video gaming classification

Improving the Craigslist User Experience

Team Unstrucata

Meet the Unstrucata Analysts

Paul Chen

Soyeon Baik

Rachel Fagan

Jai Woo Lee

Vivek Rao

Evan Tiffany

Agenda

1 background & problem statement

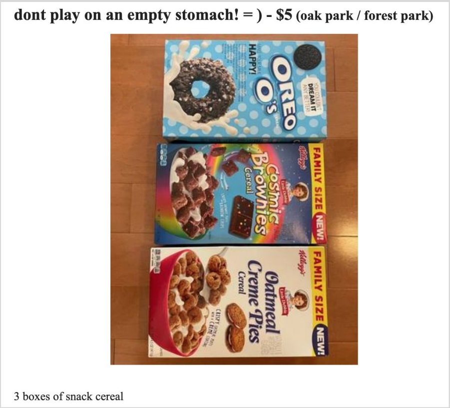
2 our analytical approach

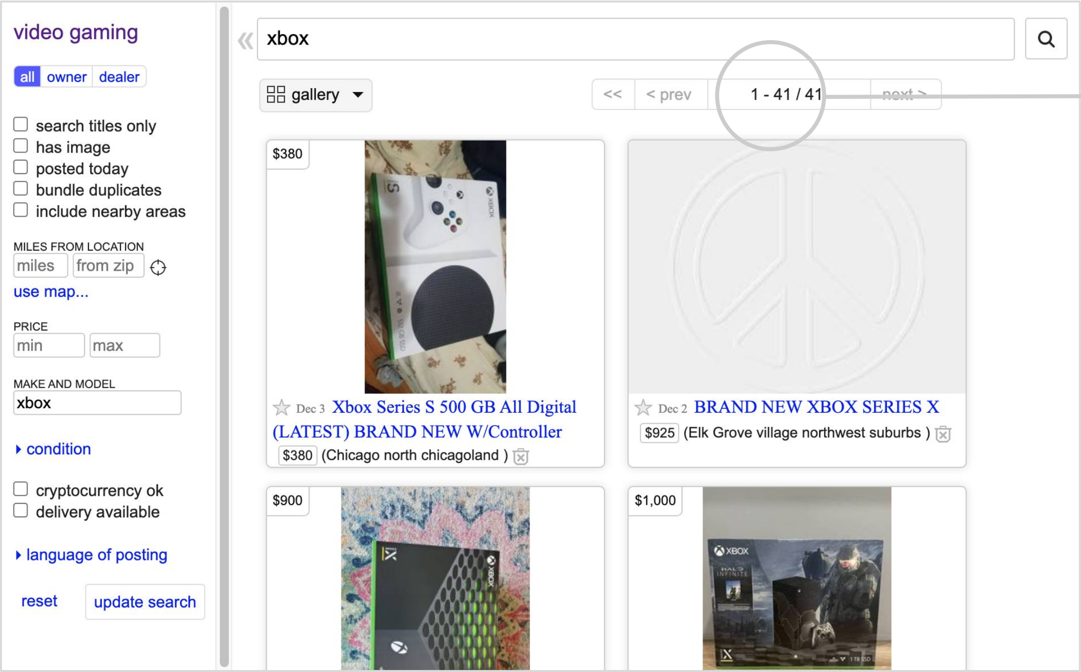
3 conclusion

background & problem statement

craigslist

Finding what you need in the video game category can be...challenging.



