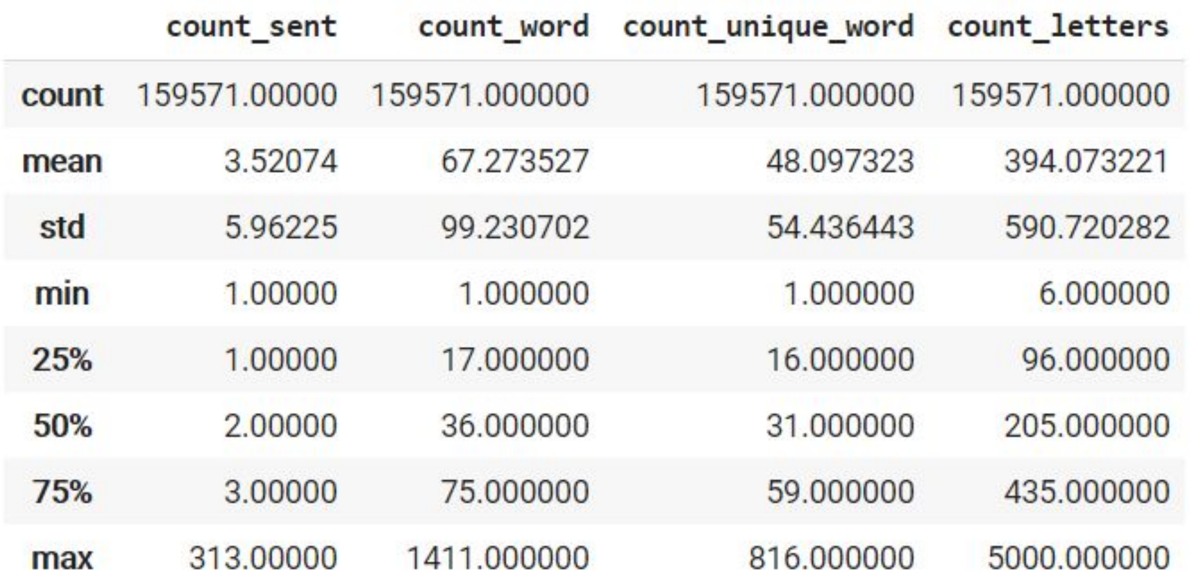
159571 rows × 9 columns

Data Cleaning & Visualization

- Comments obtained are from random users on internet
- Dataset is noisy. No fixed standard of writing across all the comments
- Dataset must be cleaned before further steps to obtain meaningful results
- Remove non alphanumeric characters (ip addresses, time, date) and punctuations.
- Convert everything to lower case letters.
- Detect and remove extremely long words (may cause overfitting), non english and meaningless words.

Data Cleaning & Visualization

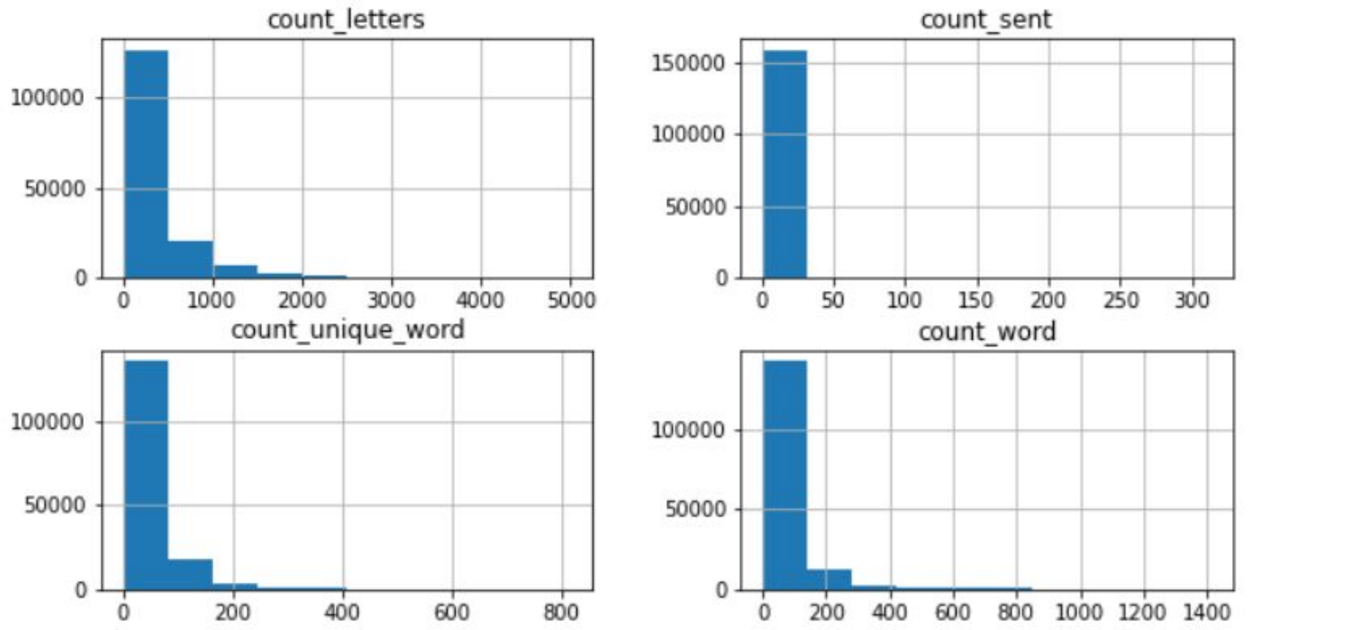
How long are comments?



	count_sent	count_word	count_unique_word	count_letters
count	159571.00000	159571.000000	159571.000000	159571.000000
mean	3.52074	67.273527	48.097323	394.073221
std	5.96225	99.230702	54.436443	590.720282
min	1.00000	1.000000	1.000000	6.000000
25%	1.00000	17.000000	16.000000	96.000000
50%	2.00000	36.000000	31.000000	205.000000
75%	3.00000	75.000000	59.000000	435.000000
max	313.00000	1411.000000	816.000000	5000.000000

Data Cleaning & Visualization

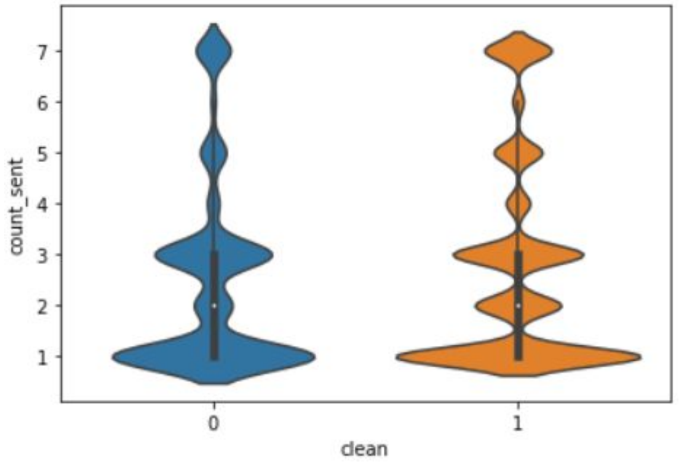
How long are comments?



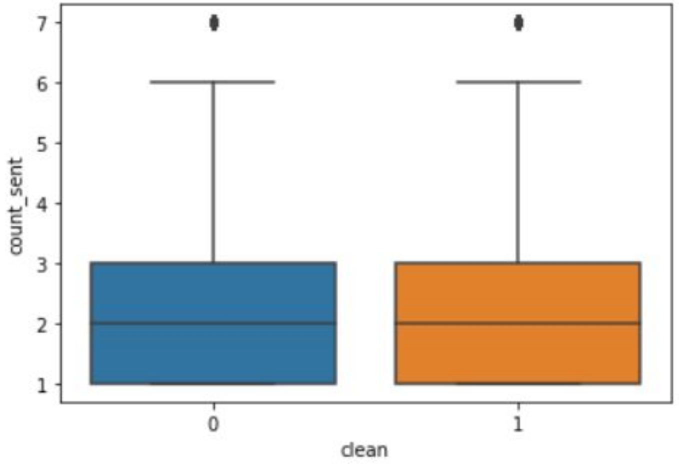
Data Cleaning & Visualization

Are longer comments more toxic?

Violin Plot



Box Plot

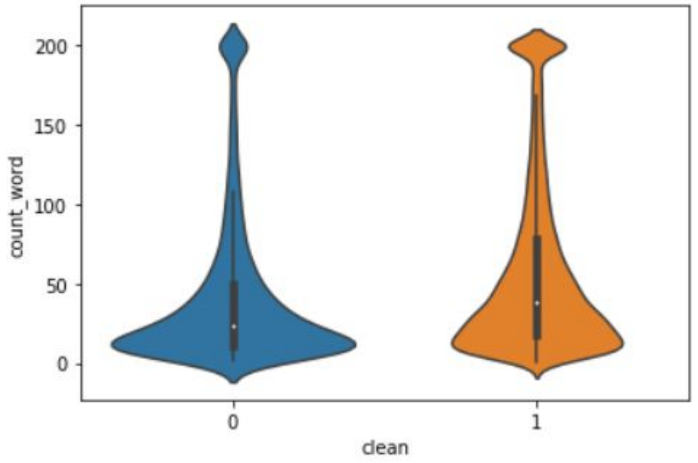


No significant correlation between length & toxicity is observed.

