Learning Weather Indications from Tweets

Sanny Kumar, Anand Kelkar & Zubin John,
Department of Computer Science and Engineering,
The Ohio State University

April 21, 2014

Abstract

What can a machine learn from tweets about the weather? Adapted from the Kaggle Competition titled 'Partly Sunny with a Chance of Hastags', this project has been sourced by the CrowdFlower Open Library initiative. The task, put forth to about 200 teams was to learn weather,sentiment and time orientation from an annotated source and use this knowledge to build a system which can combine these 3 features to build an effective weather prediction system.We are provided a set of tweets related to the weather hand annotated by a panel of human judges as our data-set. The challenge here is to fuse feature selection using natural language processing and machine learning algorithms together to work on a an informal text corpus. Our team of three adapted this task to our purpose of applying our learning from class-room interaction to successfully predict multiple classes telling us if the weather is sunny, rainy, cloudy or maybe even a combination of the two. We succeeded in building a set of 8-classifiers using a multitude of methods, the combination of these learners performs ideally on the task of predicting weather from tweets.

1 Introduction

Social networking provides a great opportunity for people to voice their opinions on a number of topics. Aggregating social network data based on certain key words (hashtags in modern jargon) and analyzing them can provide unprecedented levels of information about any given subject. This has insightful applications in marketing - getting mass honest user opinions and input on a product without the need to contact them directly, and recommendation systems - showing users content they may be interested in. The Kaggle competition: Partly Sunny With a Chance of Hashtags aims to analyze tweets about the weather to determine several aspects of their content. Successful analysis of tweets in this category and the subsequent confidence scores for each label can be applied to other subjects and products, which can allow user opinions and desires to drive various industries, policies and decisions, and allow direct advertising toward a target market. This task itself is interesting because it provides the opportunity to extract different keywords from the same data to different ends. The tasks for this project is to predict the weather information from the given tweets. The basic approach to solve the problems is first select features to represent tweets and train a machine using a combination of methods on the given training set, and finally predict the confidence scores of 15 attributes on the test set.

In the world of social media, where text is not always strictly English, but rather a potpourri of slang, misspellings, acronyms, and the like, the problem is far more challenging. In this project, we use statistical learning techniques to predict a range of weather conditions from the extremely popular social media website, Twitter. Our dataset is a collection of tweets, each having been evaluated by human 'graders' and the main problem is that given a tweet, can we accurately predict how a human might interpret information concerning the weather? To this end, we construct and evaluate a collection of models utilizing support vector regression, linear regression, random forests, etc.
As already mentioned the above given problem has been adapted from the Kaggle Competition[1] repository. It is general belief that more people are connected to Social Media on the go everyday; a lot of folks would check their feeds before leaving home without even thinking of checking the weather. It would be highly convenient if Tweets could tell us the weather as well. Training a system to learn weather is one of the approaches to building intelligent automations in the meteorological field. Human predictions have a margin of error as they seek to gauge weather ahead of time. With computing being ubiquitous it is easier to relay information of live real-time events. The Weather prediction task from Tweets is not as straight-forward as it seems. Some of the research problems here would deal with eliminating ambiguity from Tweet interpreting its meaning and finally intelligent selection of features to present accurate results. As an example consider the two sentences:

- Dark clouds looming in the horizon; wet days ahead for sunny California!
- Unexpected heat wave a welcome reprieve from the frigid extreme. Brrrrrrr!

These sentences are a typical example of what can go wrong. While, the first is relatively easier to overcome with inclusion of context, the second case might offer more of a challenge. Tweets such as those can sometimes be dubious. How we intend to solve this is by including the location and date of the tweets. Armed with that information it is easier to conclude that Wisconsin is experiencing a brief spell of warmer weather; but Wisconsin being Wisconsin the weather can definitely not be categorized as `Sunny`,`Warm` or `Humid`.

