CRAIGSLIST AD CLASSIFICATION

(TEXT AND IMAGAE ANALYTICS)

Introduction

Our client for this project is Craigslist, an American classified advertisements website with sections devoted to jobs, housing, for sale, items wanted, services, community, gigs, résumés, and discussion forums. Craigslist is one of the largest peer-to-peer user generated advertisements website serving 570 cities in over 70 countries.

As the aforementioned description suggests, Craigslist is used by a growing number of the populous to peruse through advertisements and buy products or services(jobs). These advertisements are posted by individual users without any oversight from our client. This entails that the user has the authority to classify the advertisements.

Business Problem

The authorization for classifying the information being in the hands of the users increases the propensity of human error, which in an analytics context translates to misclassification of advertisements. To observe how misclassification of advertisements actually affects the customers, we first looked at the value chain of a customer making a purchase.

Fig 1.0: Value Chain of a purchase

In this value chain, our client comes in as a choice to the customer in part 2, but we decided to look deeper into part 3, the search process, where the misclassification of an advertisement can have a negative impact to the company. Moreover, while searching for a product to purchase, customers generally have what we would like to call a ‘pocket time’, a certain amount of time

after which the likelihood of a conversion channel starts to decrease as the product is not found.

This decrease in the likelihood of conversion is also accompanied with a dose of frustration, which the customer attributes to our client, which in turn decrease the reliability and brand value of our client. For example, if one would like to buy a second-hand iPhone, one would go to craigslist and search for an iPhone, the result of which is shown in Fig 2.0.

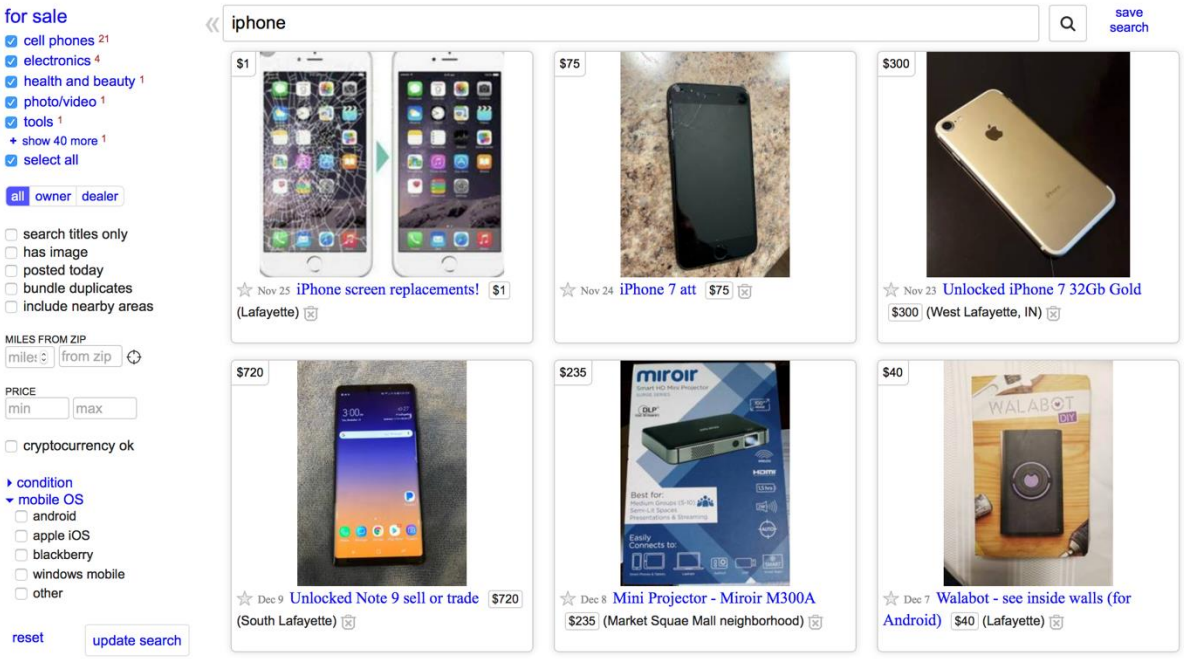

Fig 2.0: Search for iPhone on Craigslist

As we can observe in Fig2.0 the misclassification of advertisements in the search for an iPhone effectively tarnishes the user's experience and in turn our client's brand value. In attempting to solve the misclassification problem for our client we would significantly increase the reliability quotient of our client's online presence. This increase will yield benefits that amount to:

- Customer retention
- Better user experience
- Increase in web traffic

- Increase in brand value
- Increase in revenue

With the inception of this project our team endeavors to make a tool that would predict the misclassification of errors when a user posts a new advertisement. Considering the scope of the problem, the data we intend to use is the advertisement data that is currently present on the website. This data is bifurcated into:

- Image
- Text

We will gather the text data for the various categories of the goods listed on the website using Scrapy (a web-scraping tool). Using the text data for each advertisement we will build a classifier model that would predict whether or not the new advertisement is misclassified.

