Scam Classification

For Cars and Trucks Sales Ads on Craigslist

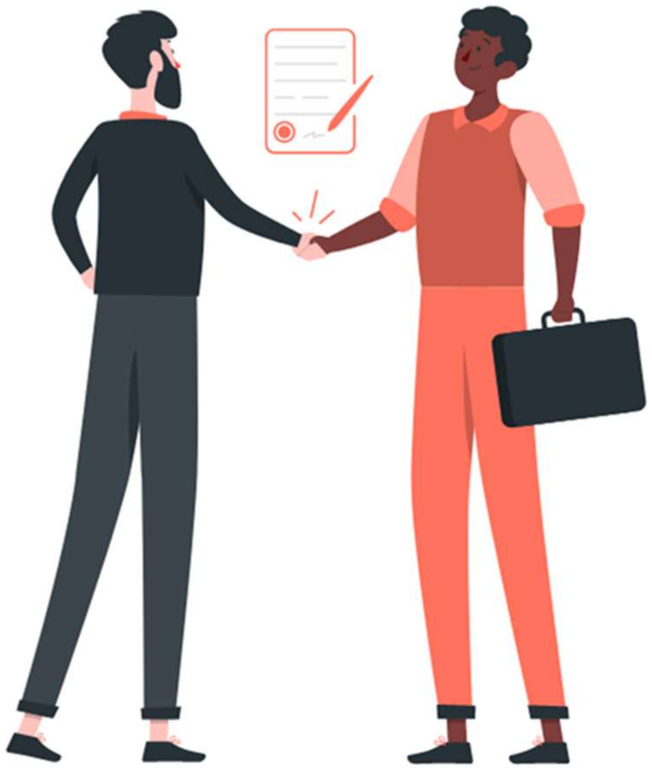
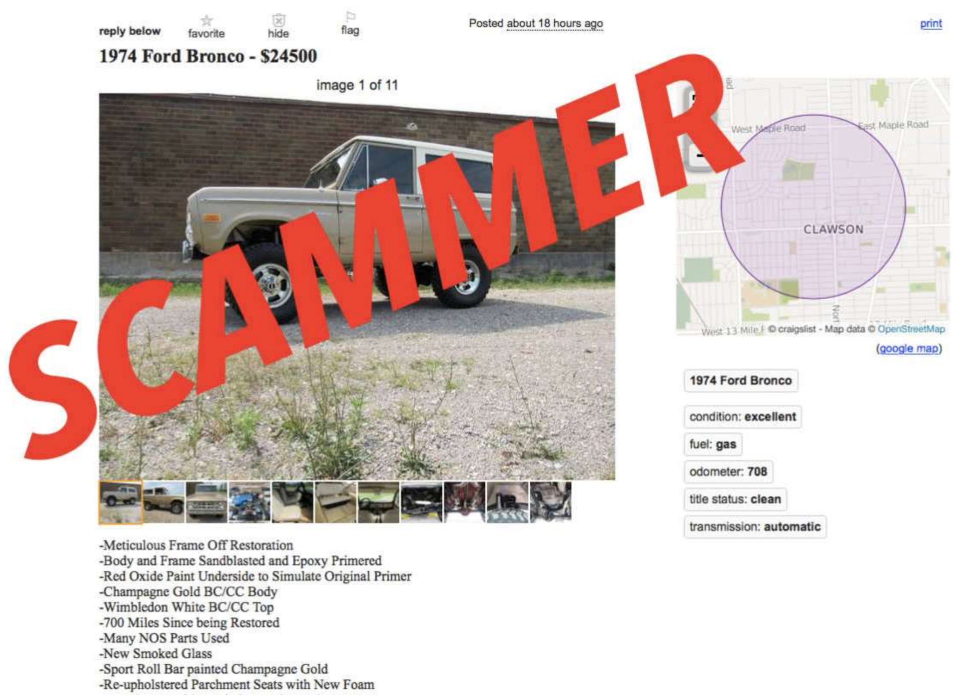


TABLE OF CONTENTS

PROBLEM STATEMENT

- Due to the high commercial potential, Craigslist is a **target for scammers**
- Scams **hinder the development** of online advertisement, as people **trust the process less**



ABOUT THE PROJECT

A scam classification model in Python which automatically detects if a particular ad listing is a scam or not based on pre-defined business rules

