CS 579 Project Report

Jiaqi Chen (A20306740)
Xingtan Hu (A20304622)
Xiaoyang Lu (A20315603)

Introduction

Nowadays, people's reviews for automobile is not only based on its price and depreciation. The ownership cost is a key factor. One biggest automobile website Edmunds.com has came up with the idea of TCO, which means the True Cost to Own. It calculates the additional costs you may not have included when considering your next vehicle purchase, including: depreciation, interest on your loan, taxes and fees, insurance premiums, fuel costs, maintenance, and repairs.[1]

In our views, we think this TCO price is more meaningful than the only price for a vehicle. We decided to find the connection between TCO price and people's review. Our hypothesis is that people's review for an automobile may be affected by TCO.

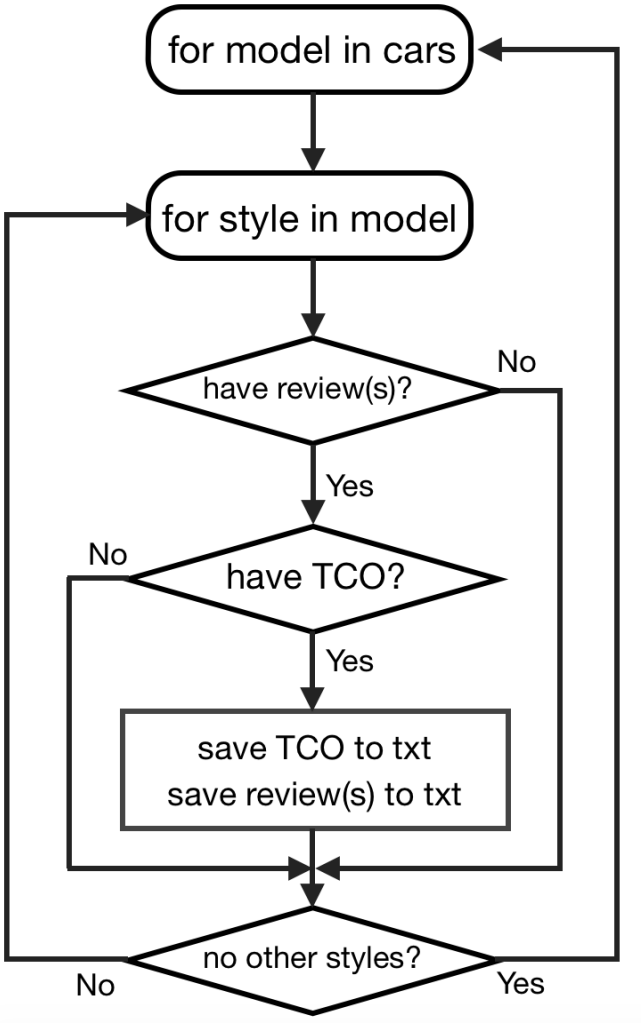
In this project, we will first collect the review data from Edmunds via Edmunds API. Then trying to take advantage of several existing classification and regression models to complete the machine learning process. At last, we will compare the result we get from our test-set, and draw the conclusion on the connection between TCO price and customer reviews.

Data Collection

We collect the reviews data and TCO prices from Edmunds.com via Edmunds Vehicle API[2].

The reviews and TCO prices data is from 12 different kinds of types in year 2013 and 2014: compact suv, small cars, minivan, compact pickups, wagon, midsize car, midsize suv, full size pickup, hybrid car, sport car, hybrid suv, covertible. And each type includes at least 10 models. And every model has different amount of styles, which also have different reviews and TCO prices along with it.

1. Based on all the card types shown in Edmunds.com, we manually choose all the cars in 2013 and 2014, then classify them in the txt files. e.g. compact_suv.txt
2. Register the Edmunds developer, and get the Key for API.
All the operation to Edmunds API can only be accessed by using the Key
3. Collect TCO price and review
we use a program(Collect_Auto_review_TCO.ipynb) to collect data. The input of this program is the txt files we established in step 1. The output should be TCO prices and reviews of cars listing in txt files. Besides that, one car(model) might have one or several TCO prices along with different amount of reviews, because all the cars in different kinds of model in different types have theirs own styleid(s). One car may has one or many different styleid(s). It's because that one car may has a two-door edition or a four-door edition.



