Coursework 1 - Team WeatherCasters

Data Mining Module (2017/18)

Chaipichet Palotaitakerng[*]
Student Id: 29125375
cp2n17@soton.ac.uk

Phisit Srirattanawong[†]
Student Id: 29764904
ps4y17@soton.ac.uk

Nutthika Rattanakornkul
Student Id: 29437431
nr1n17@soton.ac.uk

Aude Buchmuller
Student Id: 29711886
aamb1n17@soton.ac.uk

Thanadon Fuengworatham
Student Id: 29392802
tf2n17@soton.ac.uk

Susmita Gangopadhyay
Student Id:29861322
sg4g17@soton.ac.uk

ABSTRACT

This paper presents text mining on a dataset from Kaggle competition "Partly Sunny with a Chance of Hashtags". The dataset is related to weather. There are tweets in the dataset with twenty-four labels for each tweet. Each label has value lying between the range of zero to one. The tweets are pre-processed using NLTK library and the cleaned dataset is transformed to vector using Doc2Vec method. Ridge regression and LightGBM method are applied for the prediction of outputs. The results show that the dataset has imbalanced data and LightGBM models perform better than Ridge regression.

KEYWORDS

Data mining, Predictive model, Doc2Vec

ACM Reference Format:
Chaipichet Palotaitakerng, Phisit Srirattanawong, Nutthika Rattanakornkul, Aude Buchmuller, Thanadon Fuengworatham, and Susmita Gangopadhyay. 2018. Coursework 1 - Team WeatherCasters: Data Mining Module (2017/18) .

1 INTRODUCTION

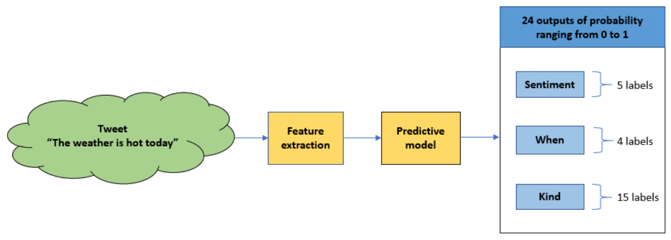


Figure 1: Overview of the whole process

Text mining, sometimes referred as "text analytics" is one way to make qualitative or "unstructured" data usable by a computer. In this paper we have taken up a text mining challenge, tried to perform feature extraction on its data and implement various machine learning techniques to interpret the result. The topic of our project is a Kaggle competition "Partly Sunny with a Chance of Hashtags". The main idea of this challenge is to analyse a set of tweets related to weather. We had to analyse tweets and determine whether they have a positive, negative, or neutral sentiment, whether the weather occurred in the past, present, or future, and what sort of weather the tweets reference to. The challenge is a regression problem and we are trying to predict the probability of each of the twenty-four outputs lying between the range of zero to one.

[*]Team Leader
[†]Deputy Team Leader

MSc Data Science, Coursework, Southampton, UK
2018.

The available data was first cleaned and then subjected to feature extraction. The extracted dataset is used to train our predictive models which helps determining the value of the twenty-four labels for unseen tweets.

2 AIMS AND OBJECTIVES

The aim of this project is to perform a series of experiments on the Kaggle problem about weather to build a set of predictive models. The models have to predict the twenty-four outputs which have value lying between the range of zero to one. Root-mean-squared error (RMSE) is the metric used to evaluate the predictive performance.

The objective of this project is to achieve the following outcomes: (1) cleaned dataset using pre-processing techniques for the English text, (2) vectors of the tweet using Doc2Vec method, (3) predictive model using Ridge regression method and LightGBM method, (4) analysis of the experimental results, (5) comparison between predictive models, and (6) discussion and reflection about the results and dataset.