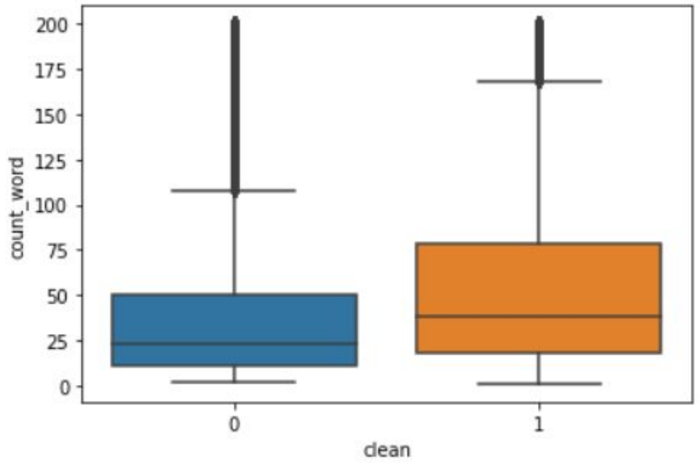
Data Cleaning & Visualization

Are longer comments more toxic?

Violin Plot



Box Plot



Slight correlation is observed. Lesser words increase the chances of toxicity.

Data Cleaning & Visualization

Are spammers more toxic?

A cursory look at comments reveal many comments which look like spam.

EPIC FAIL! EPIC FAIL EPIC FAIL! EPIC FAIL EPIC FAIL! ...
EPIC FAIL! EPIC FAIL EPIC FAIL! EPIC FAIL EPIC FAIL! ...

How to classify spammers?

Though spam detection is a wide area of research in its own we can use a very simple approach which suffice for our application.

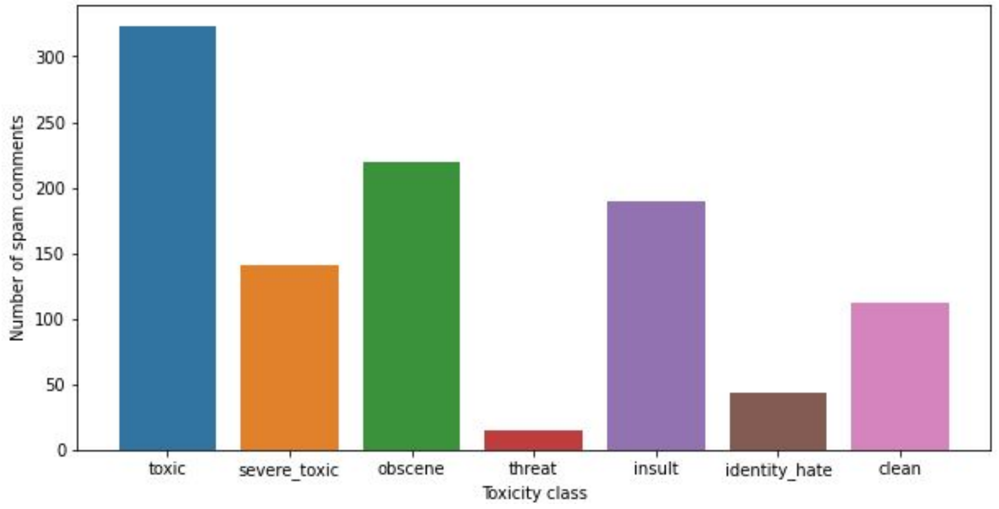
Take comments with lot of repeated words (i.e. very less % of unique words).

―

Data Cleaning & Visualization

Are spammers more toxic?

Bar Plot



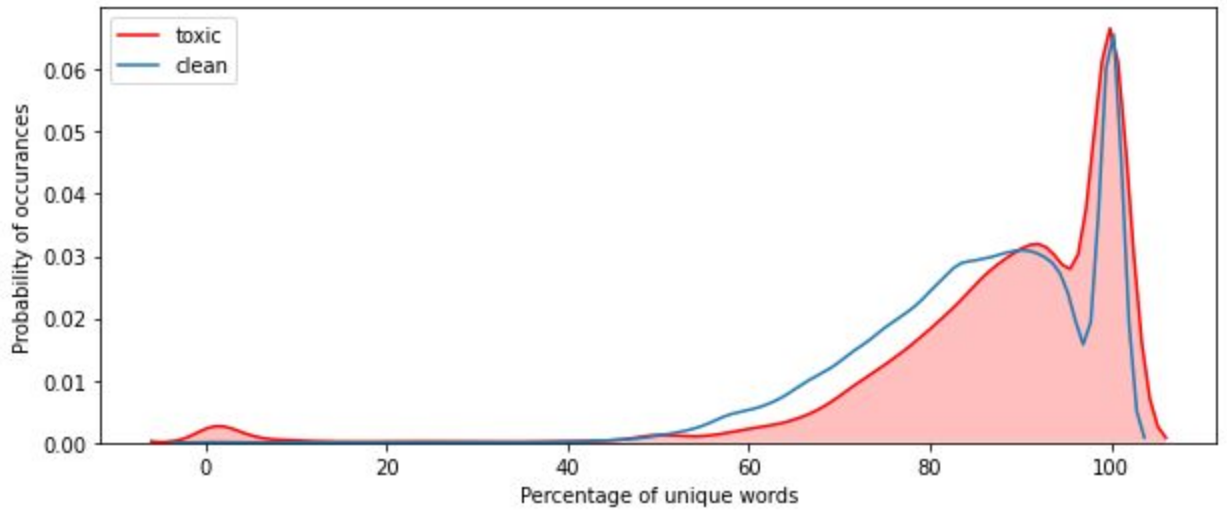
Lesser percentage of unique words increases the chances of toxicity.

—

Data Cleaning & Visualization

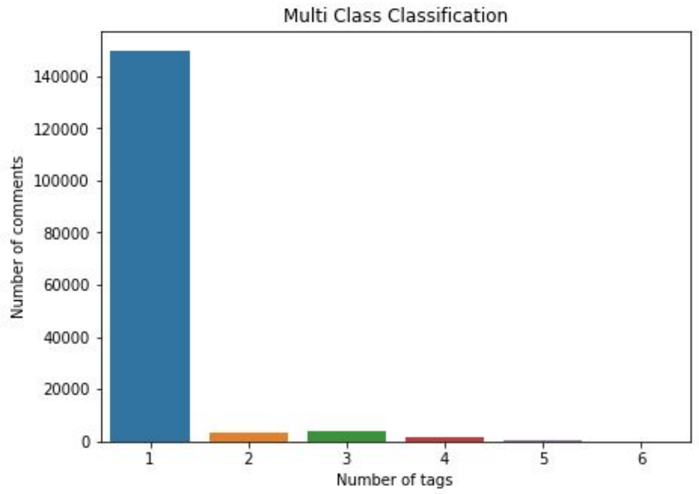
Are spammers more toxic?

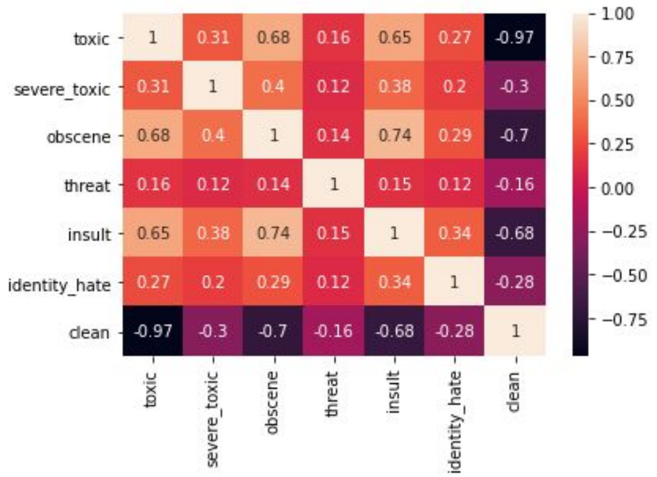
KDE Plot



More percentage of unique words **DOES NOT** decrease the chances of toxicity.

