Automatic Prediction of Car Brand Ownership Based on Tweets

Wei Xu weixu@umich.edu
Weizi Liu weizliu@umich.edu
Renhan Zhang rhzhang@umich.edu
Bohan Zhao zhaoboha@umich.edu

Abstract

In this project, we are building a successful classifier to automatically predict the car brand that someone is driving based on his relevant tweets. A training dataset containing thousands of Twitter user samples together with their tweets will be crawled from Twitter API. Each user sample will be labeled with the car brand that they drive based on the information from their tweets. Three prediction model using Decision Tree, Naive Bayes, and SVM(Support Vector Machine) classifier are built to learn the pattern from the training dataset. Given a new user sample with his several tweets, our prediction model will be able to predict the car brand that this user owns.

1 Project Description

1.1 Project Definition

Social media and social network play a very important part in people's expressing feelings and opinions in different kinds of things. Based on the observation that cars have been a majority part of people's life and social network users will often post messages which are relevant to cars, we want to actually collect and retrieve this kind of useful information to be able to analyze and evaluate people's behaviors and characteristics related with car brands.

1.2 Project Goal

The goal is to predict the car brand ownership of social network users based on the messages they post on the social website. Besides, we want to analyze and understand differences and characteristics (if any) between people driving different cars. We choose Twitter website as the social network from which we will collect user data and user tweets since its popularity among people and its powerful Twitter API which provides easy access to its user data. The classification system we build will be able to predict what kind of car the twitter user is driving, given the information (tweets) of that user.

1.3 Project Applications

This project can have two possible applications. The first is to collect Twitter user's tweets about cars so that we can identify people's opinions on certain car brand, and provide consumer feedback based on social media for car companies. The other application is to identify the Twitter users who drive expensive cars, and send them related luxury ads or car insurance ads accordingly.

2 Related Work

Car brand classification using tweets from Twitter API is a very specific problem, which we can hardly find the previous research and work on it. But when analyzing the underlying project goal and the dataset we are creating, we conclude that the car brand classification problem in our project can be simplified as common text classification problem. This area has been widely explored and researched, and we can refer to much related work which will be useful and helpful for our project.

Ravindrea C. et al. [1] asserted that hedonic and utilitarian considerations affected customers' behaviors. This effect became increasingly dominant when they purchasing goods such as houses and cars. Thus the choice of cars would implicitly reflect the owners' personalities.

On the other hand, what car one drives has effect on the drivers' social image. For example, **Gad S. et al.** [2] demonstrated that females' ratings of males attractiveness were affected by the car he drove and hence argued that a man can appear more physically attractive to women by owning a sports car.

Fabrizio, S. [3] gives an overview of text categorization and text classification problem. Their paper outlines the fundamental traits of the technologies involved in text categorization field, and explains the applications that can feasibly be tackled through text classification. They also provide the tools and resources that are available to the researcher and developer wishing to take up these technologies for deploying real-world applications.

This paper basically provides a boarder understanding about the area and research of text classification. The methods and application discussed in this paper about the performance of SVM and Boosting algorithm give us the basic thought of Machine Learning algorithms which we are going to implement in out project.

Thorsten, J. et al. [4] explores the use of Support Vector Machines (SVMs) for learning text classifiers from examples. They use this kind of SVM classifier to analyze the particular properties of learning with text data and identifies why SVMs are appropriate for this task. They firstly transform the give text document into the vector form which can be used by the common Machine Learning algorithm. In this way, they use the distinct words in all documents as the feature. To reduce the number of features, they use the information gain criterion to extract a subset of features to be used for classification. Besides, tf-idf weighting is also used to refine the vectorization of each document, which they claimed as a improvement for the performance of classifier.

From their observations and experiment results, we conclude that SVM can do well in text classification problem, and tf-idf weighting can also be applied for the vectorization of text documents. These conclusion will guide our team to try the effective methods for our prediction model.