2 Data

Source:Kaggle Competition, Partly Sunny with a Chance of Hashtags
As mentioned the data has been sourced for the competition by Crowdflower[2] Open Data Library. The data set consists of a training set and a test data set on which performance will be evaluated for the purpose of the competition. We however have used 5-fold cross validation for testing as the true classes for the test set have not been reported. All tweets are stored in comma separated value. The training file consists of the Tweet, its location and a Confidence value for each tag (sunny=0.6, rainy=0.9 and so on). Each tweet may have multiple tags associated with it as all the tags have been been reviewed by multiple raters and their might be a difference in opinion regarding the weather connotation

amongst the raters. As a result, the confidence scores accurately convey the correlation for agreement between two raters. Similarly the test data here also has confidence scores as predicted by the very same panel of raters. These values will be used in comparing performance of our system.

Ultimately, the goal is to train our model using information obtained from the training dataset, and then to try our model out on the provided test dataset to understand how well it performs on new data. The training set is simply a .csv file with 77,947 rows of data, 77,946 of which correspond to an individual tweet (the first row only contains column labels). In other words, our training set contains 77,946 unique tweets provided to us by the creator of the competition, most of which deal with the weather. Each tweet in the training set has a unique ID associated with it, along with the actual text contained in the tweet, the tweets location (a state in the U.S.), another more specific (and frequently inconsistent) sub-location (often a city in the U.S.), and finally, a series of confidence scores for each of the 24 labels associated with each tweet.

The final fifteen labels deal with the type of weather being referred to in the tweet i.e. snow, rain, hurricane, tornado, sun, clouds, etc. Each of the confidence scores provided in the training dataset are the result of human 'graders,' who manually went through each tweet and assigned a score to it in each of those three categories. The scores we see in the training dataset are simply weighted averages of the human-generated grades.

3 Approach

This task is interesting because it potentially allows for an innovative new alternative to satellites and cameras for a weather-predicting method: social media! It would be an interesting problem to see how accurate of a weather map can be drawn from tweets discussing the weather in a geographical area. A more practical use of this information can be to make recommendations to users. For example, if people are tweeting about how cold they are and their intuition saying its going to be a cold winter, it would be an ideal time for a clothing store to advertise a jacket or sweater or even better yet start preparing for larger sales this season. The advantage of this method to traditional meteorological approaches is that, it takes into account the feelings and thoughts of mass public into account rather than that of science. It doesn't matter if the weather is actually getting colder at the rate meteorologists predict; what matters is that do the people agree with the notion. This form of prediction caters to the audience flavor as an application rather than the scientific outlook.

For the task at hand there are a large number of pre-requisites befor we can succesfully implement a classifier. Some of them are pre-processing noisy data, enhancing features and selecting the best and finally the choice of the methods themselves. In the next few sections we deal with of these individual issues.

3.1 Preprocessing

Due to a large number of informal jargon and blend of different linguistic style twitter data is often ery nise. We have links, retweets(RT), hashtags and emoticons which deter proper linguistic understanding of the data. In light of the vast amount of information available in this informal context toold have to be developed to explicitly cater to this flavor of data.

One way to make this task easier is pre-processing. Pre-processing deals with cleaning and scrubbing up this data. The pre-processing that we have performed makes use of the shell tool `grep` to make pattern selection matching RT's and links and replacing and replacing them with null strings We have retained hashtags as they contain important information and may even contain unique strings which we need to retain. In a more stronger rule-based system it would be ideal to process emoticons using a dictionary and translate them into sentiment/polarity weights. However, due to the scope of this project we have refrained from this task.

Tokenizing is the next fundamental phase in our pre-processing. It is necessary to tokenize our data so that we may perform further pre-processing such as tf-idf vector initialization, POS tagging and bi-gram modeling for our purpose. There are various tools available online nowadays which are targeted towards performing tokenization and tagging n Twitter Data (see [3] and [4]). However the nature of these tools is for purposes slightly different from our own and since we don't really need state-of-the-art POS tagging, we will use the stadard NLTK inbuilt tokenizers and taggers[5]. We say that are needs are not restrictive as we will demonstrate later that POS tags do not perform as well as other features and hence it would not be wise to pursue such a system.