Data Cleaning & Visualization



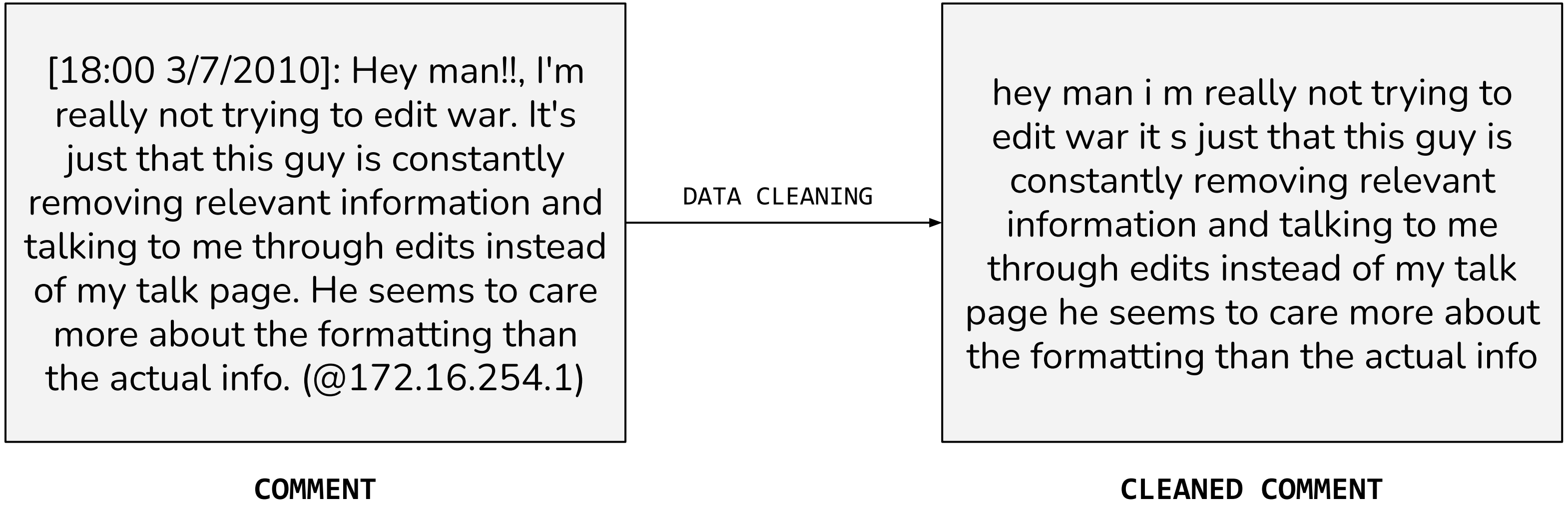


	toxic	severe_toxic	obscene	threat	insult	identity_hate	clean
toxic	1	0.31	0.68	0.16	0.65	0.27	-0.97
severe_toxic	0.31	1	0.4	0.12	0.38	0.2	-0.3
obscene	0.68	0.4	1	0.14	0.74	0.29	-0.7
threat	0.16	0.12	0.14	1	0.15	0.12	-0.16
insult	0.65	0.38	0.74	0.15	1	0.34	-0.68
identity_hate	0.27	0.2	0.29	0.12	0.34	1	-0.28
clean	-0.97	-0.3	-0.7	-0.16	-0.68	-0.28	1

Multi-class classification: A lot of comments have only one tag, yet there do exist some comment having 5 or 6 hate tags.

Heatmap of the correlation between the tags.

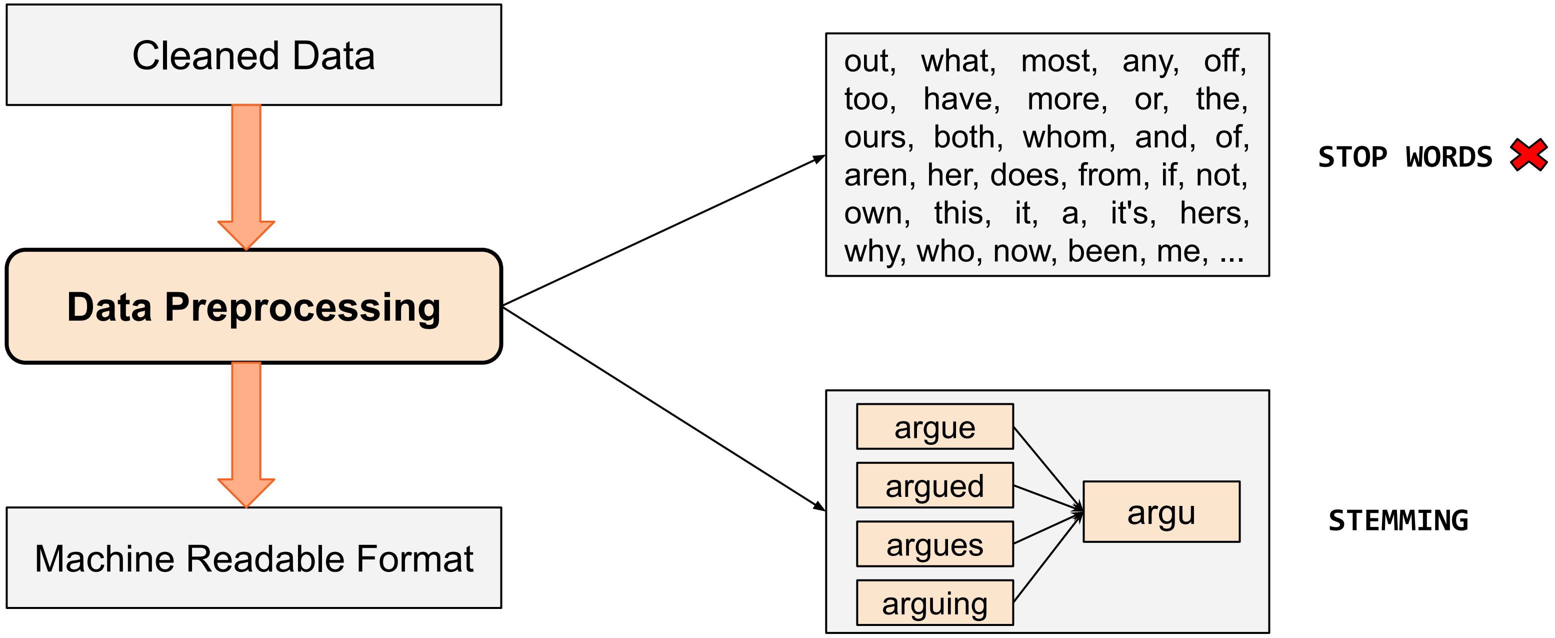
Data Cleaning & Visualization


[18:00 3/7/2010]: Hey man!!, I'm really not trying to edit war. It's just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page. He seems to care more about the formatting than the actual info. (@172.16.254.1)

hey man i m really not trying to edit war it s just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page he seems to care more about the formatting than the actual info

Data Preprocessing

Data Preprocessing


Cleaned Data
Data Preprocessing
Machine Readable Format
out, what, most, any, off, too, have, more, or, the, ours, both, whom, and, of, aren, her, does, from, if, not, own, this, it, a, it's, hers, why, who, now, been, me, ...
STOP WORDS ✖
argue
argued
argues
arguing
argu
STEMMING

Data Preprocessing

Conversion of cleaned data to machine readable format.

Stop Words

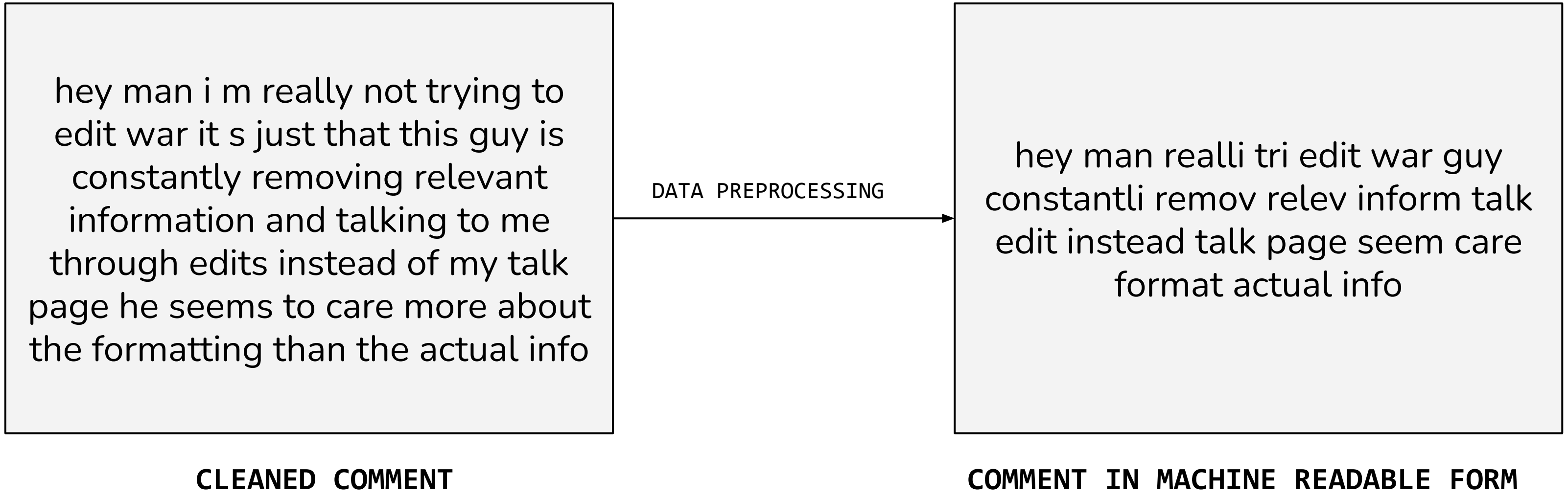
- Commonly used words which don’t add any meaning to the text.
- E.g.: ‘is’, ‘are’, ‘this’, ‘at’, etc.
- Removed the stop words using **NLTK stopwords corpus**.