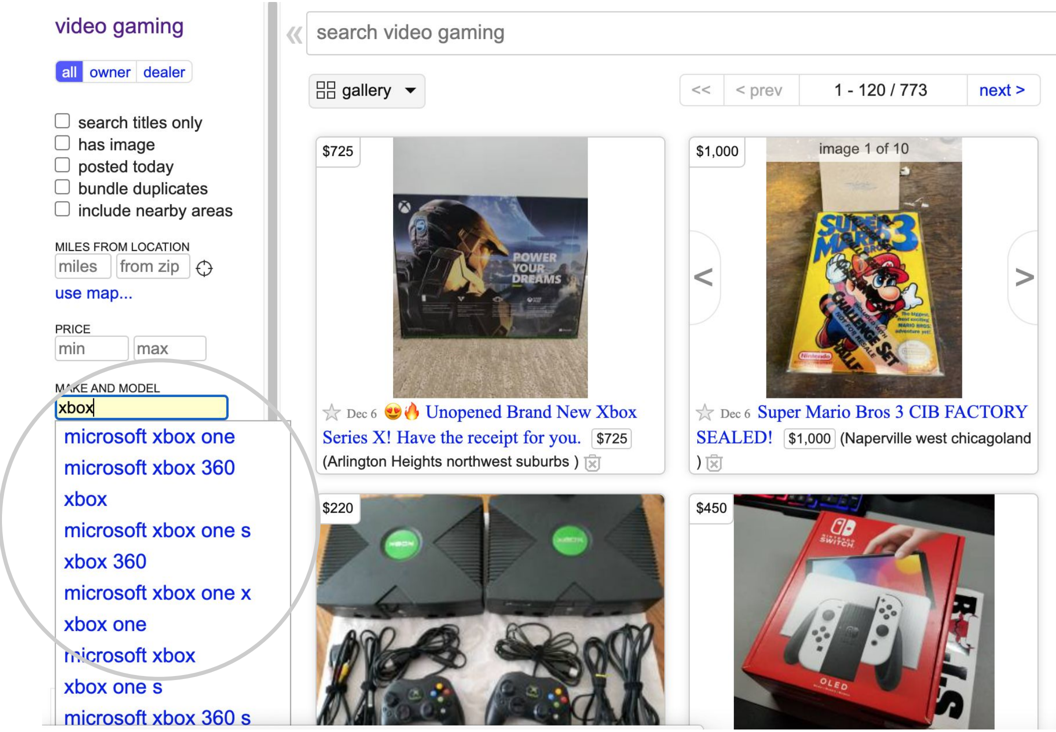


There are inconsistent filtering results.

This filter is showing that there are only **41** (when the general search shows **134**) xbox listings.

The search selection options are highly specific.

There are at least 5 different options if a user was looking for an xbox one.



Business Opportunity: Retention

The Video Gaming category does not generate revenue for craigslist.

By increasing customer satisfaction in the video gaming category with improved filters, we can increase the retention rate for the site overall, driving the company’s revenue.

our analytical approach

Our Analytical Approach | **Web Scraping**

Cities | New York, Chicago, Miami, Washington DC, Los Angeles, and Philadelphia

posting_id	datetime	city	title	price	place	desc
7414934575	12/1/2021 8:45	miami	Arcade Video Game Machine With Thousands Of Pre Loaded Retro Games	$700	(Cutler Bay)	QR Code Link to This Post PRICE IS FIRM This is a real Multi Game Arcade packed with over 10,000 games. Perfect for a kids room, personal bar or man cave. Includes games for the following systems. - Arcade games - Atari 2600 - Atari 5200 - Atari 7800 - Atari Jaguar - Intellivision - Colecovision - Nintendo Entertainment system - Super Nintendo Entertainment System - Nintendo Gameboy - Nintendo Gameboy Color

Our Analytical Approach | **Data Pre-Processing**

title	price	place	desc
sony playstation five - ps5 NEW	$850	(Fort Lauderdale)	QR Code Link to This Post Brand new sealed

title	price	place	desc
sony playstation five - ps5 new	850	Fort Lauderdale	brand new sealed
battlefield 2042	50	Hollywood	like brand new also still in plastic and for ps5 ,ps4 and xbox. c
playstation 5 disc sealed	850	Hollywood	brand new sealed, only willing to meet at a gas station.