3.2 Feature Selection and Extraction

For our purpose we have targeted three different feature sets for our purpose. Following is a description of the them, however due to the superiority of one of the features we are more inclined to use it in our system a=in contrast to the other two.

3.2.1 TF-IDF

Among other natural language processing methods we use to extract features from the tweets is the TF-IDF (term frequency inverse document frequency) weighting scheme. Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

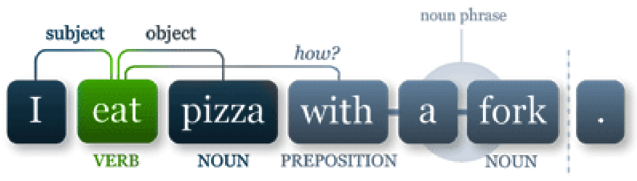
One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model. Our dataset contained 5,495 distinct words in total, resulting in 5,495 TF-IDF features for each tweet considering words which have minimum frequency of 10 in the overall corpus consisting of all the tweets. This score places weight on words in tweets that occur many times in that specific tweet but comparatively fewer times in other tweets. Since each tweet is very short, and so will not often contain repeated words, weight is given to words that simply do not occur often in the dataset overall, but happen to occur in a given specific tweet. The vast percentage of these scores for each tweet were, of course, zero, since tweets are given TF-IDF scores of zero for words they do not contain, and each tweet contains very few words. To extract and format these features appropriately we used a Python script, which is included in the appendix. The scriptt uses the Sci-Kit Learn, Scipy, and Nltk packages to calculate all of the TF-IDF scores from the tweets, and represent them in a suitable sparse matrix.