4. Filter data

Here's our method to filter data:

1)For each model in car list, we first need to get all the styles for this model.

2)For each style, we check if it has review(s), if it doesn't have, we find next style of this model until there's no other styles in this model.

3)If this style has review(s), we get the TCO of this style, if this style doesn't have TCO price, we find next style of this model until there's no other styles in this model.

4)After get the TCO along with review(s) for this style, we save the TCO to a txt file along with the make, model, year and styleid. Then saving the review(s) to a txt which named as styleid_make model year.txt.

5)After find all the styles in this model, we start to find next model.

6)Repeat step(2-6)

Moreover, there won't be any replicate TCO price or reviews for a same style in our dataset. Because, all the same styles or models will replace the previous data.

Methods

1. Merge data

After we got many txt files for review and TCO price, we have to merge all the TCO files to a txt file named total_cars_TCO.txt file. And merging all reviews folder in a folder named total_cars_review.

2. Collect data

We collect the data from all .txt file which including reviews and TCO prices. We classify the TCO price into 7 price ranges: 0~15000, 15000~25000, 25000~35000, 35000~45000, 45000~55000, 55000~85000,85000~Max, which is based on the price ranges in the Edmunds.com's Car Finder page[3]. And these price ranges become our price label. Moreover, we collect all the reviews as review_data.

3. Tokenization
we tokenize all of reviews by using the sklearn's built-in tokenization function. And we choose one of the vectorizer function - TF/IDF vectorizer to transform terms of reviews into a sparse matrix.[4] And we also use three parameters: stop words, ngram and min_df.

4. Machine Learning method
1) Regression(sklearn)
In our case, because there's no obvious linear relationship between our review_data and price_data, we decided to use the Logistic Regression method which also a built-in function in sklearn[5]. We set the parameter c for regularization, it means inverse of regularization strength, smaller values specify stronger regularization.

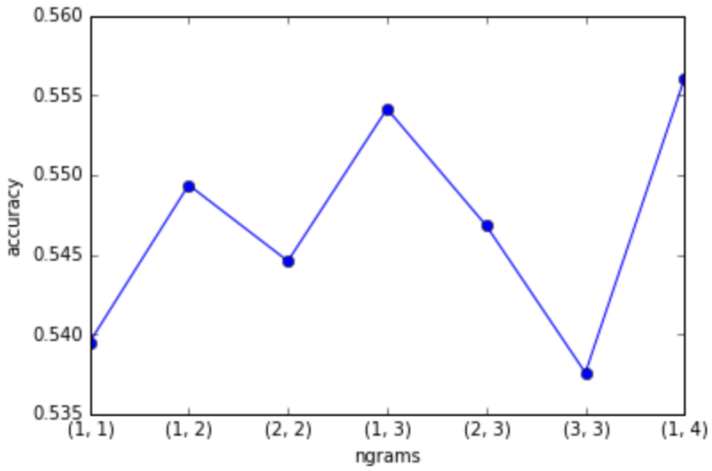
2) Classification(sklearn)
And we also use two classification method of Naive Bayes: Gaussian Naive Bayes[6] and Bernoulli models[7]. Besides that, we use the same estimators in the Regression to get mean accuracy.

3) Textblob
We also tried another approaches—textblob, of which we use a Naive Bayes classifier. we established the initial training model based on reviews of several models, and updating the classifier with reviews of other models, we can only do this prediction in each type of the automobiles, but not all types of automobiles, because it is really time consuming, almost need 10 days to finish.

5. Cross Validation
In order to get a much more reliable results of accuracy, we did cross validation for both regression and classification method we have mentioned before. We use a method called KFold[8] which is built in sklearn. Then we get an average cross validation score.