Eibe, F. et al. [5] presents an improved Multinomial naive Bayes (MNB) algorithm for text classification with unbalanced classes. Based on the basic MNB algorithm, they research on another transformation that is designed to combat a potential problem with the application of MNB to unbalanced datasets. in this paper, they propose an appropriate correction by adjusting attribute priors, which can be implemented as another data normalization step, and their paper shows that it can significantly improve the area under the ROC curve, which means that the performance is leagely improved.

The reason that we are interested in this paper is that it gives attention on the particular problem of unbalanced classes. Since in our project we observe that some car brands are really popular and can provide large amount of Twitter users who own this type of car, while other car brands are relatively uncommon in people's tweets. This observation reminds us that we may need to deal with the problem of unbalanced dataset. This paper definitely give an optimized solution for this problem, and we may refer to their method when we build our prediction model and classifier.

Peter D. Turney. et al. [6] gives a deep survey and explanation on Vector Space Model. In this paper the concept of TF-IDF weighting is deeply discussed, and many other document prepossessing methods such as sparse matrix reduction are presented.

We refer to this paper to get more understanding of Vector Space Model since we think that the first step of text classification should be the transformation and vectorization of the documents. we will also use tf-idf weighting as an efficient method to vectorize our collected documents.

Wikipedia page[7] gives a list of almost all car brands in the world. This may not be a strictly research work, but it provides primary and very important information for our project. From those car brands, we are able to select some common car brands which can be widely used or talked by people.

3 Data Description

3.1 Data Collection Method

We use Twitter API to grab tweets from Twitter users. Application access is created on the Twitter Developers website. **Tweepy package**(http://www.tweepy.org/), which is a easy-to-use Python library for accessing the Twitter API is used to help build our Python code of retrieving tweets data. We compile a list of common car brands to be used as the filter words to retrieve relevant tweets which can indicate that the twitter user does process this car.

We obtain the complete dataset for our project based on the steps described as follows:

- Firstly, we finish the work of crawling tweets from Twitter API, and we crawl 5000 tweets using Python and Tweepy Library for 12 car brands. Besides, the USER ID of each tweet is also obtained, so that we can get more tweets from this Twitter user in the later steps. Those tweets are saved in a csv file for the further manual checking work.
- Next, we manually check all 5000 tweets to see if each tweet is able to indicate the ownership of this Twitter user. For the tweet who really demonstrates that the Twitter user it belongs to does own this car, we regard it as a valid tweet and save its USER ID and corresponding car brand.
- One problem is that when we look at the result of our manually checking, we found that we get very few valid instances for certain car brand, such as we only get 2 valid instance for the car brand "Chevrolet". This results from the inferred fact that very few tweets which we retrieved from the Internet talked about this kind of car. Considering that the Machine Learning algorithm will not be able to learn the pattern from too few samples. We remove the above unbalanced car brands and the corresponding tweets. To still be able to achieve the goal of building our prediction model based on enough car brands, we add several other car brands and repeat the work from Step A to Step C.
- Then, using the set of valid USER ID obtained from Step C, we go back and retrieve more tweets for each user. We then combine all the tweets from every user into one document, which will be regarded as one data sample. We totally get 1181 documents for 1181 distinct Twitter users. These documents will be used as the complete dataset to train and test our Machine Learning algorithm and prediction model.

3.2 Data Annotation Method

Since our project goal is to build a classifier to successfully predict the car brand based on the information from user's tweets, we need to define and specify the useful features for our classifier and the label our classifier will output.

- **Manually check the label**. Note that there may be multiple tokens about car brands in one document, and there may be semantic difference in the tweet, so there are wrong labels when we only use algorithms to assign the label for that document. Thus we manually check the correctness of the labeling work for documents, making sure that the correct label (car brand) is assigned for each document.

- **Feature definition**: For each Twitter user, we grab the tweets about the car he/she is driving or the car he/she endorses. After checking the correctness of the assigned label for each tweet, we also go back to retrieve more tweets about that user which hopefully will present relevant topics about his/her car. We want to use them as the relevant features or factors for the prediction of our classifier.

- **Data Labeling**: For all the tweets we retrieve for each user, we combine all tweets into one document for that user. This kind of document is be considered as one data sample. The label for that data samples is assigned to be the car brand of that user. We basically use this method to annotate and label our documents or data samples.