WHAT WE ARE WORKING ON

CARS

PICKUP TRUCKS

Cars/Pickup Trucks are one of the most selling items on Craigslist

Only focusing on these items being sold in the 60-mile radius of Chicago

PROJECT GOALS

Novel scam detection approach that could discriminate between scam and legitimate advertisement posts

Gain trust of audience

Increase the credibility of the platform

Sell more quickly

Genuine ads gather more traction from buyers

Shorter turnaround time for car buying

Hassle-free buying experience for the customers

Save people from getting scammed

Increase awareness within the customers

PROJECT METHODOLOGY

01 Scraping web data

Scrape the required data about all the listings from Craigslist

02 Data Cleaning

Pre-processed the data according to the correct data types

03 Manual Labelling

Classified ads manually, based on the conditions – absence of phone numbers and extreme prices

04 Dimension Reduction

Performed lemmatization and removed all stop words

05 Word Frequency Normalization

Implemented TF-IDF with 1-gram and 2-grams Minimum document frequesncy set as 3

06 Setting up Data

Split the data into 80% training sets and 20% testing sets

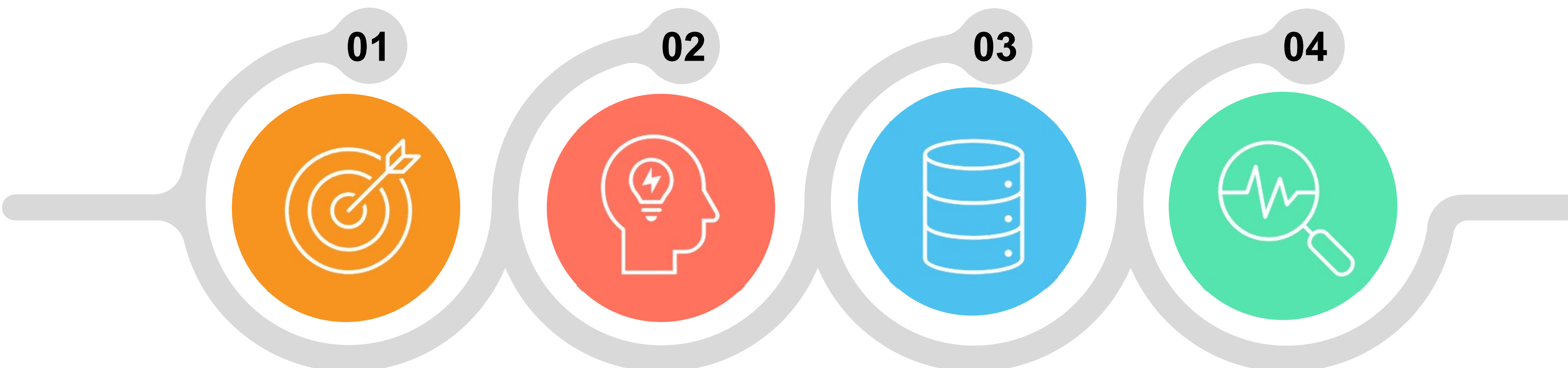
07 Applying machine learning and testing accuracy

XGB, RF, SVM, ET, Ensemble, LSTM

08 Topic Modelling

To identify features that are crucial for legitimate ads

WEB SCRAPING – DATA SOURCING



URL Extraction

Extracted URLs for each of the sales ad listings

Saving URLs

Saved each of the URLs in an Excel File

Data Extraction

Traversed through the saved URLs to get description, price, etc.

Saving data

Saved all the information scraped in an Excel file

We limited our search results to 25 pages (3000 results)

WEB SCRAPING CODE – URL EXTRACTION

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
first_part_url="https://chicago.craigslist.org/search/chicago-il/cta?"
URL=first_part_url+"lat=41.7434&lon=-87.7104&search_distance=60"
page=requests.get(URL)


url_list=[]
soup = BeautifulSoup(page.content,'html.parser')
results = soup.find(class_='rows')

car_elems = results.find_all('li',class_='result-row')


for car_elem in car_elems:
   url_elem=car_elem.find('a',class_='result-title hdrlnk')['href']
   url_list.append(url_elem)
```