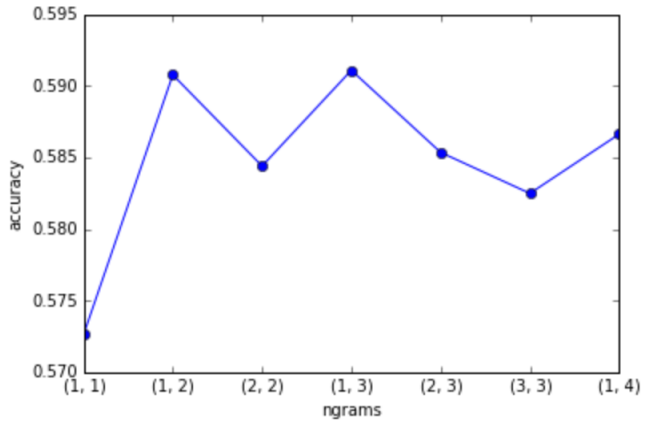
Experiments

1. Comparing different preprocessing decisions(ngram):
1) Logistic Regression(L2 regularized)



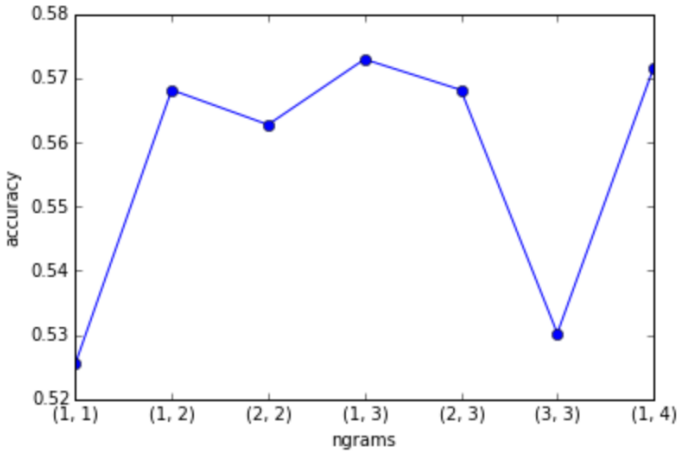

```
(1, 1)
X dimensions= (3140, 6925)
(1, 2)
X dimensions= (3140, 46984)
(2, 2)
X dimensions= (3140, 40059)
(1, 3)
X dimensions= (3140, 94421)
(2, 3)
X dimensions= (3140, 87496)
(3, 3)
X dimensions= (3140, 47437)
(1, 4)
X dimensions= (3140, 136180)
```

2) Bernoulli models




```
(1, 1)
X dimensions= (3140, 6912)
(1, 2)
X dimensions= (3140, 43748)
(2, 2)
X dimensions= (3140, 36836)
(1, 3)
X dimensions= (3140, 76666)
(2, 3)
X dimensions= (3140, 69754)
(3, 3)
X dimensions= (3140, 32918)
(1, 4)
X dimensions= (3140, 105840)
```

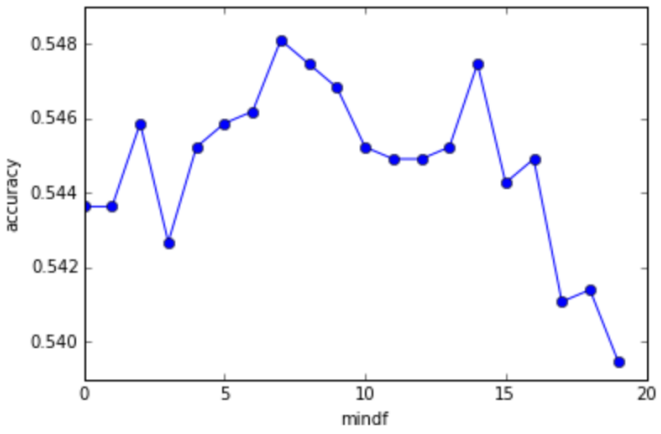
3) Gaussian Naive Bayes




```
(1, 1)
X dimensions= (3140, 6912)
(1, 2)
X dimensions= (3140, 43748)
(2, 2)
X dimensions= (3140, 36836)
(1, 3)
X dimensions= (3140, 76666)
(2, 3)
X dimensions= (3140, 69754)
(3, 3)
X dimensions= (3140, 32918)
(1, 4)
X dimensions= (3140, 105840)
```