3.3 Interesting Data Samples

Please refer to the complete dataset that we submit on CTools. We name each user document with its USER ID and its car brand, which can be easier to identify and extract necessary information. In each user document, we combine the retrieved user tweets as the whole document.

3.4 Statistics on the data collected to date

A total of 1813 valid users and corresponding documents for 12 labels were collected in our complete dataset. Below is the distribution of different car brands.

We use this complete dataset to build our prediction model and evaluate its performance. Note that we have an in-balanced dataset here because we have large portion of "Jeep" car labels. To evaluate our prediction model based on the knowledge of in-balanced factor, we evaluate the baseline accuracy which is 0.28. This baseline accuracy basically evaluate the extreme case that what our classification accuracy will be if all data samples are classified as the most frequent car label - "Jeep".

Name	BMW	Honda	Jeep	Audi	Ford	Hyundai	KIA	Lexus	Mazada	Nissan	Toyota	Ferrari
Number	510	169	142	144	50	110	97	104	112	87	246	41
Percentage	28%	9%	7%	8%	2%	6%	5%	5%	6%	4%	13%	2%

4 Method Description

Based on our complete dataset, we firstly use text prepossessing methods to tokenize and vectorize the documents into feature matrix. we implement three machine learning algorithms, including Decision Tree, Naive Bayes and SVM to build our prediction model.

4.1 Text Preprocessing

4.1.1 Introduction

In the preprocessing section, we need to handle two mainly problem: tokenization and vectorization. We have applied the function *TfidfVectorizer* from scikit-learn to pre-process the documents. Besides the *TfidfVectorizer*, we have also applied the tokenizer from CMU Tweet NLP which is specially designed to tokenize the tweets.

4.1.2 Methodology

- Firstly, we use the function *TfidfVectorizer* to initialize object *vectorize* for the tokenization and remove the stop word.
  TfidfVectorizer(encoding='ISO-8859-1', sublinear_tf = True, max_df = 0.5, stop_words = 'english')
  the '*max_df = 0.5'* means that a word will be ignored if its *df* is larger than 0.5;
  the '*stop_words = 'english"* means need to remove the stop-words from the documents.
- Second, we use the function *vectorize.fit_transform()* the vectorize the training corpus, then use the function *vectoriz.transform()* to vectorize the test corpus upon the already built matrix.
- Finally, we have applied the tokenizer from CMU Tweet NLP by using the same function.

TfidfVectorizer(encoding='ISO-8859-1', sublinear_tf = True, max_df = 0.5, stop_words = 'english', tokenizer = twokenize.tokenizeRawTweetTect)

4.2 Decision Tree Classifier

4.2.1 Introduction

In this project, we firstly use decision tree to multi-label classification and we use information gain as criterion to select feature for each node. The advantage of it is that it is easy to understand and interpret.

4.2.2 Methodology

- Firstly, we create a dictionary which maps car brands to numbers: *car_brands = {'bmw': 0, 'honda': 1, 'jeep': 2, 'audi': 3, 'ford': 4, 'hyundai': 5, 'kia': 6, 'lexus': 7, 'mazda': 8, 'nissan': 9, 'toyota': 10, 'ferrari': 11}*;
- Second, reading all the data to a list named data, all the relative labels (as numbers) into a list named label_number.
- Third, in order to make the vectorizer => transformer => classifier easier to work with, scikit-learn provides a Pipeline class that behaves like a compound classifier:
  clf = Pipeline([('vect', TfidfVectorizer()), ('clf', tree.DecisionTreeClassifier,])
  which means that our vectorized method is using tfidf, and our classifier model is decision tree. However, we modify scikit-learn source code because 1)DecisionTreeClassifier requires a non-sparse matrix input, but when we fit data, which output a sparse input, so we need to transform this sparse matrix to a non-sparse matrix, just set X = X.toarray() in fit function of pipeline.py; 2)In order to output visualized decision tree for each fold(we use K-fold method to evaluation), we just add some parameters to record information(such as map from vocabulary index to real tokens) of each fold, and also we use pydot.graph_from_dot_data to draw decision tree. Details refer to code description and README file.
- Fourth, we apply *cross_validation.StraitifiedFold* function to separate our data into 5 groups, using 4 of these as our training data and another one is test data in a running time, do these for 5 times to get 5 different accuracies. The reason why we do that is to get an average accuracy, and make sure that the method is not overfitting.
- Then, after getting training data and test data, we can apply classifier on training data to train a model and predict labels for test samples.
- Finally, we get the result including precision, recall, f1 score and confusion matrix as well as visualized decision tree and a map from vocabulary index to real tokens.