- Removed the dollar sign in price column
- Removed parentheses in place column
- Made the description column one line
- Lowered the title case

Our Analytical Approach | **Keyword Extraction**

Filter by Brand

SONY

Nintendo

Microsoft

ARCADE

Meta

Other

	Microsoft	Nintendo	Sony	Arcade	Meta	Other
0	0	2	0	4	0	0.5
1	0	0	3	0	0	0.5
2	1	0	2	0	0	0.5
3	0	0	1	0	0	0.5
4	2	0	0	0	0	0.5
...	...	...	...	...	...	...
3006	0	0	0	0	0	0.5
3007	0	0	0	0	0	0.5
3008	0	0	0	0	0	0.5
3009	0	0	0	0	0	0.5
3010	0	1	0	0	0	0.5

Our Analytical Approach | **Model Building**

Target Variable

Brand Classification

Input Variables

Review Texts

Models

Naive Bayes

Logistic Regression

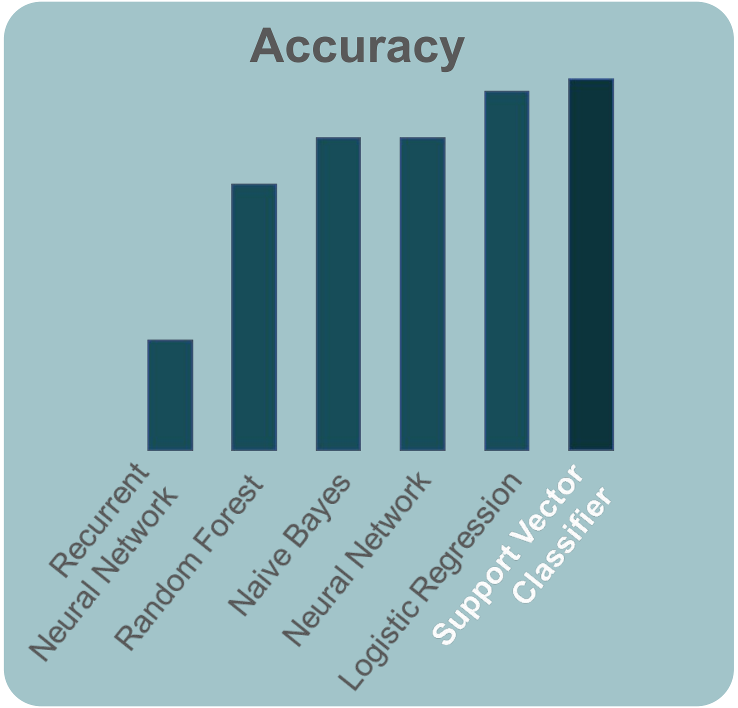
Random Forest

Support Vector Classification

Neural Network

Recurrent Neural Network

Our Analytical Approach | **Validation**



With our testing data,

SVC showed the highest accuracy with the score of **88.45%**

Logistic Regression | 84.99%

Neural Network | 74.24%

Naive Bayes | 73.97%

Random Forest | 63.75%

Recurrent Neural Network | 25.90%

Final Model: SVC (Support Vector Classification) model.

Confusion Matrix

	Arcade	Meta	Microsoft	Nintendo	Other	Sony
Arcade	43	0	0	1	1	1
Meta	0	4	1	3	3	3
Microsoft	0	0	147	0	4	6
Nintendo	0	0	1	173	13	8
Other	0	1	0	12	73	20
Sony	1	0	1	1	6	226

False Negative

False Positive

conclusion

Conclusion | **Impact: Information Gain**

Our Model → craigslist →

Problem Solved?

High cost for…

Customer to extract information

Conclusion | Impact: Information Gain

	Manual Search Rate	Model Classification Rate	Information Gain
SONY	54%	96%	77%
Nintendo	74%	89%	20%
Microsoft	92%	94%	1.5%
ARCADE	42%	93%	123%
Meta	55%	29%	-48%
Total	**68%**	**92%**	**34%**

Conclusion | **Impact: Information Gain**

Customer Experience

Retention Rate

Time on Site

Limitations

- The **amount of data we could scrape was limited** due to the Timeout Errors caused by Craigslist's servers.
- A filter-by-product classification was unrealistic without **manually entering the target variable** for each advertisement.
- Meta (Oculus) is an upcoming video gaming company, therefore the **data for these products is sparse** on Craigslist.

Looking Forward

- This feature could also be used within the **“posting details” user interface for advertisers**. It could automatically filter their listing into a brand category that would be confirmed by the advertiser.