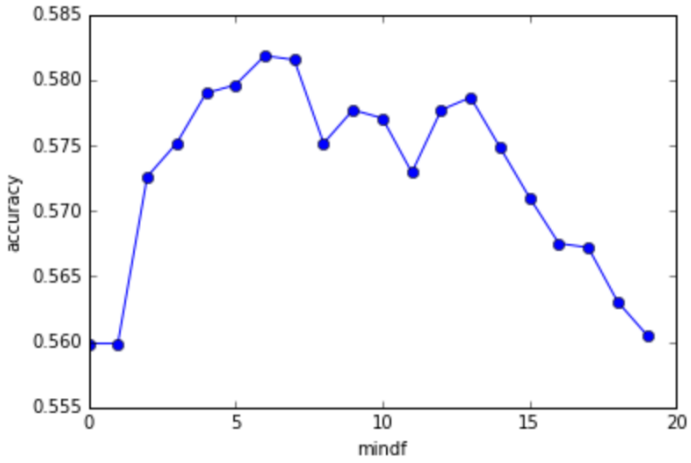
2. Comparing how filtering rare words(min_df) affect accuracy

1) Logistic Regression



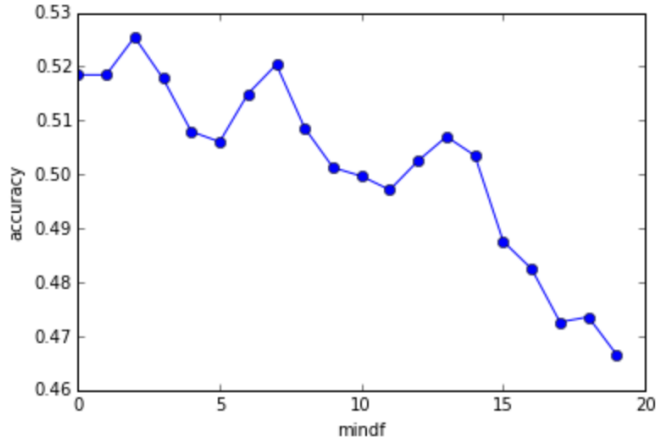

```
X dimensions= (3140, 11442)
X dimensions= (3140, 11442)
X dimensions= (3140, 6912)
X dimensions= (3140, 5186)
X dimensions= (3140, 4369)
X dimensions= (3140, 3767)
X dimensions= (3140, 3331)
X dimensions= (3140, 3009)
X dimensions= (3140, 2760)
X dimensions= (3140, 2570)
X dimensions= (3140, 2401)
X dimensions= (3140, 2258)
X dimensions= (3140, 2138)
X dimensions= (3140, 2037)
X dimensions= (3140, 1923)
X dimensions= (3140, 1848)
X dimensions= (3140, 1777)
X dimensions= (3140, 1699)
X dimensions= (3140, 1632)
X dimensions= (3140, 1571)
```

2) Bernoulli models



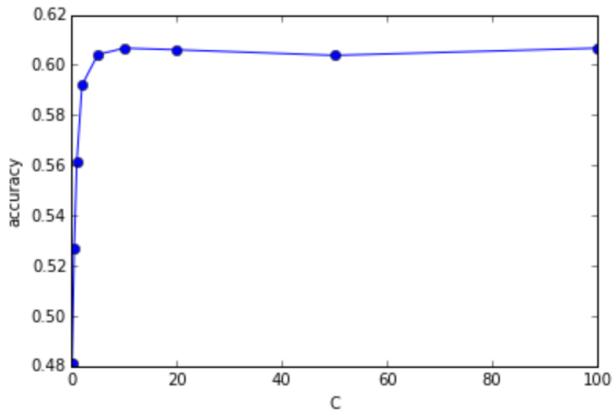

```
X dimensions= (3140, 11442)
X dimensions= (3140, 11442)
X dimensions= (3140, 6912)
X dimensions= (3140, 5186)
X dimensions= (3140, 4369)
X dimensions= (3140, 3767)
X dimensions= (3140, 3331)
X dimensions= (3140, 3009)
X dimensions= (3140, 2760)
X dimensions= (3140, 2570)
X dimensions= (3140, 2401)
X dimensions= (3140, 2258)
X dimensions= (3140, 2138)
X dimensions= (3140, 2037)
X dimensions= (3140, 1923)
X dimensions= (3140, 1848)
X dimensions= (3140, 1777)
X dimensions= (3140, 1699)
X dimensions= (3140, 1632)
X dimensions= (3140, 1571)
```