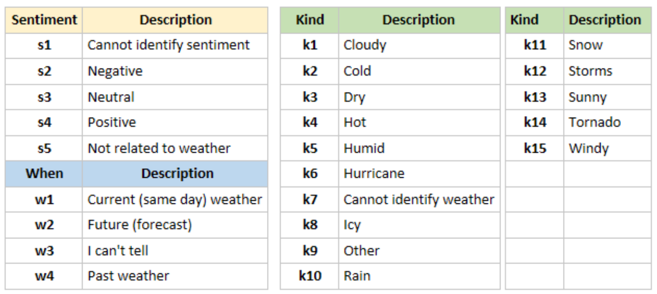
3 DATASET

The data provided in the Kaggle competition have been gathered from Twitter and stored in the open data library of CrowdFlower.

Link for the Kaggle competition:
https://www.kaggle.com/c/crowdflower-weather-twitter

3.1 Training set

Data in the training set was classified by CrowdFlower .



Sentiment	Description	Kind	Description	Kind	Description
s1	Cannot identify sentiment	k1	Cloudy	k11	Snow
s2	Negative	k2	Cold	k12	Storms
s3	Neutral	k3	Dry	k13	Sunny
s4	Positive	k4	Hot	k14	Tornado
s5	Not related to weather	k5	Humid	k15	Windy
When	**Description**	k6	Hurricane		
w1	Current (same day) weather	k7	Cannot identify weather		
w2	Future (forecast)	k8	Icy		
w3	I can't tell	k9	Other		
w4	Past weather	k10	Rain		

Figure 2: Description of 24 labels categorized in 3 groups Sentiment(S), When(W) and Kind(K).

The training data that is provided has the following form:

- Tweet:
  - Message: the original text from a tweet
  - Location: State, town
- Labels: as described in Figure 2

- s: likelihood value for each sentiment classification (+ve,-ve, neutral) and related to the weather or not (Yes, Not sure)
- w: likelihood value for each time frame category: past, present, future or not sure
- k: likelihood value for the weather state: clouds, cold, dry, hot, humid, hurricane, ice, not sure and other values

The training set contains 24 labels in total comprising of the three categories: sentiment, when, and kind. Human raters can choose only one label for the "sentiment" and "when" categories, but are allowed multiple choices for the "kind" as shown in Figure 2.

3.2 Test set

The test set is similar to the training set. Our goal was to predict a confidence score of all 24 labels for each tweet in the test set.

3.3 Imbalanced data

The labels of this dataset contains many zero values. Figure 3 shows the sample of imbalance in positive sentiments label. This leads to imbalanced-data problem which makes the machine learning model difficult to predict an accurate result which will be discussed in the next sections.

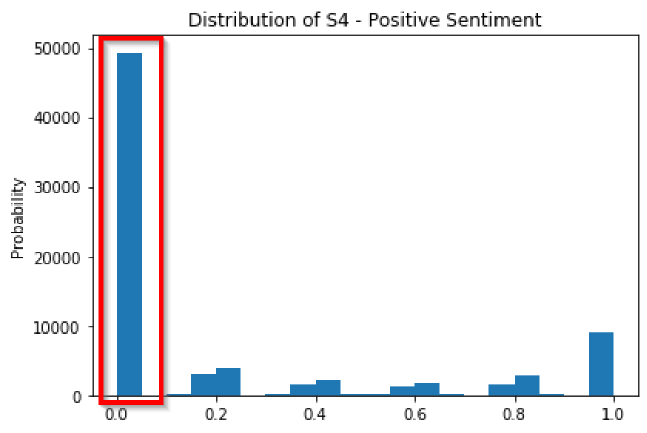


Figure 3: Histogram of positive sentiment output(S4) showing imbalance in the data

3.4 Noise in the dataset

Since each tweet was reviewed by multiple raters, some amount of disagreement on the labels were present. The confidence score accounts for two factors - the mixture of labels the raters gave a tweet and the individual trust of each rater. Since some raters are more accurate than others, they count more in the confidence score. This "unknown trust" issue was a source of noise in the problem and was taken into account into our analysis. The raters could choose only one label from the "sentiment" and "when" categories, but multiple choices for the "kind". The result was that confidences for the "sentiment" and "when" categories summed up to one whereas the sum of the "kind" category did not always sum to one.