Data Preprocessing

Stemming

- Transforming a word into its base stem, i.e., a set of characters to construct a word and its derivatives.
- Helps in reducing the vocabulary of words, which convey the same meaning.
- E.g. *'argue'*, *'argued'*, *'argues'* and *'arguing'* are reduced to the base stem *'argu'*.
- Used NLTK implementation of the **Porter Stemming Algorithm** - it contains a set of rules to transform a word into its base stem.

Data Preprocessing



hey man i m really not trying to edit war it s just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page he seems to care more about the formatting than the actual info

hey man realli tri edit war guy constantli remov relev inform talk edit instead talk page seem care format actual info

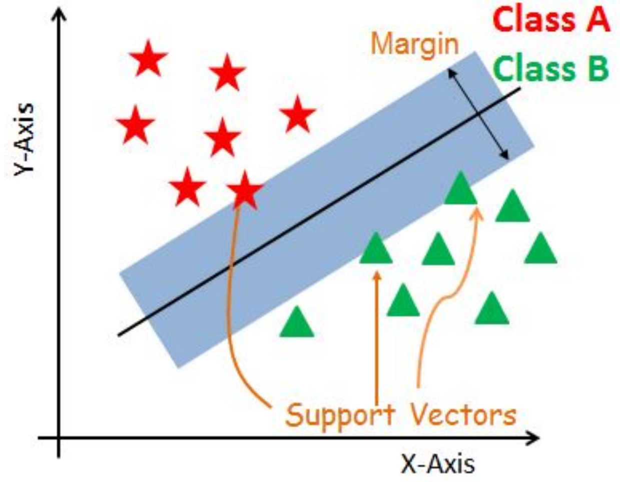
CLEANED COMMENT

COMMENT IN MACHINE READABLE FORM

PPT on 30.10.2020

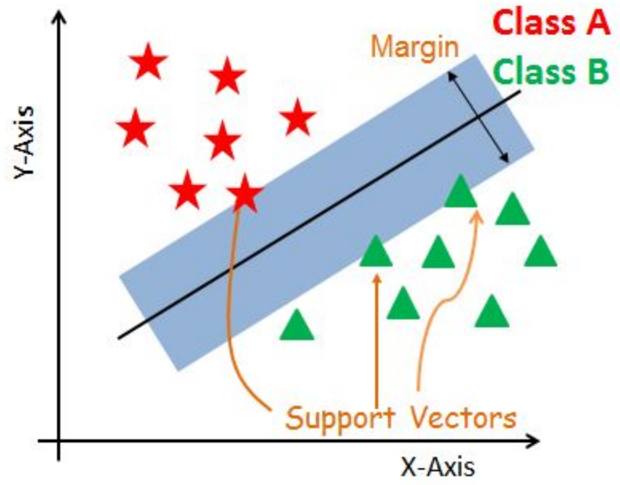
Support Vector Machines

- Constructs a Maximum Marginal Hyperplane that best divides the dataset into classes.
- **Hyperplane:** decision plane that separates a set of objects belonging to different class.
- **Support Vectors:** Data points which are closest to the hyperplane.
- **Margin:** gap between the two lines on closest class points.



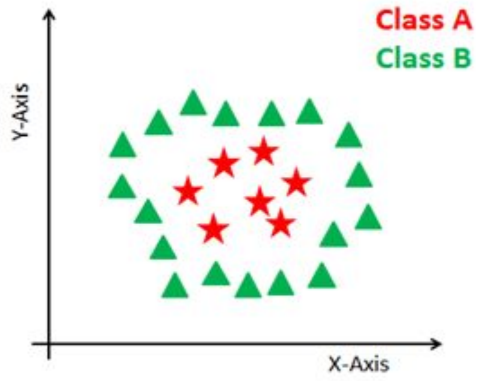
Support Vector Machines

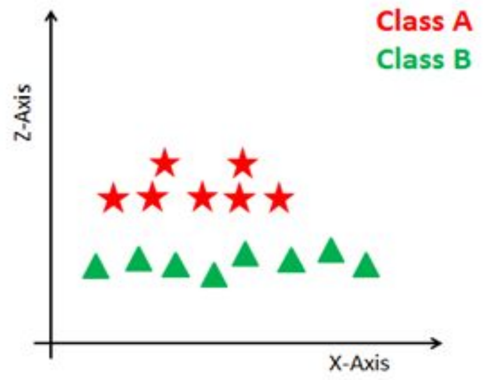
- Generates optimal hyperplane iteratively to minimize the error.
- **Objective:** Find hyperplane with maximum possible margin between support vectors.
- Can be used for both linearly separable as well as non-linearly separable data.



Support Vector Machines

- For **non-linear** and inseparable planes, SVM uses **Kernel trick**.
- Transforms input space to higher dimensional space.
- Here, data points are plotted on x-axis and z-axis, where $z = x^2 + y^2$.
- These data points can be segregated linearly.





Encoding Text Data

Count vectoriser and TF-IDF vectoriser:

- Used for text feature extraction
- Transform text → meaningful representation of numbers
- To fit Machine Learning algorithm for prediction.
- Tokenizing, counting, normalizing are used to extract features from text.
- **Tokenizing:** Text → split into tokens & assigning integer value to each token.
- **Counting:** counting occurence of token in document
- **Normalizing:** Diminishing those tokens that occur in most of the documents.

—

Encoding Text Data

Count vectoriser:

- Converts text to lowercase, removes punctuations & stopwords.
- Builds vocabulary of text, containing text from all documents
- Encodes the text into **vectors with dimension |V| x |D|**, where |V| is size of vocabulary, and |D| is the number of documents.
- Each element of the vector represents the frequency of the token in the document.
- **Issue:** Certain words which occur in most of the documents won't be very meaningful in the encoded vectors.

Encoding Text Data

TF-IDF vectoriser:

- Reduces impact of tokens that occur in most of the documents.
- Value of a term in vector increases proportionally to frequency of words in document
- If the token appears in most of the documents, then that term is penalized.
- **Term Frequency - TF(t, d):** Number of times a term ‘t’ appears in document ‘d’.
- **Inverse Document Frequency - IDF(t):** Inverse of document frequency DF(t) - number of documents that contain a term t.

$$IDF(t) = \log \frac{1+|D|}{1+df(d,t)} + 1$$

Encoding Text Data

TF-IDF vectoriser:

$$\text{TF-IDF} = TF(t, d) \times IDF(t)$$

- TF(t, d): High frequency of term in document increases weight of TF-IDF (local parameter).
- IDF(t): Low document frequency of term in whole corpus increases weight of TF-IDF (global parameter).
- TF-IDF: Considers both local and global impact of a word in vocabulary.
- Performs better than Count Vectorizer for feature extraction.

Encoding Text Data

Count Vectorizer

- Simple way to tokenize text and build vocabulary of words.
- Directly gives frequency of token w.r.t index of vocabulary.

TF-IDF Vectorizer

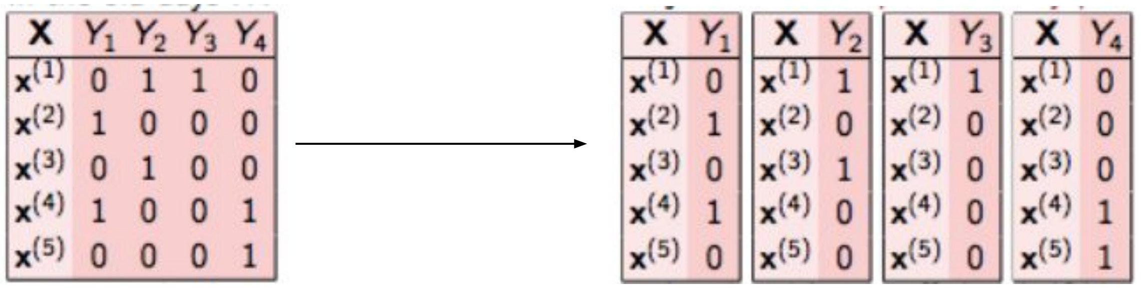
- Compares no. of times word appears in document with no. of documents containing that word.
- Assigns the value by considering overall document of words, penalizing the words having high document frequency.

Problem transformation

- Transforms the problem into separate single class classification problem.
- Methods Used - Binary Relevance, Classifier Chains
- **Binary Relevance:** Each label is treated as separate single class classification problem.
- **Classifier Chains:** First classifier is trained on the input, then subsequent classifiers are trained on input + all previous classifiers in the chain.