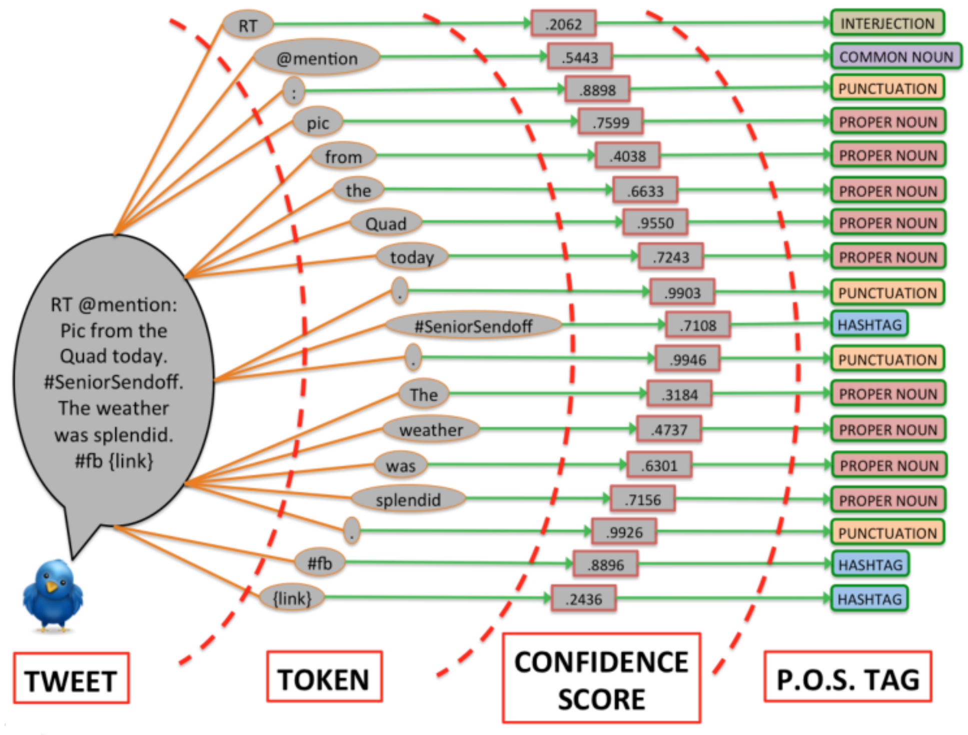
3.2.2 POS Tags

The second feature set we extracted from our training tweets was the Part-of-Speech (POS) we obtained from each word in a given tweet. The POS a word takes on does not always tell much alone, however, in congregate, this information has the potential to correlate powerfully with certain sentiments, which is obviously of great importance to us. For instance, an example correlation we may want to investigate is how likely a person is to be speaking negatively about the weather, given that the person uses a certain number of adjectives, nouns, or punctuation marks in his or her language. Note that it may not even matter what the parts of speech are in this case, only that the individual used a certain number of one or another. This very well could be enough information for us to make a guess as to whether the person felt positively or negatively about the subject of the tweet. We might also be interested in understanding how retweets (where one tweet responds to another user's tweet), links, ASCII emoticons, and the like are associated with sentiments. Using the POS feature space, we hope to unearth some interesting correlations between sentiments and parts-of-speech.

Our experiments proved that while POS tags may be astute and an excellent feature space to explore the sentiment associated the tweets they are not very informative for classifying weather conditions. However, they are an excellent possible extended feature space and hence we have used them nonetheless.

The illustrations below follows the POS tagging tasks on structured text data and contrast it with POS tagging on tweets. The difference is quite obvious.





3.2.3 Bi-grams

We use the NLTK to create a dictionary of bi-gram sequences for training a Naives-Bayes classifier. Bi-grams perform quite well in the domain of text classification and word sense disambiguation tasks and we would like to see just how well it can be adapted to our feature-based weather classifier. Context enabled classification is normally a good move, however again due to the nature of our rather noisy data it is difficult to predict how this would play out without any experimentation.

After training we reported the most informative features from two of our feature-sets; the Twitter POS tags and the TF-IDF Vector. The results have been recorded in the table below.

TF-IDF	POS Tags
URL	#
Emoticon	activation
$(Numeral)	advisory
Punctuation	afternoon
@Mention	expected
Interjection	forecast
Adverb	glasglow
Verb Participle	scattered
Existential, Predeterminers	storm
Hashtag	tragic

3.3 Algorithms and Methods

This section gives a brief description of the algorithms used in building the prediction system. For the first six algorithms we have used the TF-IDF vector space as features, whereas in the final ttwo implementations we have POS tags and bi-grams respectively.

3.3.1 Support Vector Regression

Support vector regression (SVRs) have been shown to be par- ticularly effective on regressions involving text data. SVMs work by finding a hyperplane for the data that maximizes the sum over all of the Euclidean distances (although the concept of distance can be generalized) from each data point to the hyperplane. This is called the optimal separating hyperplane. SVMs are consider- ably less prone to erroneous values in features, as we make a soft fit penalizing for every data instance falling within the margin. We began by training a number of models with radial, polynomial, and linear kernels in addition to various margin values of C between 100 to 0.001. Ultimately, the model that performed best on the validation set was the model with a radial (RBF) and a margin value of 1.0. We used the Sci-Kit Learn package in Python to construct and evaluate our SVR models.

3.3.2 Linear SVC

Given data points of multiple classes in a data set, the basic aim of support vector machines is to maximize the distance between the hyperplane separating the data point of the different classes and the closest points of each class to this plane. This is because, the larrger the margin that we have, the lower the generalization error ofthe classifier.

3.3.3 Linear Regression

Linear regression is one of the most commonly used algorithms that exploits lin ear relationships between the predictor and the target variable (sentiments, in our case). Linear regression makes use of the objective function that determines the linear line that passes through the data subject to minimum sum of square errors over all data points. Surprisingly, linear regression consistently showed better per- formance against other complex algorithms. In linear regression, we try to predict the value of a variable y basedon the value of a known variable x know as the predictor variable. Pictorially, linear regression basically is finding the best fitting straight line through a set of data points. The predictor variables areparameters or features which are selected for the dataset. Linear Regression gives us a way to predict the output (or class) of the data point based on its features. The difference between the predicted class and the actual class is the error of the prediction.