4 APPLICATION OF TECHNIQUES

4.1 Data cleansing and pre-processing

Data in each record comprised of tweet id, tweet text, and location (state name and city name). Tweet text is the main feature that was processed to predict the labelled outputs. We have used Natural Language toolkit (NLTK) for tokenization, stemming, parsing and processing of the tweets. In this project, we did not include the state and location variables as a significant input feature as the discussion from the Kaggle's competition [4] explained that location slightly decreases the score of Root Mean Square Error(RMSE). Many competitors argued that using these variables made the RMSE score worst.
Three datasets have been specifically cleaned for each category which were used to train corresponding predictive models.

4.1.1 *S Preprocessing for the sentiment.*

- Included the emoticons list - to capture the sentiment reflected in tweets i.e. :-), :-(, :O
- Removed English stopwords - to reduce noises in the tweet to effectively extract underlying sentiment.
- Included English negative modal verbs list. For example "This weather may not be bad to go outside" is positive sentiment but "This weather may be bad to go outside" is negative sentiment. Including the negative modal verb helps us to capture the real context of the tweets.
- Converted to lower case, Tokenized texts and emoticons.
- Lemmatized text to verb form

As a result we got a list of words that contains emoticons and negative modal verb words.

4.1.2 W Preprocessing for the When. The primary consideration of W preprocessing were words related to time in tweets (past, present, future or unknown). The processing methods are presented as follows:

- Tokenized tweet and converted to lower case.
- Included the adverbs and nouns list that is related to time.
- Included all verbs and Part of Speech Tagging (POS) that are verbs and modal verbs.

As a result we got a list of words that contains words of time.

4.1.3 K Preprocessing for the kind of weather. The primary consideration of K preprocessing are words that describe what kind of weather is described in the tweet. For k processing we had to do the following process:

- Converted text to lower case and tokenized tweets including the emoticons related to weather and punctuation as %,- for temperature figures i.e. -20C.
- Included all parts of speech which are verbs, adjectives, nouns and adverbs in the list - to capture different parts of speech that are related to types of weather.

A a result, we got a list of words that contained words and emoticons related to weather .

4.2 Feature extraction

The words that have related meaning were pre-classified into the same term. The terms were used as features (inputs) of the predictive model instead of directly using the words. Using the *Doc2Vec* model, the tweets were transformed into vector forms for analysis in the next step.

4.2.1 ***Doc2Vec Method****.* When it comes to text classification or clustering, bag of words is one of the most common fixed-length features method used in the analysis. However, this technique has two main weaknesses: it loses the ordering of the words which can result as having two different sentences seen as similar in representation, and it ignores the semantics of the words. To avoid these issues, we have chosen to use the Doc2Vec framework, also called Paragraph vector.[8] This unsupervised algorithm learns feature representations from texts like paragraphs or documents. The features are represented in a form of vectors called paragraph vector. Some of the main advantages of paragraph vector is that the vector inherit the context of the words.

Compared to Word2Vec that work on the prediction of words context, Doc2Vec is applicable to texts of any length and does not rely on parse trees. The Word2Vec is a neural network which consists of three layers, one input, one hidden layer and one output layer. The context-words are fed to the input layer and the output layer has the predicted probability to have the target word given that context-words. The Doc2Vec method works in a similar way to Word2Vec, but the Doc2Vec model has additional inputs; The additional inputs are the labels of the document which use a binary number 0 and 1 to represent that the words belong to which document.

The structure of Doc2Vec model is illustrated in Figure 4. As illustrated in the figure, the input Word Ids represent the context words in a sliding window surrounding the target word. In addition, the input Document Ids represent a document that has that words. The model is trained to predict the probability of having the target word appear with the context words. W_j and D_i the weights before the hidden layer. The weights are used as the numbers in an output vector of the model. The weights D_i are used to represent as a vector of a document i. The weights W_j are used to represent as a vector of a word j.