- We recommend that Craigslist design and implement a filter-by-product option for customers. Customers could filter the Video Gaming Category for **consoles, accessories, or video games**.

- If Craigslist wished to continue computer-driven filtering, it could implement additional variables, like price, to best match the brand.

Thank you!

Any questions?

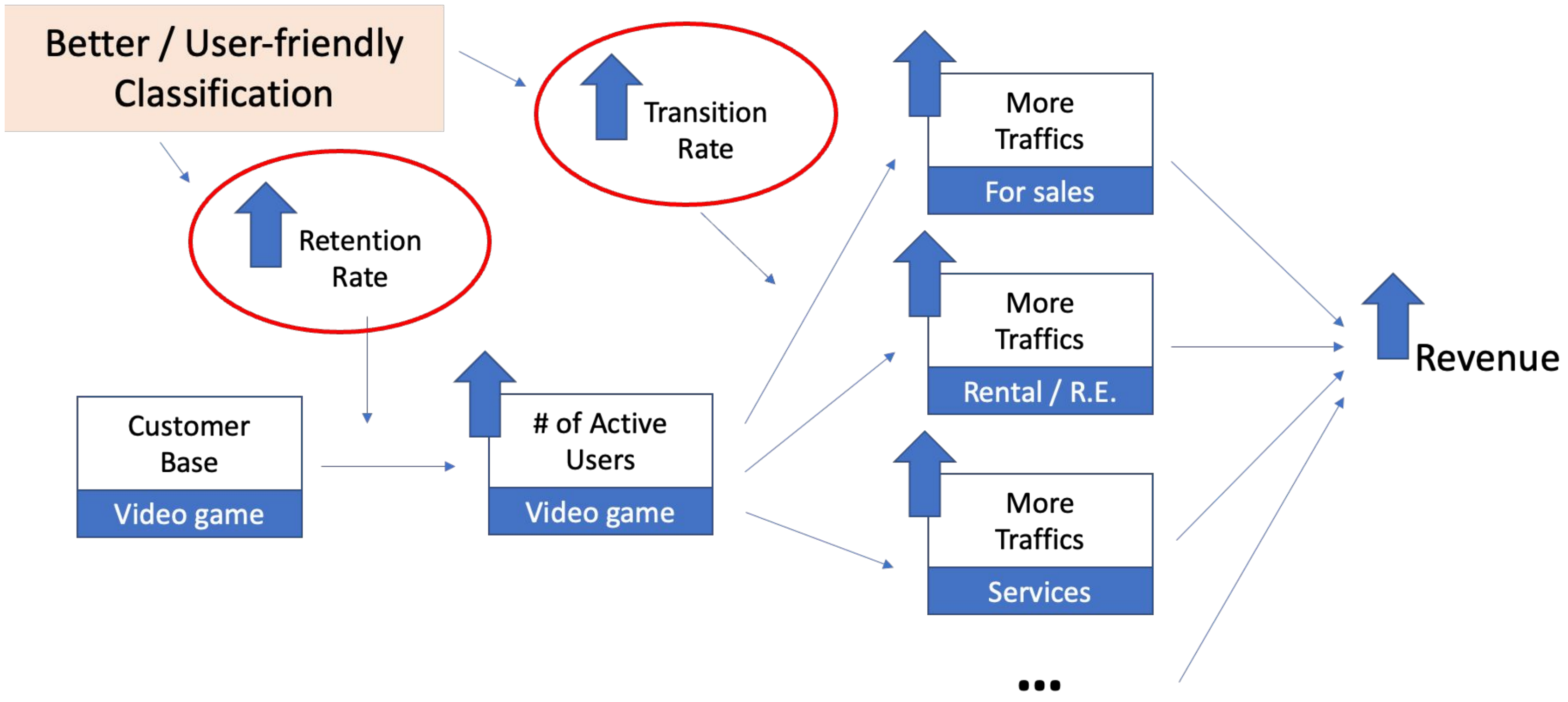
GameCube lot bro - $1,000 (Suburbs)

image 1 of 2

Lot of gamecube games . the Mario sunshine cases contain metroid prime echoes and the other is Luigi's mansion. The ereader cards are opened but complete in each pack. Don't low ball dude , I know what I got

appendix

Retention



Better / User-friendly Classification
Transition Rate
Retention Rate
Customer Base
Video game
# of Active Users
Video game
More Traffics
For sales
More Traffics
Rental / R.E.
More Traffics
Services
...
Revenue