4.3 Naive Bayes Classifier

4.3.1 Introduction

Naive Bayes method produces a posterior probability distribution over the possible categories given a description of an item, which is based on Bayes Theorem.

4.3.2 Methodology

- Firstly, we create a dictionary to index car brands as what we do in Decision Tree method.
- Second, reading all the data to a list named data, all the relative labels (as numbers) into a list named label_number.
- Third, in order to make the vectorizer => transformer => classifier easier to work with, scikit-learn provides a Pipeline class that behaves like a compound classifier:

  clf = Pipeline([('vect', TfidfVectorizer(stop_words = stopwords, token_pattern=ur"\b[a-z0-9_\-\.]+[a-z][a-z0-9_\-\.]+\b",)), ('clf', MultinomialNB(alpha=0.01)),])

  which means that our vectorized method is using tfidf and removing stopwords, and our classifier model is multinomial naïve bayes with alpha parameter is 0.01.
- Fourth, we apply *cross_validation.StraitifiedFold* function to separate our data into 5 groups, using 4 of these as our training data and another one is test data in a running time, do these for 5 times to get 5 different accuracies. The reason why we do that is to get an average accuracy, and make sure that the method is not overfitting.
- Then, after getting training data and test data, we can apply classifier on training data to train a model and predict labels for test samples.
- Finally, we get the result including precision, recall, f1 score and confusion matrix.

5.2 Support Vector Machine(SVM)

5.2.1 Introduction

The support vector machine (SVM) constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification. In this project, we apply the SVM algorithm with Linear kernel. To find the optimized parameters used for SVM algorithm, we use Grid Search method and apply the function '*GridSearchCV*' imported from 'scikit learn' package.

5.2.2 Methodology

- Firstly, we create the car brand dictionary as what we do in the Naïve Bayes method, and then we get the authentic labels of each sample in the data collection.
- Secondly, we apply the function '*cross_validation.StratifiedKFold*' to divide the whole data collection into training set and test set. The "*StratifiedKFold*'' function provides train/test indices to split the data into training set and test set, preserving the percentage of samples for each class. We use this function divide the entire data collection into 5 folders, which means that every time we use one folder as test set and the other four folders are training set. Therefore, we will have five results of prediction for five different folders.
- Thirdly, we need to vectorize the training data and test data by using the function

  TfidfVectorizer(encoding='ISO-8859-1', sublinear_tf = True, max_df = 0.5, stop_words = 'english')
- As shown by the name of this function, the vectorized method is using the method of removing stopwords and tf-idf weighting. Then we fit these training vectors and training labels to build the model. To build the optimized model, we apply the function '*GridSearchCV*' to get the optimized parameters.
- Finally, we use the optimized parameters to fit the SVM model. Then we use the vectorized test data to predict their labels and get the result including precision, recall, f1 score and confusion matrix.

6 Results

6.1 Evaluation Metrics

The evaluation of effectiveness of the models was made based on the following metrics.

- **Precision**: for a car brand, the precision is the number of drivers who actually drives and are deemed as driving the car of this brand divided by the total number of users who are considered as driving this category of car. For a data set, its precision is the weighted average of the precision of each of the brand.
- **Recall**: for a car brand, the recall is the number of drivers who actually drives and are deemed as driving the car of this brand divided by the total number of users who actually drive this category of car. For a data set, its recall is the weighted average of the recall of each of the brand.
- **F1-score**: an overall measurement of the effectiveness taking into consideration both precision and recall. It is given by the following equation where β is an adjustable hyper-parameter.

$$f1 - score = \frac{(1+\beta^2)\, precision * recall}{\beta^2 * precision + recall}$$

- **Accuracy**: for a data set, the accuracy is the number of correctly classified data points divided by the total number of data points

6.2 Decision Tree Classifier

We use Stratified K-fold cross validation and set n_folds=5 to present our classification results, so we will basically do the training and test 5 times. The following table give the classification result for each time.