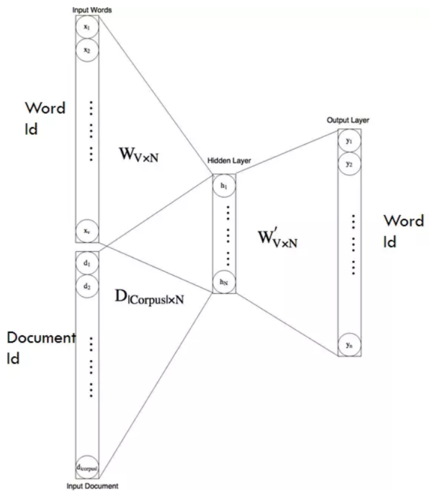


Figure 4: This diagram[1] shows an artificial neural network of the Doc2Vec model.

Doc2Vec models are created using the `Gensim` library [9]. Before the model was trained, a set of unique words extracted from the pre-processed dataset are assigned to the model as vocabularies. The model was trained to create a set of vectors from the training dataset iterated over a hundred epochs. The number of iterations could be increased to obtain a more accurate model but the training time would be expanded as well. The size of vector is set to be a hundred. This hundred numbers of a vector represent a hundred features that are used to predict the twenty-four outputs in the next step using the predictive models. After the completion of these steps, the trained model is used to *infer* a vector for each tweet in the test dataset.

4.3 Predictive methods

4.3.1 ***Ridge regression.*** Since ours was a *regression* problem, there were various ways to approach it. A simple linear regression was a good method to predict the outcomes. However, we choose to use Sparse regression as an alternative approach to try and find out the result since it works well with high dimensional data. The sparse regression includes both Ridge regression and LASSO but the one we choose is Ridge . The main reason is Ridge regression will not shrink parameter coefficients down to zero, unlike the LASSO method that shrinks some parameters out [7]. The reason to keep every parameter is that each parameter has an effect on a different context. In addition, the Ridge regression is the method that adds the regularizer term into the model which prevents it from overfitting.

The process started with using the preprocessed vectors of S, W and K to build three separate models. These models are used to predict sentiment, when and kind separately. Then the next step was to find the optimum loss function [6]. The loss function is constructed by an empirical estimate of the gamma-divergence with Ridge regularization and the parameter estimate is defined as the minimizer of the loss function. A good gamma will balance out the variance and bias of the model. To find the optimal gamma, we have done k-fold cross-validation on a range of gamma and calculated the root mean square error (RMSE). In the next step, we plotted the boxplot to find the optimal gamma which gives the lowest RMSE on cross-validation. The reason for using boxplot of RMSE rather than Ridge coefficient graph is because we have too many parameters to illustrate and putting all of them together would make the diagram too clumsy.

4.3.2 ***LightGBM.*** The imbalanced data problem that is explained in section three can lead to misleading analysis. The main causes of error in machine learning are due to noise, bias and variance. [10] Ensemble is one of the methods to tackle this problem. It will combine a set of weak learners and create a single strong learner that obtains better performance. It helps to decrease variance without increasing bias.Gradient Boosting is an example of this ensemble method.

LightGBM is a gradient boosting framework based on decision tree algorithms that is used for regression and classification problems.The goals is to minimize a loss function on the training data that could achieve accurate prediction for the unseen test data.[2] The benefit of using LightGBM compared to other Gradient tree boosting methods like XGBoost are higher efficiency, faster training speed, better accuracy and lower memory usage.

Most decision tree learning algorithm grows trees depth-wise (horizontally), while LightGBM grows tree leaf-wise(vertically).It will construct decision trees with initial weight. Then sort the features by the degree in trees in descending order. After that, it will check each feature in the order list. The weight is increased for misclassified class and decreased for correctly classified ones . It will choose the leaf with maximum delta loss to grow and merge correct feature to the existing bundle [5]. Comparing to depth-wise growth; this method can converge much faster. To find a better accuracy, these are some important parameters[3] :

1. learning_rate: shrinkage rate. The default value is 0.1. Assigning small learning_rate with large num_iterations could improve the accuracy.

2. num_iterations: number of boosting iterations. The default value is 100.

3. num_leaves: number of leaves in one tree. The default value is 31. It determines the complexity of the tree model. The larger the num_leaves, the better the accuracy, however the optimal number of num_leaves should be less than $2\hat{6}$(max_depth) to avoid over-fitting problems.