Problem transformation

Binary Relevance:



X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

X	Y_1
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0

X	Y_2
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	1
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

X	Y_3
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

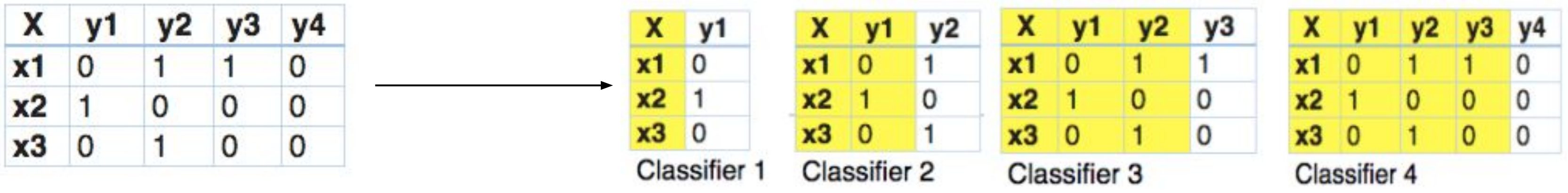
X	Y_4
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	1

Four different single-class classification problem

Drawback: Doesn’t consider correlation between the labels, treats every target variable as independent.

Problem transformation

Classifier Chains:



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

X	y1
x1	0
x2	1
x3	0

Classifier 1

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

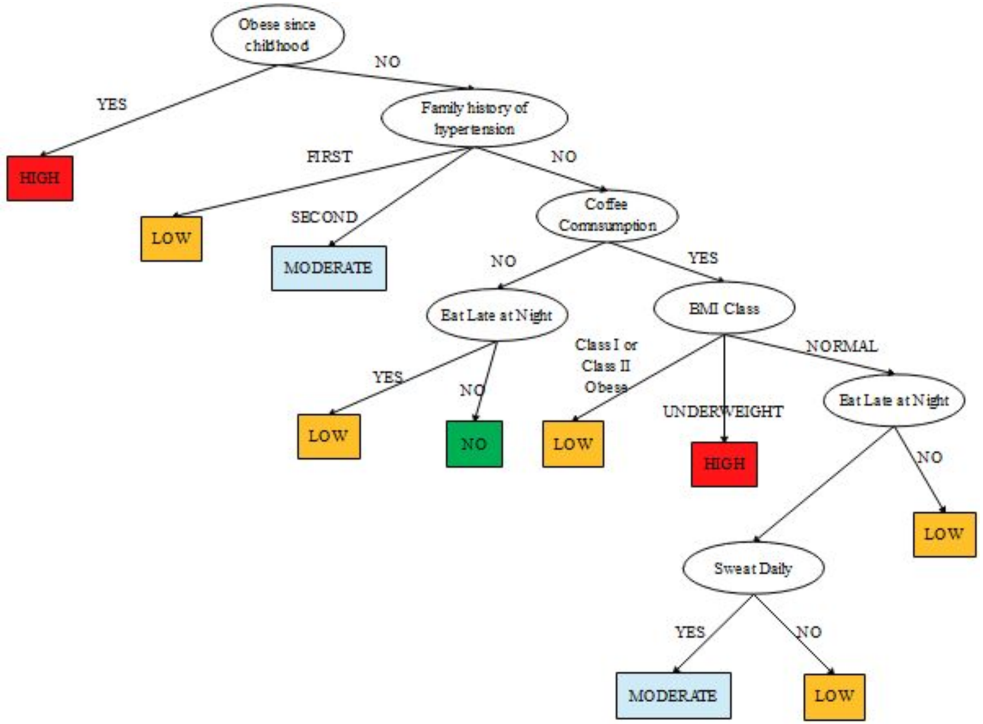
Four different single-label problem
Yellow: input space, White: target variable

- Forms chains in order to preserve label correlation.
- Generally performs better than binary relevance when correlation between labels is involved.

Decision Trees

- A **non-parametric** supervised learning method used for classification and regression.
- The goal is to create a model that predicts the value of a target variable by learning **simple decision rules** inferred from the data features.
- In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making
- Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop

Decision Trees



Decision Trees

- Tree models where the target variable can take a discrete set of values are called **classification trees**
- In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels
- Advantages:
    - Able to handle multi-output problems.
    - The cost of using the tree is logarithmic in the number of training data points.
    - Uses a white box model
    - Requires little data preparation

Decision Trees

Shortcomings:

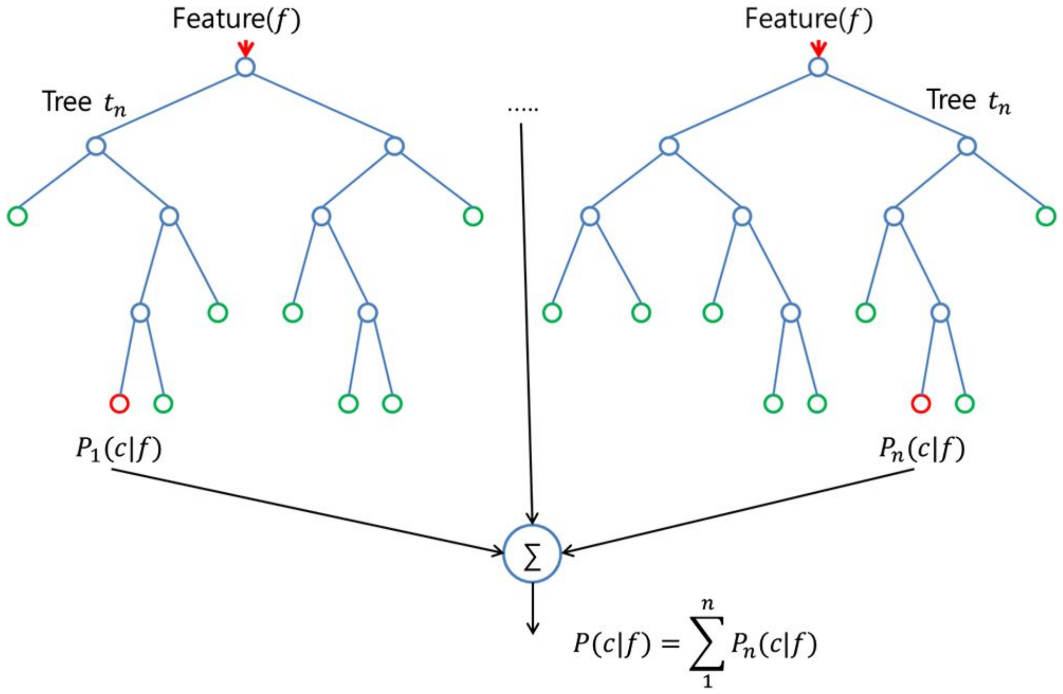
- Decision-tree learners can create over-complex trees: **overfitting**
- The problem of learning an optimal decision tree is known to be **NP-complete**.
- There are concepts that are **hard to learn** because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create **biased trees** if some classes dominate.
- Decision trees can be **unstable** because small variations in the data might result in a completely different tree being generated.

Some of these problems can be mitigated by using decision trees with an ensemble.

Decision Trees Ensembling

- Ensemble methods, combines several decision trees to produce better predictive performance than utilizing a single decision tree.
- A group of weak learners come together to form a strong learner!
- Ensembled Trees are composed of a **large number of decision trees**, where the final decision is obtained taking into account the prediction of every tree.
- Specifically, by **majority vote** in classification problems, and by the **arithmetic mean** in regression problems.
- All of these ensemble methods take a decision tree and then apply either **bagging** (bootstrap aggregating) or **boosting** as a way to reduce variance and bias.