Fold	precision	recall	f1-score	accuracy
1	0.18	0.19	0.19	0.19
2	0.23	0.22	0.22	0.22
3	0.17	0.16	0.16	0.16
4	0.23	0.23	0.23	0.23
5	0.26	0.26	0.26	0.26
Average	0.214	0.212	0.212	0.212

6.3 Naive Bayes Classifier

We use Straified KFold cross validation and set n_folds=5 to present our classification results, so we will basically do the training and test 5 times. The following table give the classification result for each time.

Fold	precision	recall	f1-score	accuracy
1	0.25	0.32	0.21	0.3189
2	0.26	0.33	0.22	0.3260
3	0.31	0.37	0.27	0.3656
4	0.24	0.35	0.23	0.3463
5	0.23	0.34	0.23	0.3352
Average	0.258	0.342	0.232	0.3384

6.3 SVM Classifier

We also use Strasified KFold cross validation method and set n_folds=5 to evaluate our classifier. The following table gives the best result in the 5 times classification result.

Fold	precision	recall	f1-score	accuracy
1	0.41	0.4	0.31	0.4
2	0.40	0.41	0.32	0.41
3	0.42	0.44	0.35	0.44
4	0.42	0.4	0.3	0.4
5	0.31	0.39	0.29	0.39
Average	0.392	0.408	0.314	0.408

6.4 Comparison Against Baseline

By examining the distribution of the data set, we noticed that by always predicting the most frequent brand, Jeep, we can obtain an accuracy of 28%, a figure that an effective classifier should do better than. Below is a summary of the comparison between the results of different models and the baseline.

Baseline	Decision tree	Naive Bayes	SVM
28%	21.2%	33.84%	40.8%

6.5 Summary

From 6.4, it's concluded that decision tree is ineffective in that it does worse than the baseline while Naive Bayes and SVM are more effective. SVM performs uniformly better than other models in each of the fold at the cost of longer training time.

7 Conclusions

7.1 Analysis

7.1.1 Informative Words

When using decision tree, we also analyze the most distinguishing words which have the highest information gain, which can be found in our visualized decision tree and corresponding map. And it turns out that words like road, beach and mom are quite distinguishing. The words related to car brand, such as r8 and cruiser have similar distinguishing power.

7.1.2 Most Frequent Words

For each of the car brand, we also record the most frequent words in this category. Below is a subset of these frequent words of interest.

Audi: can, should, do, check, like
BMW: get, on, time, change, love
Ferrari: time, race, saturday, love
Ford: new, now, do, need, people, work
Honda: best, drive, work, love
Hyundai: work, life, thank, reality

Jeep: love, people, girl, lol
Kia:don't, say, lol
Lexus: still, we, leave
Mazda: look, love, do, so, man, now
Nissan: do, go , agree, #beer, #health
Toyota: new, we, today, work

7.2 Reflection and Discussion

7.2.2 Over-fitting and Cross Validation

When applying the different classifiers, we observe that our dataset contain 1181 documents, which is a large amount. On the other hand, because the documents distribution of each brand is uneven, we need to keep the percentage of each car brand in test corpus and training corpus be same. Therefore, we have applied the cross validation to divide the whole corpus into 5 folder, the car brand distribution is same among these 5 folders. Once a folder being the test corpus, the other 4 folders are training corpus. After testing on each folders, we finally get 5 results in Decision Tree, Naive Bayes and SVM classification. We have observed that the 5 results are similar to each other, which means that the cross validation is stable for evaluation the result. What's more, the cross validation can provide the result with lower variance. Since a simple split of training set and test set may result in the over-fitting problem, we expect to use this k-fold cross validation to experiment on more splits and thus avoid possible over-fitting problem in our project.