```
for i in range(1,25):

   next_url="s="+str(120*i)+"&"
   URL=first_part_url+next_url+"lat=41.7434&lon=-87.7104&search_distance=60"
   page=requests.get(URL)
   soup = BeautifulSoup(page.content, 'html.parser')
   results = soup.find(class_='rows')
   car_elems = results.find_all('li',class_='result-row')
   for car_elem in car_elems:
#    price_elem=car_elem.find('span',class_='result-price')
      url_elem=car_elem.find('a',class_='result-title hdrlnk')['href']
      url_list.append(url_elem)

print(len(url_list))

df = pd.DataFrame(url_list)
df.columns=["URL"]
writer = pd.ExcelWriter('URL_List.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='urlList', index=False)
writer.save()
```

WEB SCRAPING CODE – DATA EXTRACTION

```
import time

import requests
from bs4 import BeautifulSoup
#import numpy as np
import pandas as pd

#startpage
car_name=[]
price_list=[]

description=[]
df=pd.read_excel("URL_List.xlsx")
urls=list(df['URL'])
output = pd.DataFrame()

#print(urls)
for url in urls:
    page=requests.get(url)
    time.sleep(1)
    soup=BeautifulSoup(page.content,'html.parser')
    try:
        try:
            heading=soup.find('span',{'id':'titletextonly'})
            car_name.append(heading.text)
        except:
            car_name.append("")
        #print(heading.text)
        try:
            price=soup.find('span',class_='price')
            price_list.append(price.text)
        except:
            price_list.append("")
        #print(price.text)
        try:
            attributes=soup.find_all('p',class_='attrgroup')

            attr=[]
            for attribute in attributes:
                spans=attribute.find_all('span')
                for span in spans:
                    text = span.text.strip()
                    #print(text)
                    attr.append(text)
        except:
            attr.append("")
        #print(attr)

        attrsplit = [item.split(':') for item in attr]
        del attrsplit[0]
        #print(attrsplit)

        attrdict={}

        for item in attrsplit:
            attrdict[item[0]]=item[1:]

        #print(attrdict)


        #final_data=pd.DataFrame.from_dict(attrdict)
        #df_dictionary = pd.DataFrame([attrdict])
        output = output.append(attrdict, ignore_index=True)
        #df_dictionary = pd.DataFrame(attrdict)
        #output = pd.concat([output, df_dictionary], ignore_index=True)
        try:
            body=soup.find('section',{'id':'postingbody'})
            description.append(body.text)
        except:
            description.append("")
        #print(body.text)
    except:

        continue
    #print(body.text)
    print("................//NEXT......")

#print(output.head())

#print(output.head())

df1 = pd.DataFrame(output)

consolidated_v1 = pd.DataFrame(list(zip(car_name,
urls,price_list,description)),columns =['Car Name', 'Car URL','Price',
'Description'])
result=pd.concat([consolidated_v1,df1], axis=1, sort=False)

#df.columns=["URL"]
writer = pd.ExcelWriter('data from url4.xlsx', engine='xlsxwriter')
result.to_excel(writer, sheet_name='data', index=False)
writer.save()
```

DATA DESCRIPTION

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Car Name           3000 non-null   object
 1   Car URL            3000 non-null   object
 2   Price              2882 non-null   float64
 3   Description        3000 non-null   object
 4   cylinders          2169 non-null   object
 5   fuel               3000 non-null   object
 6   odometer           2999 non-null   object
 7   title status       2948 non-null   object
 8   transmission       2996 non-null   object
 9   type               2627 non-null   object
 10  drive              2208 non-null   object
 11  paint color        2324 non-null   object
 12  condition          1646 non-null   object
 13  size               503 non-null    object
 14  VIN                1540 non-null   object
 15  Clean_Description  3000 non-null   object
 16  Call_Text_Flag     3000 non-null   bool
dtypes: bool(1), float64(1), object(15)
memory usage: 378.1+ KB
```