In the Kaggle's competition, the aim is to minimise the root mean square error (RMSE). First, we built the models from three different preprocessed vectors of S, W and K. Next we found the optimal loss function by setting up the test parameters as explained above. We started the tuning by performing K-fold cross-validation on different numbers of leaves with fixed other parameters as default and selecting the best one with lowest RMSE. Next, we performed the same method to find the optimal learning rate with the fixed optimal number of leaves. This tuning method finally gave us the optimal hyper parameters that will be used to predict the test data for submission in the Kaggle competition.

The plot comparing the tuned results will be presented and analysed in the next section.

5 RESULT ANALYSIS

5.1 The result of Ridge regression

Comparing all the results of various models with different gamma we have found the best result with RMSE around 0.30x as shown in the **Figure 5**. These models were quite different compared to other competitors. To investigate the difference, we checked the predicted output. The results show that our model will not predict 0. As a consequence, the RMSE might be high because of unbalanced output data which most of them should be zero.

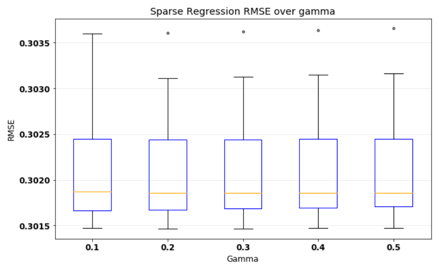


Figure 5: The box plot of RMSE threshold in different gamma

Since we used the vector with 100 parameters to predict 24 outputs, it's hard to pick the significant parameters. The reason is each parameter effect different output and many coefficients are close together.

5.2 The result of LGBM

With regards to the tuning process, we started by tuning the number of leaves. During the early process, an increase in number of leaves showed significant improvement in the model. This is because more leaves allow model to elaborately fit the data. However, at the optimal point of 180, increasing number of leaves did not significantly improve the model. Instead, at this stage, the model was turning to be over-fitting with the data which will cause lower accuracy in the unseen dataset. Thus, the optimal point of 180 was chosen from the point that showed significantly less improvement from the previous round.

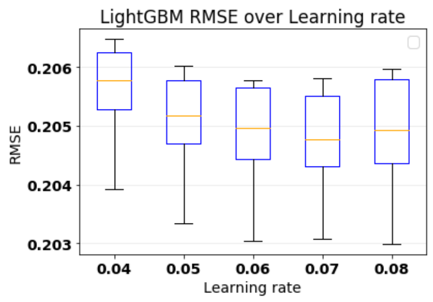


Figure 6: The box plot of RMSE in different learning rate with optimal number of leaves = 180

Next, learning rate was tuned with fixed number of leaves at 180. This initial learning rate starts from 0.01 until 0.1. As presented in Figure 6, the RMSE decreased to the minimum point at learning rate 0.07 then start rising again. This can be concluded that the optimal parameters are at learning rate 0.07 and number of leaves 180. With these optimal parameters, the model could predict all labels with average RMSE at around 0.2.

5.3 Model comparison

This section is to compare the predicted result between LightGBM and Ridge regression model.

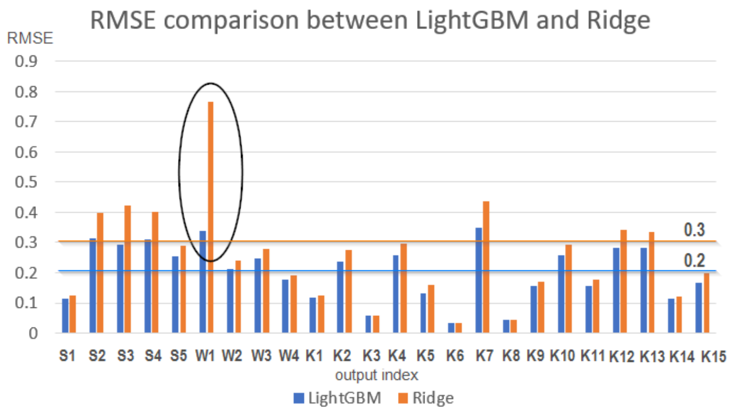


Figure 7: RMSE comparison between LightGBM and Ridge regression model - showing distinct difference on RMSE of W1 , which is probability that tweet mentions about *current weather*

According to Figure 7, overall, LightGBM model produced better result with average RMSE 0.2 which is more accurate compared to RMSE 0.3 from Ridge regression model. When comparing individual result on each label, it can be seen that LightGBM outperformed Ridge regression on most labels with outstanding difference on W1, which is a label for confidence that tweets mention about current weather.