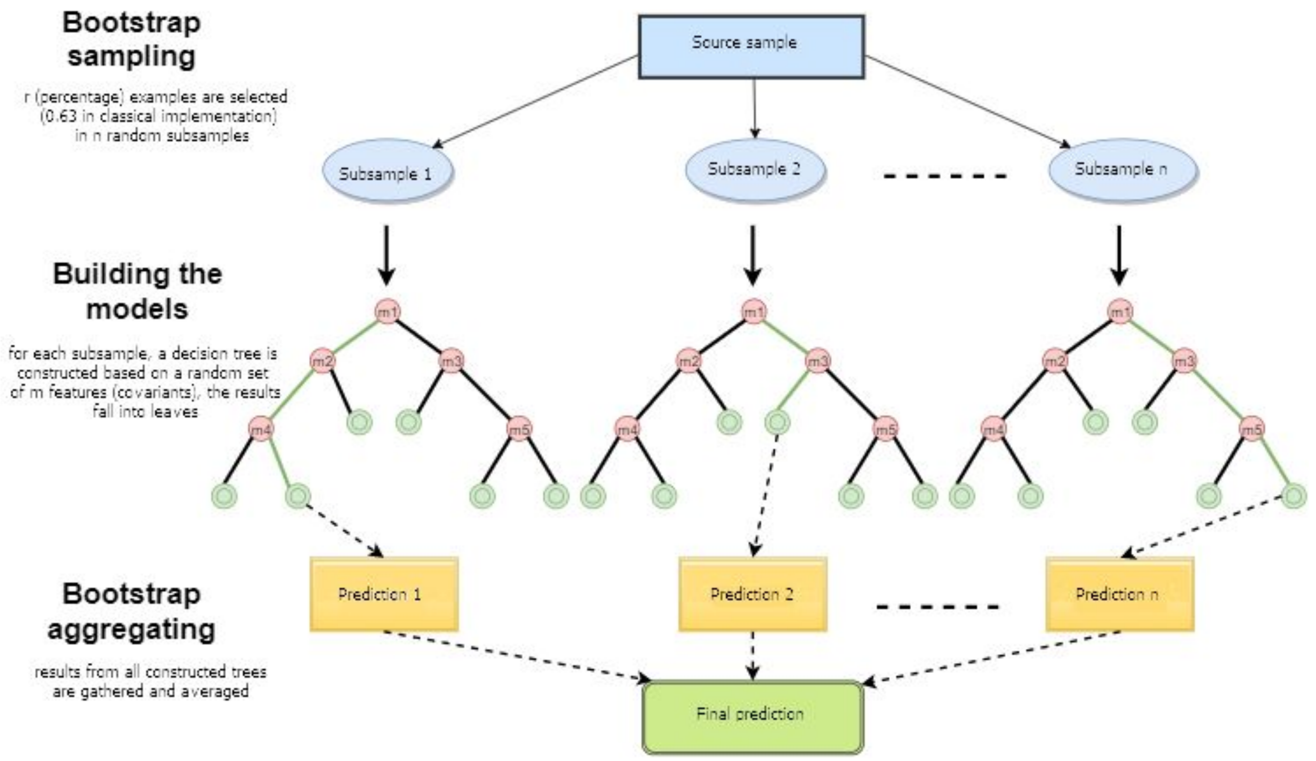
Decision Trees Ensembling



Bagging

- Bagging (Bootstrap Aggregation) is is a general technique for combining the predictions of many models
- It is used when our goal is to reduce the variance of a decision tree.
- The idea is to create several subsets of data from training sample chosen randomly with replacement.
- Now, each collection of subset data is used to train their decision trees.
- As a result, we end up with an ensemble of different models.
- Combination of all the predictions from different trees are used which is more robust than a single decision tree.

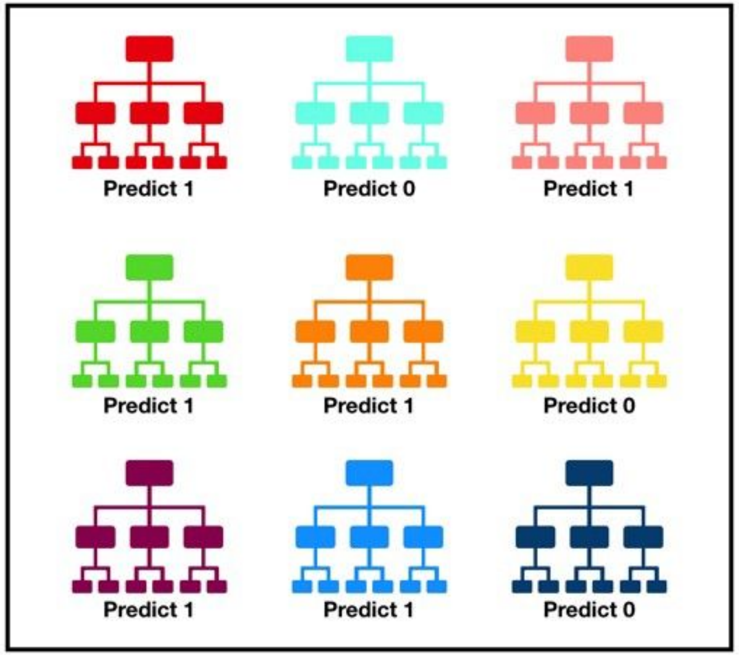
Bagging


Bootstrap sampling
r (percentage) examples are selected (0.63 in classical implementation) in n random subsamples
Source sample
Subsample 1
Subsample 2
Subsample n
Building the models
for each subsample, a decision tree is constructed based on a random set of m features (covariants), the results fall into leaves
m1
m2
m3
m4
m5
Bootstrap aggregating
results from all constructed trees are gathered and averaged
Prediction 1
Prediction 2
Prediction n
Final prediction

Random Forest

- Fitting decision trees on bootstrapped data serves to decorrelate them slightly.
- Since each node chooses what feature to split on in a greedy manner, trees can still end up very correlated with each other.
- Random forests **decorrelates the individual decision trees**.
- When the CART algorithm is choosing the optimal split to make at each node, the random forest will choose a **random subset of features** and only consider those for the split.
- From that random subset of features, the algorithm will still select the **optimal feature** and **optimum split** to make at each node.

Random Forest



Tally: Six 1s and Three 0s
Prediction: 1

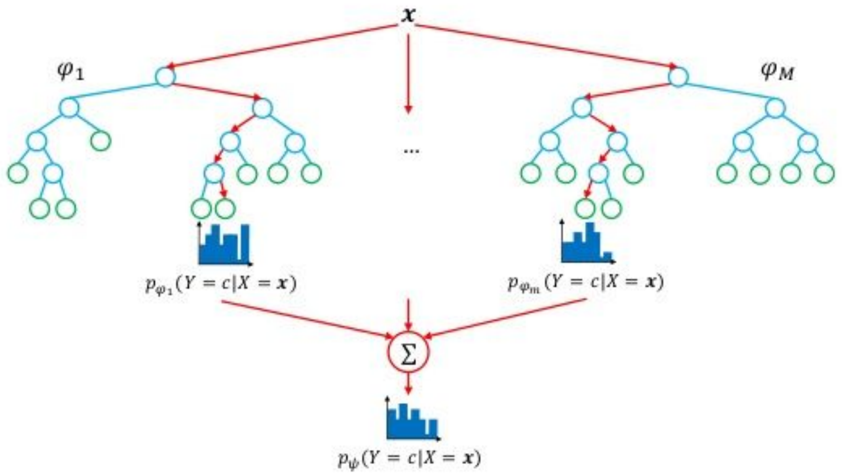
Extremely Randomized Trees

- Much more **computationally efficient** than random forests.
- Performance is almost always comparable. In some cases, they may even perform better!
- Adds one more **step of randomization** to the random forest algorithm.
- Random Forest chooses the optimum split to make for each feature while Extra Trees chooses it randomly - so it is **faster**.
- Both of them will subsequently choose the best feature to split on by comparing those chosen splits.
- From this reason comes the name of Extra Trees (**Extremely Randomized** Trees)

Extremely Randomized Trees

- Random forest uses **bootstrap replicas** it subsamples the input data with replacement.
- Extra Trees use the whole original sample.
- In the Extra Trees sklearn implementation there is an optional parameter that allows users to bootstrap replicas, but by default, it uses the entire input sample.
- These differences motivate the **reduction of both bias and variance**.
- Using the **whole original sample** instead of a bootstrap replica will reduce **bias**.
- **Choosing randomly the split** point of each node will reduce **variance**.

Extremely Randomized Trees


$\boldsymbol{x}$
φ_1
φ_M
...
$p_{\varphi_1}(Y = c|X = \boldsymbol{x})$
$p_{\varphi_m}(Y = c|X = \boldsymbol{x})$
Σ
$p_{\psi}(Y = c|X = \boldsymbol{x})$

Randomization

- Bootstrap samples
- Random selection of $K \leqslant p$ split variables
- Random selection of the threshold

} Random Forests

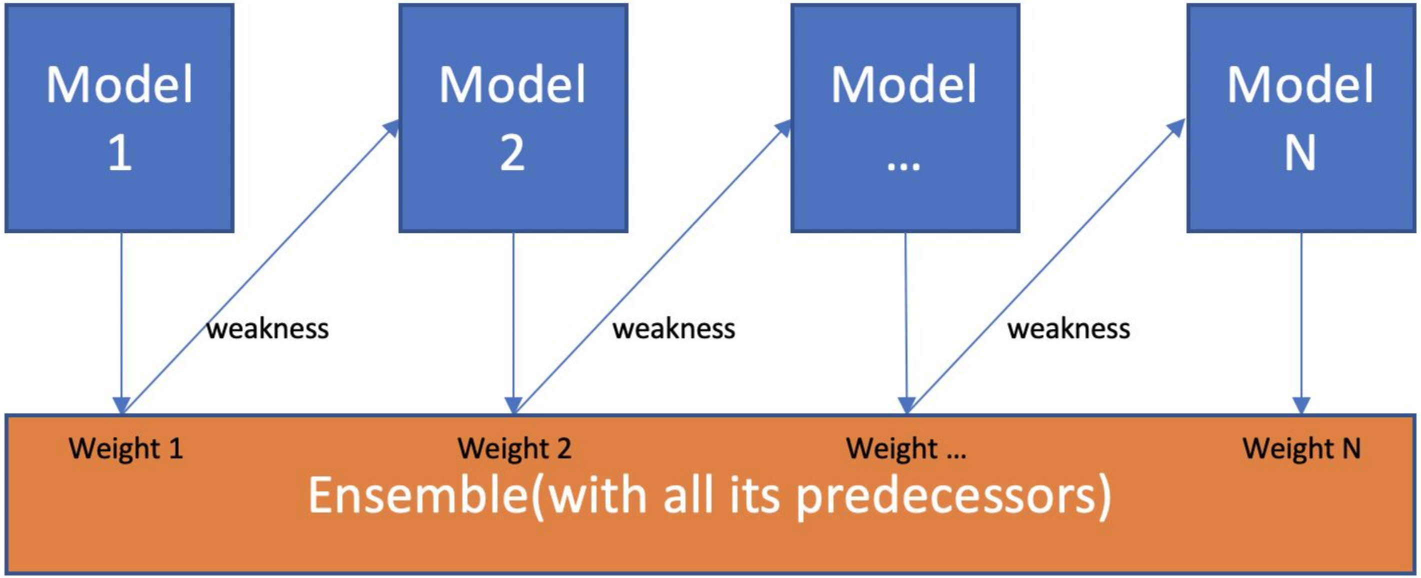
} Extra-Trees

Boosting

- **Boosting** is another ensemble technique to create a collection of predictors.
- In this technique, learners are learned **sequentially**.
- Early learners are fitted with simple models to the data. Then data is analysed for errors.
- In other words, we fit **consecutive trees** (random sample) and at every step, the goal is to solve for net error from the prior tree.
- When an input is misclassified by a hypothesis, its weight is increased so that next hypothesis is more likely to classify it correctly.
- By combining the whole set at the end converts weak learners into better performing model.