3.3.4 Gaussian Naive-Bayes

Sometimes in dealing with contiguous data, it is justified to assume that continuous valeus associated with each class are distributed according to a Gaussian distribution. It follows the Naives Bayes classification with the added assumption of the classes belongn=ing to a Gaussian distribution.The Naives Bayes is a simple classifier based on strong independence assumptions i.e they assume that the value of a particluar feature is independent of the presence or absent of other features.The NB classifier uses the method of maximum likelyhood estimation and for most tasks can be trained efficently in a supervised setting. Despite their naive design they however work pretty well in real world settings.

3.3.5 Lasso

The Lasso is a special category of linear models that is used with the assumption that only few of the all the features provided are responsible for change in the target variable Y. Since we are using a large number of features generated from TF-IDF, we speculate that many of the features might not be actually responsible for change in y. Lasso is trained with L1 prior as 'regularizer', while training loss or the empirical risk is same as that of the linear regression.

3.3.6 Ridge Regression

A drawback of linear regression is that it suffers from low bias but high variance.Alternatively, with a large number of predictors, it can be helpful to identify a smaller subset of important variables. Linear regression though does not make use of this. Ridge regression is similar to a least-squares approach which shrinks the estimated coefficients towards zero.A tuning parameter Λ controls the strength of the penalty term. With increase in Λ the bias increases but the variance decreases.As a result the correct selection of Λ is of import. Our code uses this tuning parameter at Λ=1 to get the best results. Our results show that the Ridge Regression achieves best results for our data-set.

3.3.7 Naive Bayes with POS Tags

The Naive Bayes Classifier we have used for this scenario is trained on POS tags as features. What this means is that each training sample is composed of a set of 43 features, each feature indicating the count of a particular POS tag in the tweet in question. As already mentioned above a good number of these features are bound to be zero, however the top most informative features gives an indication of which of the POS tags are good at uniquely identifying classes. On a general basis, POS tags are not ideal for a multi-class problem: if we were to identify sentiment or maybe time of occurrence we could assume the POS trained NB classifier to give a good estimation of classification however, in this case the classifier is biased as a result of over-fitting towards the most statistically significant class. We know that the data-set is bound to have unbalanced classes and hence it seems like a bad idea to use POS tags as features in the scenario. The tests results do indeed indicate the very same as we expected and hence we would advise against using Naive-Bayes which does not scale well with dimensionality and still gives poor classification results.

3.3.8 Naive Bayes with Bigrams

The final implementation is pretty same in nature to the above. In both cases we are using a Linear Naives Base classifier to perform multi-class categorization. Bi-grams are pretty insensitive as features in the Twitter classification task. This is because of the existence of a lot of noise and named-entities which are equivalent to noise in this context. The system is bound to see a lot of named entities and @Mentions (a feature which contains a user identifier). In this case it would be void to create bi-gram features as many of the bi-grams in the test data will be unseen words termed as OOV or out-of-vocabulary in Natural Language Processing. With intuitive smoothing we can mitigate the effect of OOV, however we would still have terrible results for our classifier.

The main purpose of this classifier is to contrast which of the two, POS tagging or Bi-grams is worse for our weather classification task. It would be interesting to examine which classes (if any) are inclined to give positive results with either of the two implementations.

3.4 Evaluation

The main criteria to evaluate the performance is the Root Mean Square Error:

$$\sqrt{\tfrac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y_j}\right)^2}$$

The Root Mean Square Error is efficient way to measure the performance of our machine, since the core tasks for this project is predicting the confidence score for attributes. Thus, by measuring the difference between the actual confidence value and the predicted confidence value, it is possible to evaluate our machine correctly. From Kaggle, we have taken the train set and test set. However, test set does not include any confidence values and therefore cannot be used for the local training. Therefore, we used K-fold cross validation on the train set to evaluate different approaches and methods to pick the best approach. We have selected 5 fold cross validation as a good pay-off between time complexity and accuracy.