Web Scraping Code

```
for city in cities:
    print(city)
    base_url = 'https://' + city + '.craigslist.org/search/vga'
    re = requests.get(base_url, headers=headers)

    soup = bs4.BeautifulSoup(re.text)

    # find the total number of pages for the city
    count = int(soup.select('.totalcount')[0].getText())
    num_pages = count // 120

    for page in range(num_pages):
        base_url = 'https://' + city + '.craigslist.org/search/vga?s=' + str(page*120)
        re = requests.get(base_url, headers=headers)
        soup = bs4.BeautifulSoup(re.text)

        # only use HTML tags of tags that have the 'result-image' tag
        soup = soup.select('.result-image')

        # create a list of all the links on the page
        links = [x.attrs['href'] for x in soup]

        # loop through each listing on this page
        for link in links:
            posting_re = requests.get(link)
            posting_soup = bs4.BeautifulSoup(posting_re.text)
```

Data Cleaning Code

```
newp = []
for p in allclean["price"]:
    newp.append(int(p.strip("$").replace(",", "")))
allclean["price"] = newp
```

```
descrip = []
for d in allclean["desc"]:
    des = ""
    for s in d.split("\n\n\n")[1].split("\n"):
        des += s
    descrip.append(des.lower())
allclean["desc"] = descrip
```

```
places = []
for p in allclean["place"]:
    places.append(p.strip(" (").strip(")"))
allclean["place"] = places
```

```
allclean["title"] = allclean["title"].str.lower()
```

```
brands = {"Microsoft": ["xbox", "microsoft", "360", "xbox one", " rig '
```

```
import numpy as np
for brand in brands:
    allclean[brand] = np.repeat(0, 3011)
    for word in brands[brand]:
        allclean[brand] += allclean.title.str.contains(word)
        allclean[brand] += allclean.desc.str.contains(word)
```

```
allclean["Other"] = np.repeat(0.5, 3011)
```

```
allclean["Brand"] = allclean.iloc[:,-6:].idxmax(1)
```

```
allclean.to_csv("allcleanwithbrand.csv")
```

```
allclean["Brand"].value_counts()
```

```
Sony         1012
Nintendo      755
Microsoft     643
Other         380
Arcade        177
Meta           44
Name: Brand, dtype: int64
```

Tokenization Code

```
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

lemmatizer = nltk.stem.WordNetLemmatizer()
tokencomp = []
for review in list(training_x):
    tokens = nltk.word_tokenize(str(review).lower())
    lemmatized_token = [lemmatizer.lemmatize(token) for token in tokens if token.isalnum()]
    tokencomp.append([token for token in lemmatized_token if token not in stopwords.words('english')])

comp = []
for review in tokencomp:
    comp.append(" ".join(review))
vectorizer = TfidfVectorizer(ngram_range = (1,2), min_df = 2)
vectorizer.fit(comp)

train_x = vectorizer.transform(training_x)
test_x = vectorizer.transform(testing_x)
```

Model Code

Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
NBmodel = MultinomialNB()

NBmodel.fit(train_x, train_y)
y_pred_NB = NBmodel.predict(test_x)

acc_NB = accuracy_score(test_y, y_pred_NB)
print("Naive Bayes model Accuracy:: {:.2f}%".format(acc_NB*100))
```

```
Naive Bayes model Accuracy:: 73.97%
```

Logistic Model

```
from sklearn.linear_model import LogisticRegression
Logitmodel = LogisticRegression()

Logitmodel.fit(train_x, train_y)
y_pred_logit = Logitmodel.predict(test_x)

acc_logit = accuracy_score(test_y, y_pred_logit)
print("Logit model Accuracy:: {:.2f}%".format(acc_logit*100))
```