Boosting

Model 1,2,..., N are individual models (e.g. decision tree)

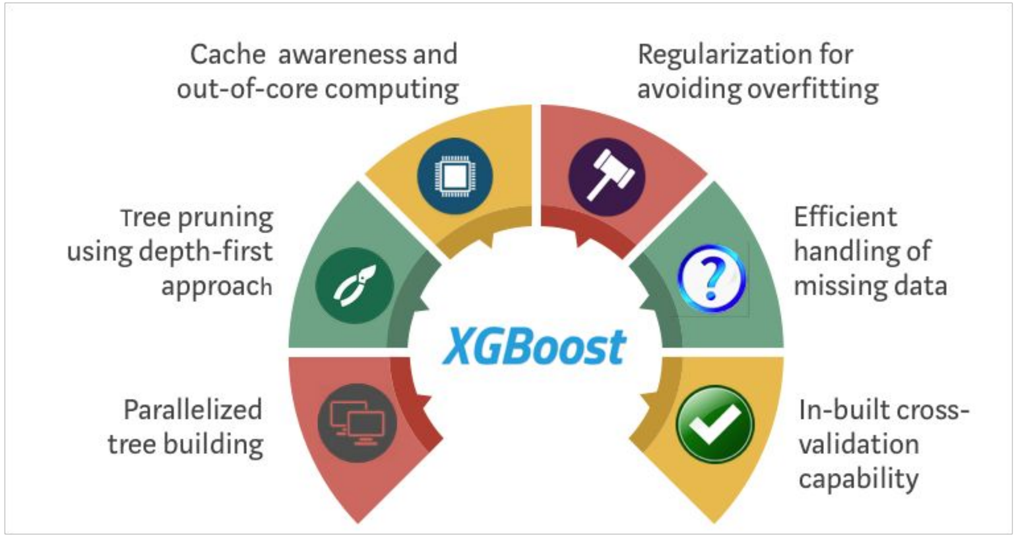


Gradient Boosting

- Gradient Boosting is an extension over boosting method.
- **Gradient Boosting = Boosting + Gradient Descent**
- It uses gradient descent algorithm for optimizing the given loss function.
- An ensemble of trees are built one by one and individual trees are summed sequentially.
- Next tree tries to recover the loss (difference between actual and predicted values).
- It can supports any **differential loss function.**

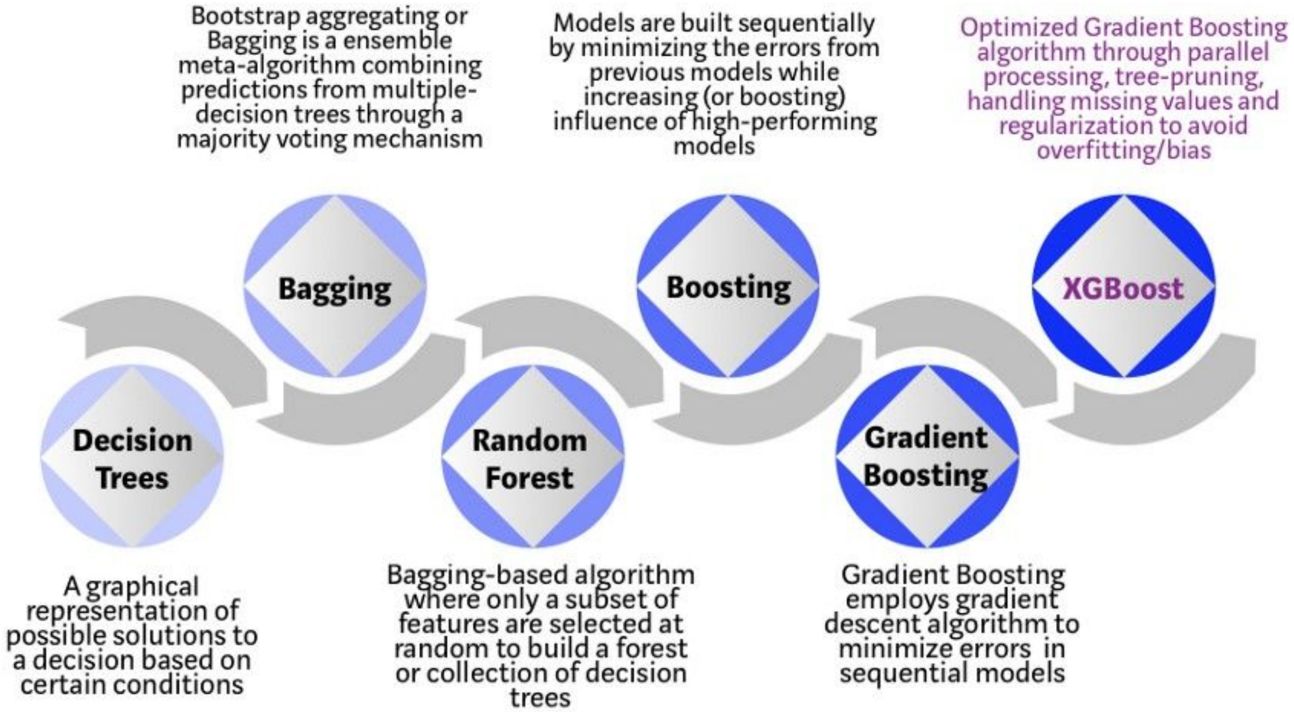
Extreme Gradient Boosting

- XGBoost is like gradient boosting on ‘steroids’!
- XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.


Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

—

Extreme Gradient Boosting


Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Extreme Gradient Boosting

System Optimization:

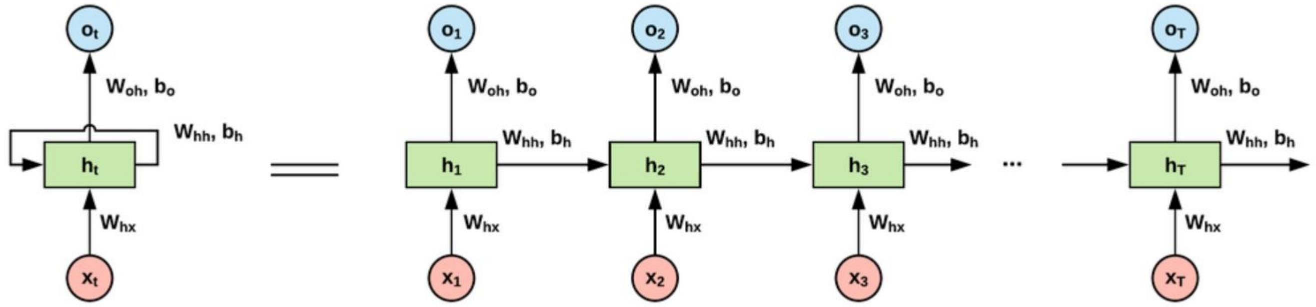
- **Parallelization:** XGBoost approaches the process of sequential tree building in parallelized approach using all of the CPU cores during training.
- **‘Out-of-core’ computing:** optimize available disk space while handling very large datasets that do not fit into memory.
- **Cache Optimization** of data structures and algorithm to make best use of hardware.
- **Tree Pruning:** XGBoost uses ‘max_depth’ parameter instead of negative loss criterion, and starts pruning trees backward. This ‘depth-first’ approach improves computational performance significantly.

Extreme Gradient Boosting

Algorithmic Enhancements:

- **Regularization:** It penalizes more complex models through both LASSO (L1) and Ridge (L2) regularization to prevent overfitting.
- **Sparsity Awareness:** XGBoost handles different types of sparsity patterns in the data as well as missing values more efficiently.
- **Weighted Quantile Sketch:** XGBoost employs the distributed weighted Quantile Sketch algorithm to effectively find the optimal split points among weighted datasets.
- **Cross-validation:** The algorithm comes with built-in cross-validation method at each iteration, taking away the need to explicitly program this search .