4 Results and Discussion

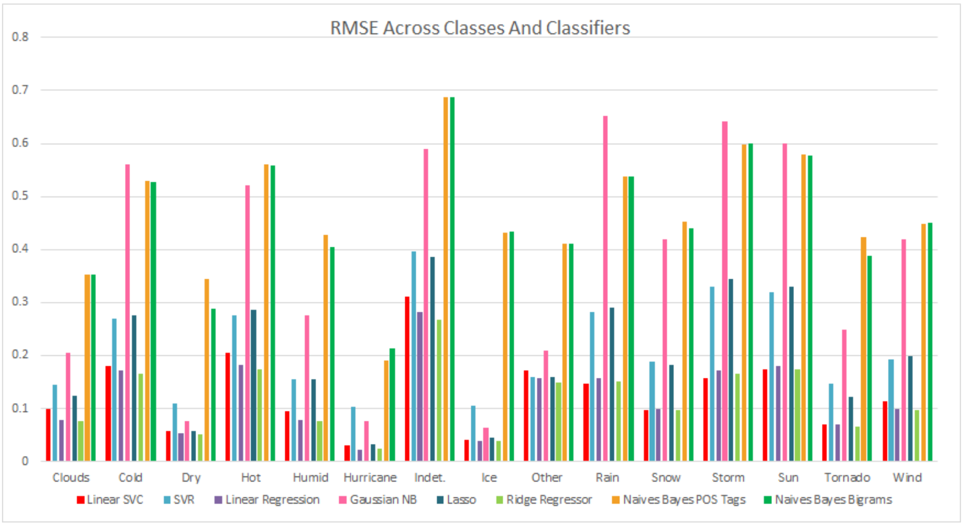
The results are quite positive for the task. Several of the methods implemented succeeded in giving us error rates of lesser than 15%. The given table reports the error using root mean square metric that we had earlier defined.

	SVR	Linear SVC	Lin. Reg.	Gaussian NB
Clouds	0.144809979	0.098948809	0.077591482	0.205220549
Cold	0.26858048	0.180281076	0.172433098	0.561049482
Dry	0.108410582	0.056292169	0.053012555	0.076688836
Hot	0.276397113	0.205956931	0.182353818	0.5212719
Humid	0.155921801	0.095650849	0.077923717	0.275146973
Hurricane	0.103948004	0.030612349	**0.022634621**	0.076314851
Indet.	0.397102722	0.31021338	0.281247235	0.588957401
Ice	0.105821107	0.041320885	0.038127718	0.063301835
Other	0.159702972	0.171356957	0.156252839	0.208734128
Rain	0.281361409	**0.145830951**	0.157450927	0.652121193
Snow	0.188641323	0.097327669	0.099015438	0.418907147
Storm	0.329369443	**0.156770617**	0.17231161	0.642425397
Sun	0.320062298	0.173691363	0.180537483	0.600500561
Tornado	0.146159668	0.070582036	0.070514844	0.249204142
Wind	0.192871073	0.112808658	0.098704185	0.419029549