```
Logit model Accuracy:: 84.99%
```

Model Code (Continued)

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

RFmodel = RandomForestClassifier(n_estimators=50, max_depth=6, bootstrap=True, random_state=0)

RFmodel.fit(train_x, train_y)
y_pred_RF = RFmodel.predict(test_x)

acc_RF = accuracy_score(test_y, y_pred_RF)
print("Random Forest Model Accuracy: {:.2f}%".format(acc_RF*100))
```

```
Random Forest Model Accuracy: 63.75%
```

SVC Model

```
from sklearn.svm import LinearSVC
SVMmodel = LinearSVC()

SVMmodel.fit(train_x, train_y)
y_pred_SVM = SVMmodel.predict(test_x)

acc_SVM = accuracy_score(test_y, y_pred_SVM)
print("SVM model Accuracy: {:.2f}%".format(acc_SVM*100))
```

```
SVM model Accuracy: 88.45%
```

Model Code (Continued)

Neural Network

```
from sklearn.neural_network import MLPClassifier
DLmodel = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(3,2), random_state=1)

DLmodel.fit(train_x, train_y)
y_pred_DL= DLmodel.predict(test_x)

acc_DL = accuracy_score(test_y, y_pred_DL)
print("DL model Accuracy: {:.2f}%".format(acc_DL*100))
```

```
DL model Accuracy: 74.24%
```

```
C:\Users\sthoy\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:500: ConvergenceWarning: lbfgs fail
ed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

Model Code (Continued)

Recurring Neural Network

```
import numpy as np

docs_x = []
docs_train_x = []
docs_test_x = []
for review in training_x:
    docs_x.append(nltk.word_tokenize(str(review).lower()))
    docs_train_x.append(nltk.word_tokenize(str(review).lower()))
for review in testing_x:
    docs_x.append(nltk.word_tokenize(str(review).lower()))
    docs_test_x.append(nltk.word_tokenize(str(review).lower()))

from collections import Counter
words = [j for i in docs_x for j in i]
count_words = Counter(words)
total_words = len(words)
sorted_words = count_words.most_common(total_words)
vocab_to_int = {w: i+1 for i, (w,c) in enumerate(sorted_words)}

text_int = []
for i in docs_train_x:
    r = [vocab_to_int[w] for w in i]
    text_int.append(r)


text_test_int = []
for i in docs_test_x:
    r = [vocab_to_int[w] for w in i]
    text_test_int.append(r)
```

Model Code (Continued)

```
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding, Flatten
from keras.layers import LSTM
max_features = total_words
maxlen = 250
batch_size = 32

x_train = sequence.pad_sequences(text_int, maxlen=maxlen)
x_test = sequence.pad_sequences(text_test_int, maxlen=maxlen)

encoded_train = [0 if label =='Sony' else 1 if label == "Nintendo" else 2 if label == "Microsoft" else 3 if label == "Arcade" els
encoded_test = [0 if label =='Sony' else 1 if label == "Nintendo" else 2 if label == "Microsoft" else 3 if label == "Arcade" else

model = Sequential()
model.add(Embedding(max_features, 20, input_length=maxlen))
model.add(LSTM(100, dropout=0.10, recurrent_dropout=0.10))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train.tolist(), encoded_train, batch_size=batch_size, epochs=2, validation_data=(x_test.tolist(), encoded_test))
```

Information Gain (Full Chart)

Brand	Manual		Classification		Manual Rate	Classification Rate	Information Gain
	Manually Typed	Total	Classfied Number(TP)	Total(Test)			
Sony	549	1012	226	235	54%	96%	77.28%
Nintendo	558	755	173	195	74%	89%	20.04%
Microsoft	593	643	147	157	92%	94%	1.53%
Arcade	74	177	43	46	42%	93%	123.59%
Meta	24	44	4	14	55%	29%	-47.62%
Total	**1798**	**2631**	**593**	**647**	**68%**	**92%**	**34.12%**

* Others excluded (Number of observations: 380)