Recurrent Neural Networks



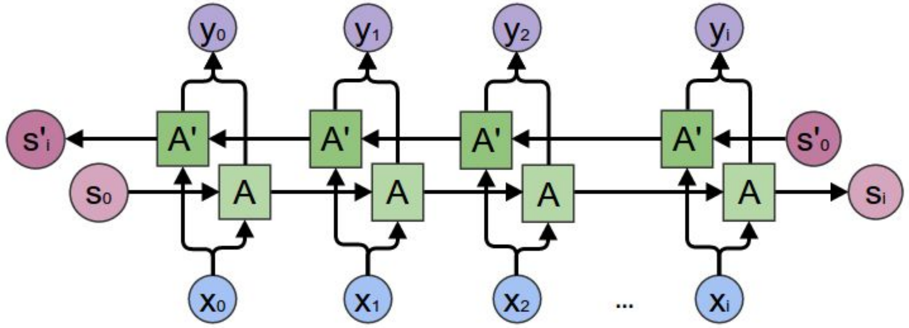
- They are the Neural Networks where the output from previous step are fed as input to the current step.
- Have a **hidden layer** which remembers some information about a sequence.

LSTM

- Vanilla RNNs face **vanishing gradients**. To eliminate vanishing gradient problem and capture long term dependencies LSTM(Long Short Term Memory) models were developed.
- LSTMs use three **gates** :
  - Input Gate
  - Output Gate
  - Forget Gate
- With the help of these **gates and Cell stat**e LSTMs are able to preserve past information of sequential data.

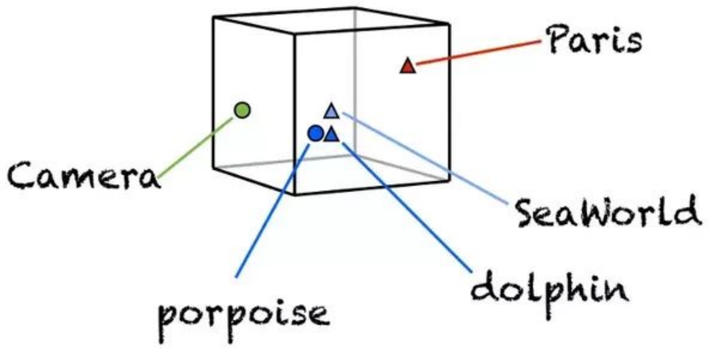
Bi-LSTM

- Bi-LSTM **couples two LSTM** such that one LSTM is trained on sequence in direction of increasing index and the other is trained in the direction of decreasing index.
- Using the hidden states of both LSTM, Bi-LSTM uses **both past and future context** to give the output.



Word Embeddings

- Learned **representation** for text where words that have the same meaning have a similar representation.
- Low-dimensional and **dense**. Number of features is much smaller than than size of vocabulary.
- Vector values are learned in a way that resembles a neural network.

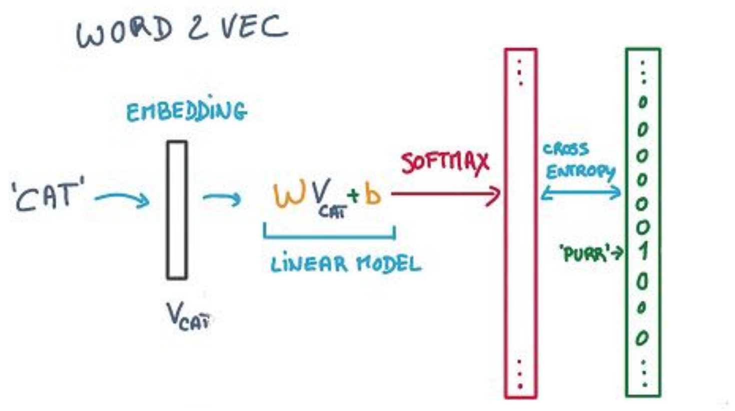


Pretrained Embeddings

- Unstructured textual data needs to be processed into some **structured**, **vectorized** formats which can be understood and acted upon by machine learning algorithms.
- Obtained by **unsupervised training** of a model on a large corpora of text.
- Captures the **syntax** and **semantic** meaning of a word by training on a large corpora.
- Increases **efficiency** of the model.
- Several pre trained word-embeddings are present namely, Word2Vec, FastText, Glove.

Word2Vec

- Statistical method for efficiently learning a standalone word embedding from a text corpus.
- King - Man + Woman = Queen
- Two popular models are **SkipGram** and **CBOW**.



Word2Vec

CBOW Model

- Learns the embeddings by predicting the current word based on its context.
- Trains faster and works better for frequent words.

SkipGram Model

- Learns by predicting the surrounding words given a current word.
- Works well with small data and can represent rare words.

FastText

- Represents each word as **n-gram** of characters to allow the embeddings to understand its prefixes and suffixes. E.g., the word vector, *apple*, could be broken down into separate word vectors units as *ap*, *app*, *ple*.
- Uses **skip-gram model** for learning embeddings.
- Works well on rare words as the word in broken down in n-grams to get its embeddings.
- Even works well on the words not present in the vocabulary because the n-gram character vectors are shared with other words.

Glove

- Stands for **Global Vectors**.
- An unsupervised learning algorithm for obtaining vector representations for words.
- Learns by constructing a **co-occurrence matrix** that counts how frequently a word appears in a context.
- Performs better on **word analogy** and named **entity recognition**.
- Does not rely only on local statistics (local context information of words), but also **incorporates global statistics** (word co-occurrence) to obtain word vectors.

Receiver Operating Characteristic

- An ROC curve is a graph showing the performance of a classification model at **all classification thresholds**.
- This curve plots two parameters:
    - True Positive Rate (**TPR**) : Fraction of positive data points classified as positive.

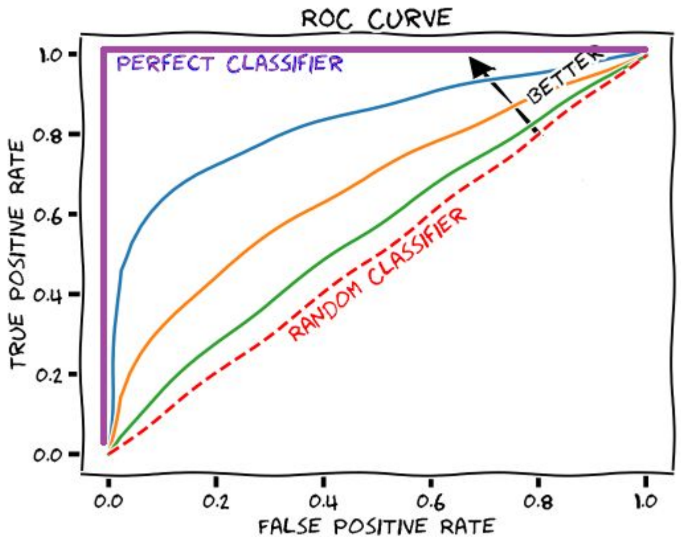
$$TPR = \frac{TP}{TP+FN}$$

    - False Positive Rate (**FPR**) : Fraction of negative data points classified as positive.

$$FPR = \frac{FP}{FP+TN}$$

Receiver Operating Characteristic

- An ROC curve plots **TPR vs. FPR** at different classification thresholds.
- Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

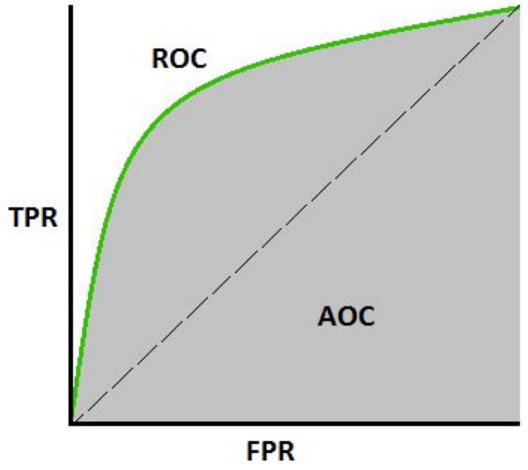


Area Under the ROC Curve

- AUC measures the **entire two-dimensional area** underneath the entire ROC curve from (0,0) to (1,1).
- More the area under curve better the classifier!
- AUC provides an **aggregate measure of performance** across all possible classification thresholds.
- AUC can be interpreted as the **probability** that the model ranks a random positive example more highly than a random negative example.
- AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

Area Under the ROC Curve

- AUC is desirable for the following two reasons:
  - AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.
  - AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.



Results and Comparison

Model	Mean AUC_ROC Score
Support Vector Machines (Binary Relevance)	0.66
Support Vector Machines (Classifier Chains)	0.67
Logistic Regression (Binary Relevance)	0.73
Logistic Regression (Classifier Chains)	0.76
Extra Trees	0.93
XGBoost	0.96
LSTM without pretrained embeddings	**0.97**
LSTM with FastText embedding	0.96
LSTM with Glove embedding	0.88
LSTM with Word2Vec embedding	0.85

Conclusion and Further Works

- Classical Models like SVM and Logistic Regression fail to achieve high AUC_ROC score.
- Classifier Chain method has a slight edge over Binary Relevance.
- Tree based ensembling methods and Recurrent Neural networks can achieve a very high AUC_ROC score.
- Transfer learning reduces the execution time but does not help with performance.
- Further Improvements can be achieved by experimenting with complicated neural networks and use of state of the art Transformers.

Thank You.