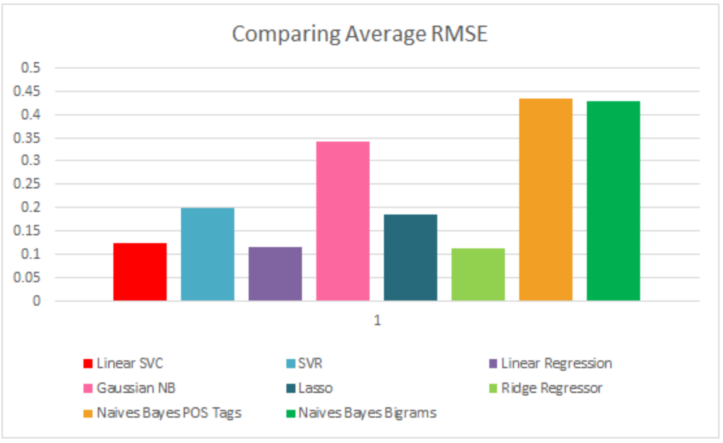
	Lasso	Ridge	NB-POS	NB-Bigrams
Clouds	0.124863632	**0.076483271**	0.35261527	0.35261527
Cold	0.276152382	**0.165228538**	0.529764949	0.527821947
Dry	0.056783086	**0.050108231**	0.343498987	0.288705056
Hot	0.286501905	**0.174338729**	0.560706451	0.558577544
Humid	0.154170351	**0.076806535**	0.428409615	0.404267636
Hurricane	0.033131001	0.023058388	0.191211185	0.213159131
Indet.	0.385332217	**0.267259625**	0.687377477	0.686662528
Ice	0.044058066	**0.037559781**	0.431021313	0.434228281
Other	0.15817709	**0.148572773**	0.410667656	0.410501049
Rain	0.289684805	0.151181514	0.537105469	0.536950864
Snow	0.181768501	**0.09586728**	0.453061492	0.440386157
Storm	0.344940499	0.165238139	0.597239298	0.600972554
Sun	0.329726005	**0.172764775**	0.580062127	0.576967073
Tornado	0.122457792	**0.066594427**	0.42318159	0.387341969
Wind	0.198303914	**0.096556588**	0.449054092	0.449357059

Table 1: Comparing result of all classifers against different classes.

From the results above we can see that Linear Regression performs well for the `Hurricane` class while the linear SVM classifier performs well for the `Rain` and `Storm` classes. For all the others the Ridge Regressor outperforms all the other classifiers. We hence built an hybrid system which picks the best classifier for each class from 5-fold cross validation training and used this collection of classifers to get an average classifiers RMSE score of *0.116891783* after training and cross-validation. The graph below allows us to visualize the performance of each classifier for the classes.



Another interesting visualization compares the performance of each classifier averaging the RMSE function over all the classes. We combined 3 different methods linear SVM classification,linear and Ridge regression. Then, we submitted the test set on Kaggle and the result for this system on the test set came out be *0.16139* RMSE. Please note that the top RMSE on the leaderboard is **0.14314**.



5 Conclusion

In conclusion we would like to add, that we observed a trend with classification that gives us reason to believe that the relation between the classes is linear. As a result we observe that linear techniques tend to give better performance. The project gave us a good platform to go beyond class lectures and explore linear regression implementation in detail. Another facet of this project was the opportunity to apply our learning in to building a real-time system that could have a strong impact on social computing. We imagine a large number of applications of building such large-scale predictions systems and performing text analytics on data, in the modern scenario. We have made the connection of this system with sentiment analysis on various occasion and once again say that the system could be trained with sentiment annotated data to perform perhaps equally well for that task. Hence, the system is quite flexible and data-intensive as it learns patterns in any kind of data to perform classification based on training. It is a required functionality of such supervised systems to scale-well with data and be adaptive. we are happy to report that our system performs well in these aspects.

References

[1] http://www.kaggle.com

[2] http://www.crowdflower.com

[3] http://github.com/aritter/twitter_nlp

[4] Gimpel, Kevin, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. "Part-of-speech tagging for twitter: Annotation, features, and experiments." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pp. 42-47. Association for Computational Linguistics, 2011.s

[5] http://www.nltk.org

[6] E. Riloff and J. Wiebe. Learning extraction patterns for subjective expressions. In *Proc. of the International Conference on Empirical Methods in Natural Language Processing* (EMNLP), Sapporo, Japan, 2003.

[7] Large-Scale Machine Learning at Twitter, Jimmy Lin and Alek Kolcz, *Twitter, Inc*

[8] Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features. Springer*, 1998.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Ma- chine learning in Python. *Journal of Machine Learning Research,* 12:2825–2830, 2011.

[10] Owen Phelan, Kevin McCarthy, and Barry Smyth. Using twitter to recommend real-time topical news. *In Proceedings of the third ACM conference on Recom- mender systems,* pages 385–388. ACM, 2009.