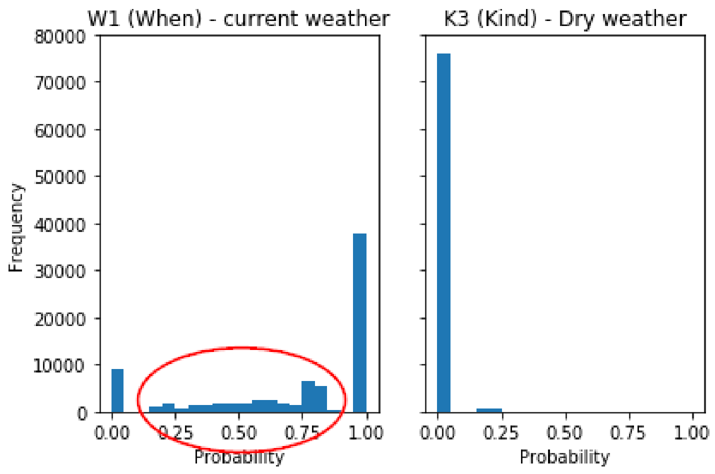


Figure 8: Comparison between distribution of W1 and K3 label where W1 shows more spreading data than K3.

To investigate this extreme difference, we looked at the distribution of all labels. In Figure 8, it can be seen that the data in W1 are more spreading compared to K3, which shows the smallest error predicted by Ridge regression. This implied that more spreading in the data caused higher error in Ridge regression model.

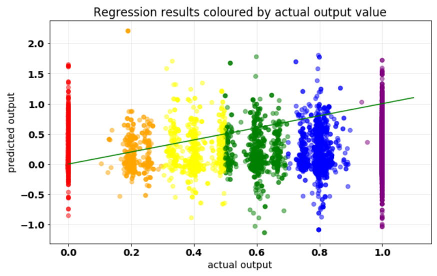


Figure 9: Predicted results from Ridge regression were out of scope of actual values between zero and one

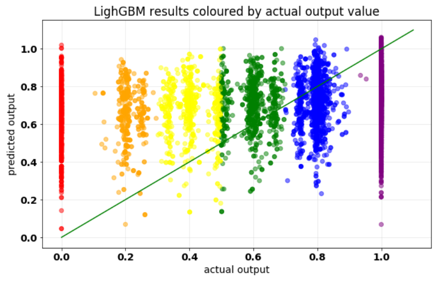


Figure 10: LightGBM predicted the output W1 in the scope between zero and one.

Next, we analysed the plot of predicted and actual results from both Ridge regression and LightGBM as presented in Figure 9 and 10. It is apparent that the result predicted by Ridge regression varied between -1 and 2, which was out of the actual value range. This can be implied that Ridge regression model tried to fit the spreading data points in the middle of W1 in Figure 8, but it could not properly fit the data. This might be because the data contained non-linear parts and Ridge regression, which is based on linear model, could not fit that data. As a result, the model was harmed from the attempt to fit all the data points, eventually, it produced the severe error as presented in Figure 7. For K3 label, Ridge regression performed well as it could fit the zero line where the model always predicted zero, while the majority of data were also zero. This resulted in the small error in K3.

On the other hand, LightGBM model could predict W1 in the proper range between 0 and 1 as shown in Figure 9. This means the model were not harmed by the spreading data points in Figure 8. Additionally, the RMSE results of this model were consistent across all labels as illustrated in Figure 7. Thus, the LightGBM could work well with this data without any distortion from spreading of data as affected in Ridge regression. This proofs that the ensemble technique could effectively handle the imbalance and tree-based algorithm used by LightGBM is suitable for fitting this non-linear data.