Data Analysis

The essence of our model lies in the data classification arena. After scraping craigslist, we have around 86,000 records pertaining to the different goods being advertised on the website and this data contains the title, description and images of each of the advertisements.
The overarching goal of our solution is to use the text information to train a classification model which will be used to classify an advertisement based on the title and the description for the item being sold. The images for each advertisement will be run through another classification model to predict what is in the image. Finally, we will stack the models and compare the classification from each model for each observation. If both the models classify the advertisement to the same category it can be said that the new advertisement has been classified correctly.

Text model:

Fig3.0: Text Model Classification Approach

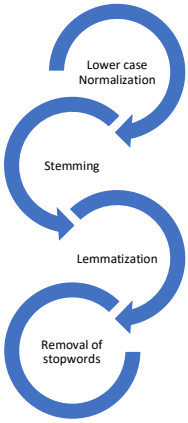

Fig4.0: Text Normalization

The text data will first be preprocessed, this cues a process (shown in Fig4.0 on the left) where the text is converted into individual tokens which are then converted into their lowercase form after which they undergo a process of stemming or lemmatization ensuring that the words in the text are in their base form (base form of the word 'eating 'is 'eat') . This text data is then split into a train and test set. The training set of the data will be passed through a vectorizer which would convert the description and title of each product into a TF-IDF matrix, this is similar to a term-document matrix. Since, all our data is labelled we have trained three models: SVM, Naïve Bayes, and a Random Forest model. Finally, the models will be tested and evaluated (as shown in Fig3.0) using the testing set of data.

Image Model:

Fig 5.0 : Image Model Classification Approach

The above figure shows the approach we took in creating the image classification model. The data was first resized to maintain uniformity amongst all the images. To actually train and run

the model the images need to be in a mathematical representation. To achieve this representation the images are converted to their RGB channels, after which they are converted to grayscale and then flattened, this is done to reduce the dimensions in the analysis. The last step of creating the mathematical representation is to reshape each image such that it is an array of one row and a set number of columns. As our data is labelled we then split this mathematical representation of the images into a training set and a test set. This training set is then used to train a variety of models and the test set is used to validate the accuracy of each model.

Validation

We used validation set approach for cross validating the results of our analysis. We divided the data into train and test sets. Model was trained on the training set and then test observations were classified using the same model. The resulting classification accuracy was then observed and noted for different model fits.

For text classification, we used five models and obtained an overall accuracy of

- 53.61% for Logistic Regression
- 51.65% for Naive Bayes
- 54.23% for Random Forests
- 57.01% for Support Vector Machines
- 49.90% for Neural Networks

Here, we have 44 categories to classify our data into, so the accuracy of null classifier here would be 2.27%. It is obvious to note that the accuracy we obtained through any of above models is way higher than with null classifier. The best accuracy is provided by Support Vector Machine Model. Screenshots are provided for these model runs in the appendix section of this document.

For the Image data:

Since the size of the image data is much larger compared to the processing power of our computing resources we could only process a subset of the image data using stratified

sampling. This data was then trained and tested resulting in similar accuracies as the text classification.

Conclusion

Our goal was to classify the advertisements correctly (prevent the misclassification of the ads displayed in Craigslist) We first scraped every advertisement under 'for sale' listing shown in the fig6.0 below. This consists of roughly 87,000 posts. By scraping here we refer to a process that includes 2 steps, namely text scraping and image scraping. Then we analyze both text and image data. Once the data is preprocessed it is inputted to the model to be trained, and to validate the model we use the test set data.

