PARTLY SUNNY WITH A CHANCE OF HASHTAGS
WEATHER AND SENTIMENT PREDICTIONS BY ANALYSING TWEETS

APOORV AGARVAL ASMITA NEGI NAMITA MHATRE

TEAM 39

INTRODUCTION

- There are approximately 200 million active users on Twitter and 400 million tweets are posted on a daily basis. With such a large volume of tweets there is huge potential for gathering information just about anything.
- This challenge involves analysis of tweets and determining whether it has a positive, negative, or neutral sentiment, whether the weather occurred in the past, present, or future, and what sort of weather the tweet references.
- Let's consider a tweet:
  - **@mention another storm? At least we're getting rain. How bad is this one?**
  - This tweet was recorded from location Austin in state Texas.
  - The sentiment S2 got the maximum score of 0.795, thus rating the sentiment on this tweet as negative
  - The value of w1 came out to 1, which means that the tweet is talking about the weather of the same day.
  - As the keywords are picked from the tweet, the classifier predicted the kind of weather to be "Stormy" and "Rainy".

MOTIVATION

- With this huge amount of data available, by applying suitable machine learning techniques it is possible to putthis data to use.
- Using this data, we can predict the sentiments, and the kind of weather for a particular tweet. Also, we have an added feature where the user inputs the location he wants the information about and after an analysis of the output labels of training and testing data, we can give a comprehensive information about the weather of that location.
- Thus, we will be doing a quantitative analysis and will be giving out a report which states the type of weather in that location for a given time.

EVALUATION SETUP

- To evaluate the performance of a classifier we can either calculate the accuracy, precision-recall, the running time of the algorithm etc.
- The Root Mean Squared Error ("RMSE") is used to measure the accuracy for our classifier:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}}$$

- Where:
  - n is 24 times the total number of tweets.
  - p_i is the predicted confidence rating for a given label.
  - a_i is the actual confidence rating for a given label.
- The RMSE gives the weighted error between the predicted and the actual labels.
- So, the lower the value of RMSE, better the classification algorithm.
- In the Kaggle challenge, the winner had an RMSE value of 0.14314.
- Deep learning was used in that model with a combination of classifiers.

ALGORITHM AND CLASSIFIERS

- This is a multi-label classification problem where we have 24 different types of labels which belong to three different categories:
  - Sentiment of the tweet
  - "when" : the time to which the tweet refers to (past, present, future)
  - Kind of weather the tweet is talking about.
- While sentiment and when can have only one type of label for a tweet, kind can have many labels assigned to one tweet.
- Following shows the labels of the three classes:

Sentiments:

S1 = "I can't tell" S2 = "Negative" S3 = "Neutral "

S4 = "Positive" S5 = "Tweet not related to weather condition"

When:

W1 = "current (same day) weather" W2 = "future (forecast)"

W3 = "I can't tell" W4 = "past weather"

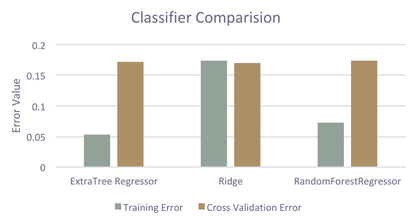
Kind of weather:

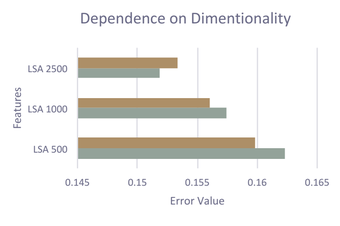
K1 = "clouds"	K2 = "cold"	K3 = "dry"
K4 = "hot"	K5 = "humid"	K6 = "hurricane"
K7 = "I can't tell"	K8 = "ice"	K9 = "other"
K10 = "rain"	K11 = "snow"	K12 = "storms"
K13 = "sun"	K14 = "tornado"	K15 = "wind"

- **Feature Extraction :** TFIDF Vectorizer
  - Using this vectorizer, we extract the features from our training set. We used the following parameters for extraction:
  - 10000 maximum features, Unicode accents only, analyzing words only, tokens with word length 3 or more, unigram and bigram baselines, and a snowball tokenizer. We can tune these parameters.
  - Snowball Tokenizer : The language people use in social media gives a lot of weightage to emoticons. Emoticons/Smileys play an important role in conveying sentiments. When we use stop words or some other type of feature filtration, as the emoticons are formed using punctuation symbols; these are filtered and are not counted as features. Same happens when we use a unigram model. A smiley **:)** is formed using a colon and a space separated closing brace. In unigram model, these will be accounted for separate features. Thus, we have to consider atleast bigram model to account for emoticons. Multi gram configuration not only provide weightage to just unigram but also to the context in which the message is being written. So as we are dealing with tweets which have a huge amount of emoticons, we are using this tokenizer to help us predicts the sentiment in a better way. A snowball tokenizer is supported in 19 languages and it implements different language specific flavours. This tokenizer helps extracting features from emoticons.
- **Feature Selection :** Multi task Lasso
  - The multi-task lasso allows to fit multiple regression problems jointly enforcing the selected features to be the same across tasks. The multi-task lasso imposes that features that are selected at one time point are select for all time point. This makes feature selection by the Lasso more stable.
- **Cross Validation :**
  - Cross validation is where we divide the data in a:b, use 'b' part for validation and rest for training. We have used 20% of the data for cross validation to improve the accuracy.
- **Classification :**
  - A **classifier** is a Supervised function where the learned (target) attribute is categorical. It is used after the learning process to classify new records by giving them the best target attribute (prediction). The target attribute can be one of k class membership.
  - **Ridge regression:** This model solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm.
  - **Random Forrest Regressor:** A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.
  - **Extra Tree Regressor :** This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. It gives a very good accuracy for multi-label classification.

RESULTS AND ANALYSIS

- The best value we obtained for the Root Mean Squared Error is : 0.15998.
- This best configuration uses a TFIDF vectorizer (10,000 maximum features, unicode accent, 2,9 gram baseline with a character analyzer) with LSA which uses truncatedSVD for reducing the dimensions of the features. It uses a ridge classifier for the multi-class classification.
- Different feature extraction parameters and different classifiers were used and following graphs show the comparisons.





- Following is the demo where we predict the kind and sentiment of an input tweet:
- **TWEET IS: SUNSHINE! It's been so long since ive seen you. Now if you could be warmer than 38 degrees right now I PROMISE I'll stop complaining**
- **'Sentiment' prediction: Positive**
- **'When' prediction: current (same day) weather**
- **'Weather' prediction: sun**

- Following is the demo of our model, where we predict the weather for a particular state:
- **STATE : Chicago**
- **NUMBER OF TWEETS : 500**
- **'When' prediction: current (same day) weather → Sun**
- **'When' prediction: past weather → Hot**
- **'When' prediction: future (forecast) weather → Sun**

CONCLUSIONS

- Different classifiers, feature extractors and selectors were used in plug and play format to learn their relative efficiency.
- For a model with multi-class classification and such a high amount of data, the Ridge classifier gave the best performance.
- Language on social media is not always standard. And the information is many time useless or unreliable. But with proper pre-processing and filtering; and a suitable machine learning model, we can turn this unrefined data into something very useful such as our weather prediction.
- Thus, we learnt how to actually apply machine learning in our day-to-day lives to get the best out of something trivial.