6 REFLECTION

In the analysis section, the predictive results suggest that the model perform well to predict some output values. However, the average error is still quite high. This section reflects about the problem and the applied techniques based on the dataset and the experimental results.

6.1 Similarity of the tweets

To gain a better understanding about the dataset, this section demonstrates of the dataset using multi-dimensional scaling (MDS) technique to map the vectors of tweet into the two dimensional visualisation. The visualisation uses colours to show the range of output for each tweet. Scatter plot is used to visualise the dataset. In this visualisation, the distance between the position of the tweet is calculated from the *cosine similarity* between the them.

There are (1) actual output value which is the label for each tweet and (2) predicted output value which is the predicted results using the predictive model. While the actual output value s are in a range from zero to one, most of the predicted values are also in a range from zero to one with some tweets that have the predicted value below zero. Since there are twenty-four labels for each tweet, only one output, label *s4*, is illustrated in the visualisation as an example; The label *s4* represents the probability of the positive sentiment for a tweet. There are six colours used to represent the actual output values as shown in the legend in **Figure 11** and seven colours used to represent the predicted output values as shown in the legend in **Figure 12**.

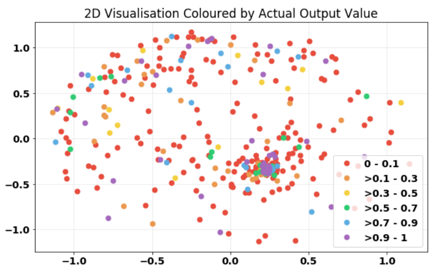


Figure 11: This scatter plot shows the actual value of output using the colour of data point. Each data point represents each tweet in the dataset. There are six colours for the actual output value. Four-hundred sample tweets are visualised.

As illustrated, the tweets that have different actual output value stay close to each other in the plot because they are quite similar to each other in term of the *cosine similarity*. The plot also shows the imbalance of output values which very frequently has the value in the range around zero (red colour) more than the other range of value. This similarity of the tweets is one of the factors that make it difficult to predict the value accurately.

The LightGBM predictive model is used to predict the output values and the results are illustrated in the following scatter plot.

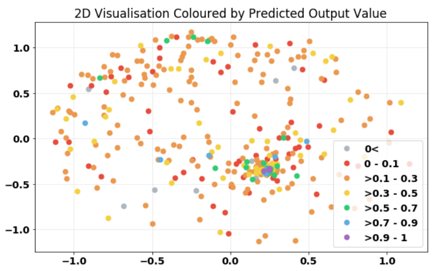


Figure 12: This scatter plot shows the predicted value of output using the colour of data point. Each data point represents each tweet in the dataset. There are eight colours for the predicted output value. Four-hundred sample tweets are visualised.

The predicted values are frequently in the range of 0.1 - 0.3 (orange colour) and 0.3 - 0.5 (yellow colour). In addition, the extreme values closed to one (i.e. range 0.9 - 1) are less predicted while another extreme values closed to zero (i.e. 0 - 0.1) are more frequently predicted.

6.2 Examples of the tweets

After application of various predictive models and visualisations of the dataset, it was observed that prediction of correct probability values is a difficult task. In most of the cases, the root mean square error is quite high and it is predicting a completely different sentiment that what is expected. To understand it clearly we have taken some examples from the data set and tried to predict the *s4* values using our model i.e. we try to predict how much positive sentiment is associated with the tweet. The result that we got could be explained in the following tables.

S4 actual	s4 predicted	RMS error	Tweet
1	-0.07346454	1.07346454	this weather is amazing! cant wait to kick some ass on the soccer field tonight. come out to field 9 at mike rose at 6:45!!

Reason - The sentiment associated with it may be positive because the person wanted to speak about an amazing weather but since it has some negative words, the prediction went down to a negative value.

S4 actual	s4 predicted	RMS error	Tweet
0	1.05560674	1.05560674	The weather is here, I wish you were beautiful.