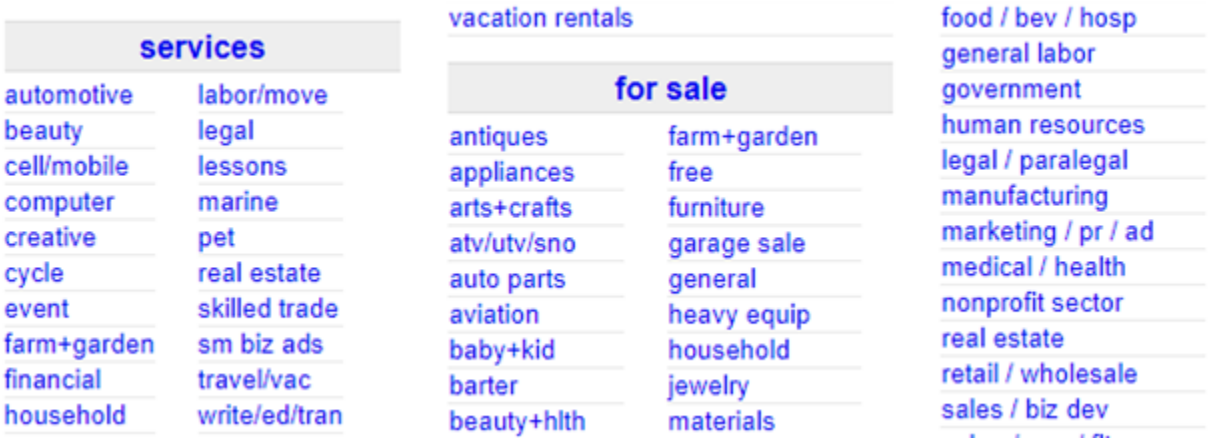

Fig6.0 : Craigslist Listings

Analysis resolves the business problem that we face to a great extent. Our model accuracy is 57%. Given that we were working with 44 categories suggests that the accuracy we obtain is high because a random assignment of class when there are 44 categories yields an accuracy of 2.72%. Compared to 2.72%, we are getting an accuracy of 57%, which is a great improvement.

Although our model accuracy is relatively high the course of action to take should be to let the users classify their posts but flag the posts that our model identifies as a misclassification. Then an employee at Craigslist should go through those posts to see if they are classified correctly by

the user. If not it should be assigned to its category. Since our model accuracy is high and users don't tend to misclassify their ads too often, we are not expecting a workload that will be unmanageable.

By following the steps described above, Craigslist will be preventing misclassification of ads, which causes buyers not to be able to locate the prospective purchases much quicker. This will introduce an increase for Craigslist's reputation and reliability in online advertisement community.

Limitations

Due to insufficient human capital and time, we are training our model using the posts categorized by users and assume that they have correctly categorized them. This is a limitation that our model faces since there are wrong classifications within the data (the very reason for the existence of our project) and these affect our results and decrease the accuracy. As a further extension, posts can be manually labelled and the model can be trained on that data, ideally increasing the accuracy by a great extent.

Once done so, this model can reach an accuracy level where no user input might be asked at the time of the advertisement creation and the model can correctly predict the category the user was intending to post their advertisement in.

APPENDIX

Text model accuracies:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)

In [252]: y_pred_logit = Logitmodel.predict(test_x)
     ...: # evaluation
     ...: from sklearn.metrics import accuracy_score
     ...: acc_logit = accuracy_score(test_y, y_pred_logit)
     ...: print("Logit model Accuracy:: {:.2f}%".format(acc_logit*100))
Logit model Accuracy:: 53.61%
```

```
In [253]: from sklearn.naive_bayes import MultinomialNB
     ...: NBmodel = MultinomialNB()
     ...: # training
     ...: NBmodel.fit(training_x, training_y)
     ...: y_pred_NB = NBmodel.predict(test_x)
     ...: # evaluation
     ...: acc_NB = accuracy_score(test_y, y_pred_NB)
     ...: print("Naive Bayes model Accuracy:: {:.2f}%".format(acc_NB*100))
Naive Bayes model Accuracy:: 51.65%
```

```
In [255]: from sklearn.ensemble import RandomForestClassifier
     ...: RFmodel = RandomForestClassifier(n_estimators=500, max_depth=100,bootstrap=True, random_state=0) ## number of
trees and number of layers/depth
     ...: # training
     ...: RFmodel.fit(training_x, training_y)
     ...: y_pred_RF = RFmodel.predict(test_x)
     ...: # evaluation
     ...: acc_RF = accuracy_score(test_y, y_pred_RF)
     ...: print("Random Forest Model Accuracy: {:.2f}%".format(acc_RF*100))
Random Forest Model Accuracy: 54.23%
```

```
In [254]: from sklearn.svm import LinearSVC
     ...: SVMmodel = LinearSVC()
     ...: # training
     ...: SVMmodel.fit(training_x, training_y)
     ...: y_pred_SVM = SVMmodel.predict(test_x)
     ...: # evaluation
     ...: acc_SVM = accuracy_score(test_y, y_pred_SVM)
     ...: print("SVM model Accuracy:{:.2f}%".format(acc_SVM*100))
SVM model Accuracy:57.01%
```

```
In [266]: from sklearn.neural_network import MLPClassifier
     ...: DLmodel = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(50), random_state=1)
     ...: # training
     ...: DLmodel.fit(training_x, training_y)
     ...: y_pred_DL= DLmodel.predict(test_x)
     ...: # evaluation
     ...: acc_DL = accuracy_score(test_y, y_pred_DL)
     ...: print("Neural Network Model Accuracy: {:.2f}%".format(acc_DL*100))
Neural Network Model Accuracy: 49.90%
```