7.2.3 CMU Tokenizer

Besides applying the tokenizer from build-in function from scikit-learn, we also have applied the tokenizer from CMU Tweets NPL. Results show that using the CMU Tweets NPL tokenizer dose not significantly promote the predict performance. Therefore, we decide to use the scikit-learn bulid-in tokenizer for the preprocessing in this project.

7.2.4 SMOTE and Imbalance Dataset

When observing the dataset we have, it's obvious that we have a in-balance dataset since 28% of data samples belong to the label "Jeep". We evaluate the baseline accuracy for our dataset so that we have a minimum requirement for the performance of our prediction model.

Besides, we also researched on the common methods of dealing with in-balance dataset. Under-sampling the majority label can be a method to solve the in-balance problem. We implement this and can not get improvement for the performance. We then try another method which basically over-samples the minority labels. A popular algorithm called SMOTE is developed for this over-sampling method. We also implement it, but unfortunately we find that this algorithm can not work on sparse matrix that we have in our text classification problem.

7.3 Future Work

In this project, we have successfully implemented Decision Tree, Naive Bayes, and SVM to build a prediction system to predict user's car ownership based on the tweets he/she posts on Twitter website. There are still several points that we anticipate for the future work:

- Dataset can be enlarged to get more user documents as training set and test set.
- More car brands can be added so that the generality of the project can be improved.

- Linguistic analysis methods can be implemented to analyze and find more interesting patterns or characteristics in different car labels, such as dominant word classes in each car label documents

8 Individual Contributions

Wei Xu: Manage project progress and deadlines, and organize group discussions; Write checkpoint reports; Implement the code of grabbing tweets from Twitter API given car brands; Manually check the car label for retrieved tweets; Implement the method of SVM classifier with teammate Bohan Zhao.

Bohan Zhao: Manually check the retrieved tweets; Implement the code of crawling tweets with a specified user id using Twitter API; Establishing SVM model; Apply the tokenizer from CMU; Participate writing the checkpoint reports and final reports.

Weizi Liu: Manually check the retrieved tweets; Use Twitter API to crawl tweets with a specified user id; Build Naive Bayes model; Build Decision tree model; Write reports.

Renhan Zhang: Manually check the retrieved tweets; Use Twitter API to crawl tweets eligible for training. Linguistic analysis on most frequent words of each car brand.

9 Reference

1. Gad Saad. 2010. Men: You're Only As Good-Looking As the Car That You're Driving https://www.psychologytoday.com/blog/homo-consumericus/201012/men-you-re-only-good-looking-the-car-you-re-driving
2. Ravindra Chitturi, Rajagopal Raghunathan, Vijay Mahajan (2008) Delight by Design: The Role of Hedonic Versus Utilitarian Benefits. Journal of Marketing: May 2008, Vol. 72, No. 3, pp. 48-63.
3. Fabrizio Sebastiani. 2005. Text Categorization. Text Mining and its Applications to Intelligence, CRM and Knowledge Management, WIT Press, 109-129.
4. Thorsten Joachims. 1998. Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. In Proceedings of the 10th European Conference on Machine Learning (ECML '98), Claire Nedellec and Céline Rouveirol (Eds.). Springer-Verlag, London, UK, UK, 137-142.
5. Eibe Frank and Remco R. Bouckaert. 2006. Naive bayes for text classification with unbalanced classes. In Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases (PKDD'06), Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (Eds.). Springer-Verlag, Berlin, Heidelberg, 503-510.
6. Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. Volume 37, pages 141-188
7. List of Car Brands http://en.wikipedia.org/wiki/List_of_car_brands
8. Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Sch. 2005. A tutorial on ν-support vector machines: Research Articles. Appl. Stoch. Model. Bus. Ind. 21, 2 (March 2005), 111-136. DOI=10.1002/asmb.v21:2.
9. Scikit Learn Package Documentation http://scikit-learn.org/stable/documentation.html
10. Twitter API Documentation https://dev.twitter.com/rest/public
11. Tweepy Package Documentation for Twitter API http://tweepy.readthedocs.org/en/v3.2.0/
12. Tweepy Youtube Tutorial to use Twitter API