Reason - Here the actual value of the column is zero.Clearly there is no positive sentiment associated with the tweet because the person is not talking about weather at all.But the prediction give *s* positive value since it finds the positive word "beautiful" associated with the tweet.

S4 actual	s4 predicted	RMS error	Tweet
1	0.00940419	0.99059581	and it's Chaco weather!! #sohappy .

Reason - Originally a positive word "sohappy" is associated with the weather, but the prediction gives a very low positive value probably because "sohappy" is grammatically wrong which is not found in any tweets labelled as positive and machine cannot differentiate word from meaning.Also chaco is not a word in the english vocabulary. Therefore, the model could not interpret this tweet as happy sentiment.

7 IMPROVEMENT AND FUTURE WORK

There could have been other methods adopted for improving the score.We could have also done the following

7.1 Predict by using Location

The location column was left unused by us. We could have thought of some way to predict the kind of weather based on geographical location.That would have added another dimension to the project.

7.2 Predict by using output as features

As this problem contains 24 output labels and each category could be related to each other. For example, Storm weather could result in negative sentiment. This can be taken advantage by feeding back the previous predicted result as a feature to predict the next output labels.

7.3 Predict by using real-time tweet data

The next step of this project would be to take it one step forward and predict the weather using real time data or live data stream.The intention is to predict the probabilities as soon as possible instead of using a fixed dataset

8 CONCLUSIONS

In this challenge we have analysed the tweets related to weather. Ridge regression and LightGBM method are applied to predict the twenty-four outputs of each tweet.

From our experiment, the results show that LightGBM has performed the best compared to the other model. The RMSE is the less in this case accounting to 0.2 compared to sparse regression which has not been very efficient in handling the imbalanced data. Also we observed that predicting human sentiment based on varied confidence score is a difficult task.

This challenge sought our attention in particular since this has a wide application in real world. For example, this can be used to predict the sentiment associated with each hashtag in a live stream of data. The intention is to find out what the common people are saying about it. This will be extremely helpful for big brands to predict the review of the tweets associates with their products.

REFERENCES

[1] Piyush Bhardwaj. 2016. How does doc2vec represent feature vector of a document? Can anyone explain mathematically how the process is done? - Quora. (2016). https://www.quora.com/How-does-doc2vec-represent-feature-vector-of-a-document-Can-anyone-explain-mathematically-how-the-process-is-done.

[2] Microsoft Corporation. [n. d.]. Microsoft/LightGBM: A fast, distributed, high performance gradient boosting (GBDT, GBRT, GBM or MART) framework based on decision tree algorithms, used for ranking, classification and many other machine learning tasks. It is under the umbrella of the DMTK(http://github.com/microsoft/dmtk) project of Microsoft. https://github.com/Microsoft/LightGBM. ([n. d.]).

[3] Microsoft Corporation. 2017. Parameters Tuning — LightGBM documentation. http://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html. (2017).

[4] CrowdFlower. 2014. Partly Sunny with a Chance of Hashtags | Kaggle. https://www.kaggle.com/c/crowdflower-weather-twitter/discussion/6488. (2014).

[5] Guolin Ke et el. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf. (2017).

[6] Ramiro H. Gálvez and Agustín Gravano. 2017. Assessing the usefulness of online message board mining in automatic stock prediction systems. *Journal of Computational Science* 19 (2017), 43–56.

[7] L. E. Melkumova and S. Ya Shatskikh. 2017. Comparing Ridge and LASSO estimators for data analysis. *Procedia Engineering* 201 (2017), 746–755.

[8] Wooju Kim Sangheon Lee, Xiangdan Jin. 2016. Sentiment classification for unlabeled dataset using Doc2Vec with JST. https://dl.acm.org/citation.cfm?id=2971631. (2016).

[9] Gidi Shperber. 2017. A gentle introduction to Doc2Vec – ScaleAbout – Medium. https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e. (July 2017).

[10] Vadim Smolyakov. 2017. Ensemble Learning to Improve Machine Learning Results. https://blog.statsbot.co/ensemble-learning-d1dcd548e936. (August 2017).