3) Gaussian Naive Bayes




```
X dimensions= (3140, 11442)
X dimensions= (3140, 11442)
X dimensions= (3140, 6912)
X dimensions= (3140, 5186)
X dimensions= (3140, 4369)
X dimensions= (3140, 3767)
X dimensions= (3140, 3331)
X dimensions= (3140, 3009)
X dimensions= (3140, 2760)
X dimensions= (3140, 2570)
X dimensions= (3140, 2401)
X dimensions= (3140, 2258)
X dimensions= (3140, 2138)
X dimensions= (3140, 2037)
X dimensions= (3140, 1923)
X dimensions= (3140, 1848)
X dimensions= (3140, 1777)
X dimensions= (3140, 1699)
X dimensions= (3140, 1632)
X dimensions= (3140, 1571)
```

3. Compare C in logistic regression model



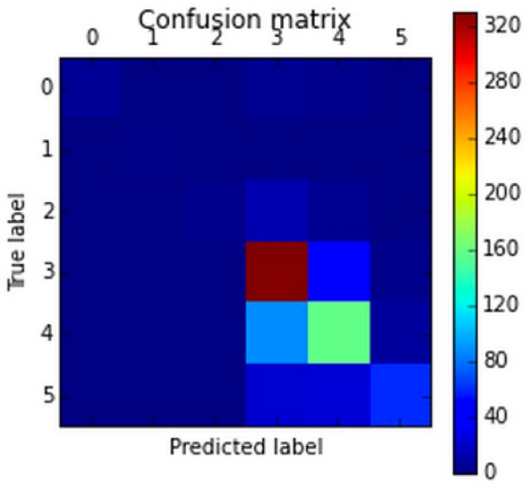

```
X dimensions= (3140, 76666)


0.01 0.480892
0.1 0.480892
0.5 0.526752
1 0.561465
2 0.592038
5 0.60414
10 0.606688
20 0.606051
50 0.603822
100 0.606688
```

The left column represents the value of C(Inverse of regularization strength), the right column represents the accuracy

4. Cross Validation

	Cross Validation Accuracy
Logistic Regression (L2)	**60.6%**
Naive Bayes (Bernoulli Model)	**56.0%**
Naive Bayes (Gaussian)	**51.8%**
accuracy of most_frequent baseline : 48.1%	

5. Confusion Matrix




```
[[  6   0   0   4   3   0]
 [  0   2   1   0   1   0]
 [  0   0   3  14   4   0]
 [  0   0   0 331  51   3]
 [  0   0   0  88 159   9]
 [  0   0   0  22  24  60]]
```

Future Work

- As we mentioned above, the connection between TCO price and review is not so obvious, we need to find the reason behind it, and try to improve it.
- We think if we can classify the TCO price range in a much more general way, or try to find another regression model, the results will be improved.
- Also, the amount of reviews we can collect from edmunds.com is vital here, unfortunately, the amount of people's review available is limited which may due to the visitor volume of the website. Here we have collected all the reviews for 2013 and 2014 automobiles, suppose the amount of the people's review will increase in future, we believe the accuracy of our model will be remarkably improved
- A professional words dictionary may be need which include words describing cars and evaluating cars

Conclusion

Based on the accuracy and cross validation results, we can draw the conclusion that TCO price has some sort of connection with review which prove our hypothesis, although the connection is not so obvious.

Reference

[1]. True Cost to Own® (TCO®)
http://www.edmunds.com/tco.html
[2]. VEHICLE API
http://developer.edmunds.com/api-documentation/vehicle/
[3]. Car Finder
http://www.edmunds.com/finder/car-finder-results.html
[4]. sklearn.feature_extraction.text.TfidfVectorizer
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer
[5]. sklearn.linear_model.LogisticRegression
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression
[6]. sklearn.naive_bayes.GaussianNB
http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB
[7]. sklearn.naive_bayes.BernoulliNB
http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB
[8]. sklearn.cross_validation.KFold
http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.KFold.html#sklearn.cross_validation.KFold