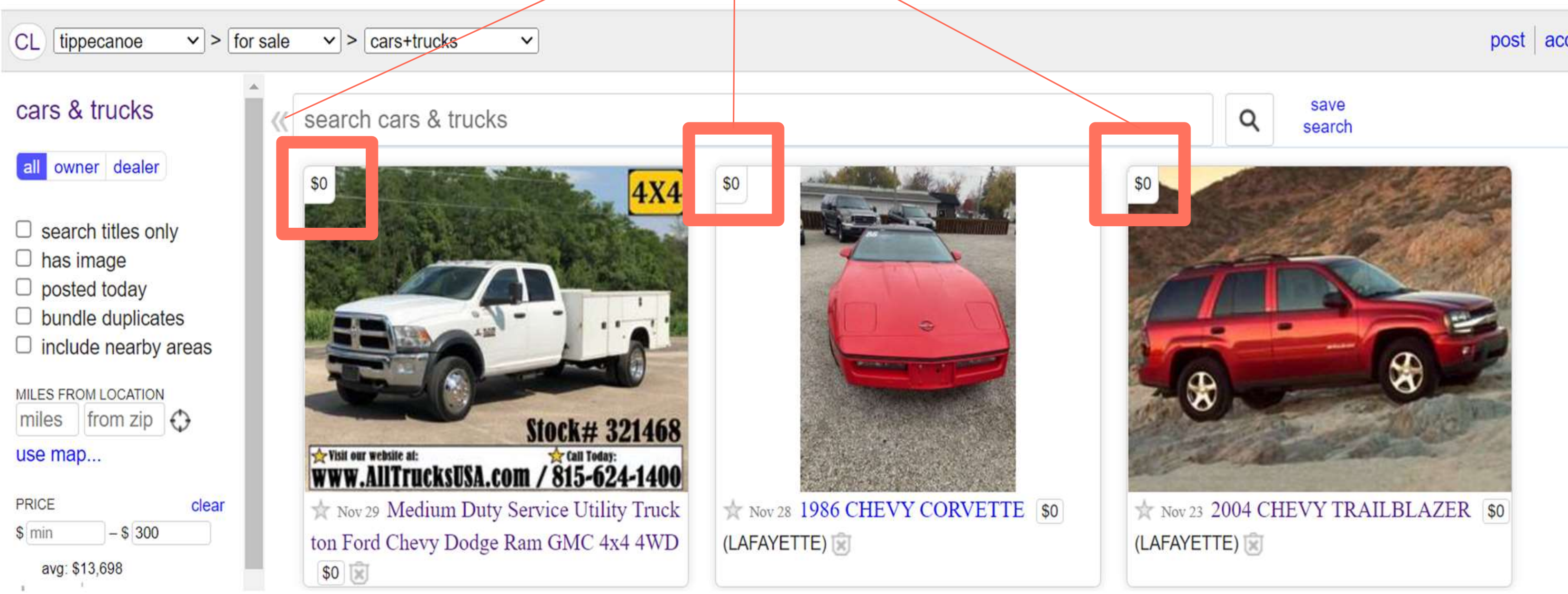
Used heuristic method to label scam[1]

- Price - It is unusual for someone for someone to advertise product with significantly high or low prices
- Description – It is often suspicious for someone to not provide contact information

[1]Alsaleh, Hamad, and Lina Zhou. "A Heuristic Method for Identifying Scam Ads on Craigslist." *2018 European Intelligence and Security Informatics Conference (EISIC)*, 2018, https://doi.org/10.1109/eisic.2018.00019.

BUSINESS ANALYSIS

Prices are listed as $0



FIELDS USED FOR MODELLING

```
scam_final
```

	Clean_Description	Probable_Scams
0	qr code link to this post 2007 volvo xc70 stat...	False
1	qr code link to this post 2006 lexus gs300 mil...	False
2	qr code link to this post this 2018 toyota rav...	False
3	qr code link to this post selling my 2010 focu...	False
4	qr code link to this post hello i have a 2008 ...	False
...	...	...
2995	qr code link to this post 2016 mercedes-benz c...	False
2996	qr code link to this post 2016 land rover rang...	False
2997	qr code link to this post 2016 keystone impact...	False
2998	qr code link to this post 2016 gmc savana 2500...	False
2999	qr code link to this post 2016 ford explorer p...	False

2445 rows × 2 columns

MODEL FLOW TESTING TRAINING

Total Dataset:
2445 unique records

Probable Scam
True : 77 records
False : 2368 records

=

Training Set:
1956 records

Probable Scam
True : 64 records
False : 1892 records

+

Validation Set:
489 records

Probable Scam
True : 13 records
False : 476 records

TEXT PRE-PROCESSING

We performed the following pre-processing of description column before feeding it to the model:

- Punctuation and common words removal
- Tokenization
- Lemmatization
- Stop words Removal
- TF-IDF Vectorization

```
X=scam_final[['Clean_Description']]
y=scam_final[['Probable_Scams']]



lemmatizer = nltk.stem.WordNetLemmatizer()

#Tokenizing the description
X['Clean_Description']=X['Clean_Description'].apply(lambda x: nltk.word_tokenize(x))

#Lemmatizing the description
X['Clean_Description']=X['Clean_Description'].apply(lambda x :[lemmatizer.lemmatize(item) for item in x if item.isalnum()])

#Removing some common occuring words across the descriptions
X['Clean_Description']=X['Clean_Description'].apply(lambda x:[item for item in x if item not in ['qr','code','link','post','this

#Removing Stop Words
from nltk.corpus import stopwords
stop=stopwords.words('english') #Corpus
X['stopwords_removed']=X['Clean_Description'].apply(lambda x:[item for item in x if item not in stop if item.isalnum()])
#Joining back the tokens
X['final']=X['stopwords_removed'].apply(lambda x: " ".join(x))

X=X.drop(['Clean_Description','stopwords_removed'],axis=1)

# create text variable
X_text = X['final']

# TF-IDF Vectorization for weighted frequency of words and transform into vector of 1/0
tvf = TfidfVectorizer(stop_words=stopwords.words('english'),min_df=8,ngram_range=(1,2),lowercase=False)
X_text = tvf.fit_transform(X_text)
print(tvf.vocabulary_)
```

CLASSIFICATION MODELS

```
#Random Forest Classifier
clf2 = RandomForestClassifier (n_estimators=300, n_jobs=-1,max_depth=8,max_features=200)
clf2.fit(X_train,y_train)
predict_rf=clf2.predict(X_test)
summary_statistics['random_forest']=roc_auc_score(y_test,predict_rf)

#Extra Tree Classifier
clf4 = ExtraTreesClassifier(n_estimators=100, n_jobs=-1, criterion='entropy')
clf4.fit(X_train,y_train)
predict_extratree=clf4.predict(X_test)
summary_statistics['extra_tree']=roc_auc_score(y_test,predict_extratree)

#Support Vector Classifier
clf5 = SVC()
clf5.fit(X_train,y_train)
predict_svm=clf5.predict(X_test)
summary_statistics['svm']=roc_auc_score(y_test,predict_svm)


#XG Boost RF Classifier
clf10=XGBRFClassifier(learning_rate=0.05,n_estimators=200,max_depth=6)
clf10.fit(X_train,y_train)
predict_xgb=clf10.predict(X_test)
summary_statistics['xgb']=roc_auc_score(y_test,predict_xgb)


#Voting Classifier
predictors=[ ('RF_E', clf2), ('ET_e', clf4),
        ('svc', clf5),("XGB",clf10)]


# building voting
VT=VotingClassifier(predictors)
VT.fit(X_train,y_train)
predicted_VT=VT.predict(X_test)
summary_statistics['VT_classifier']=roc_auc_score(y_test,predicted_VT)
```

```
from keras import Sequential
from keras.layers import SpatialDropout1D
# Define parameter
embedding_dim = 16
drop_value = 0.2
n_dense = 24
n_lstm = 128
drop_lstm = 0.2
# Define LSTM Model
model1 = Sequential()
model1.add(tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_len))
#model1.add(SpatialDropout1D(drop_lstm))
model1.add(tf.keras.layers.LSTM(n_lstm, return_sequences=False))
#model1.add(tf.keras.layers.Dropout(drop_lstm))
model1.add(tf.keras.layers.Dense(1, activation='sigmoid'))

print(model1.summary())

model1.compile(loss = 'binary_crossentropy',
               optimizer = 'adam',
               metrics = ['AUC'])

num_epochs = 30
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_AUC', patience=5)
history = model1.fit(training_padded,
                     y_train,
                     epochs=num_epochs,
                     validation_data=(testing_padded, y_test),
                     verbose=2)
```

We implemented the following models for classifying probable scams based on description of Ads:

- **Classical Models :** Random Forest | Extra Tree Classifier | Support Vector Machine | XG Boost | Classifier | Voting Ensemble Classifier
- **Advanced Model :** Recurrent LSTM

MODEL EVALUATION

ML Algorithms	Test ROC AUC Score
Random Forest	0.57
Extra Tree	0.65
Support Vector	0.57
XG Boost	0.68
Voting Ensemble	0.61
Recurrent LSTM NN	**0.74**

Predicted Labels

True Labels	False	True
False	466	7
True	11	5

Confusion Matrix for LSTM

IDENTIFYING LEGITIMATE LISTINGS

Data Processing: Identifying Noun Phrases using n-grams

Bigram:

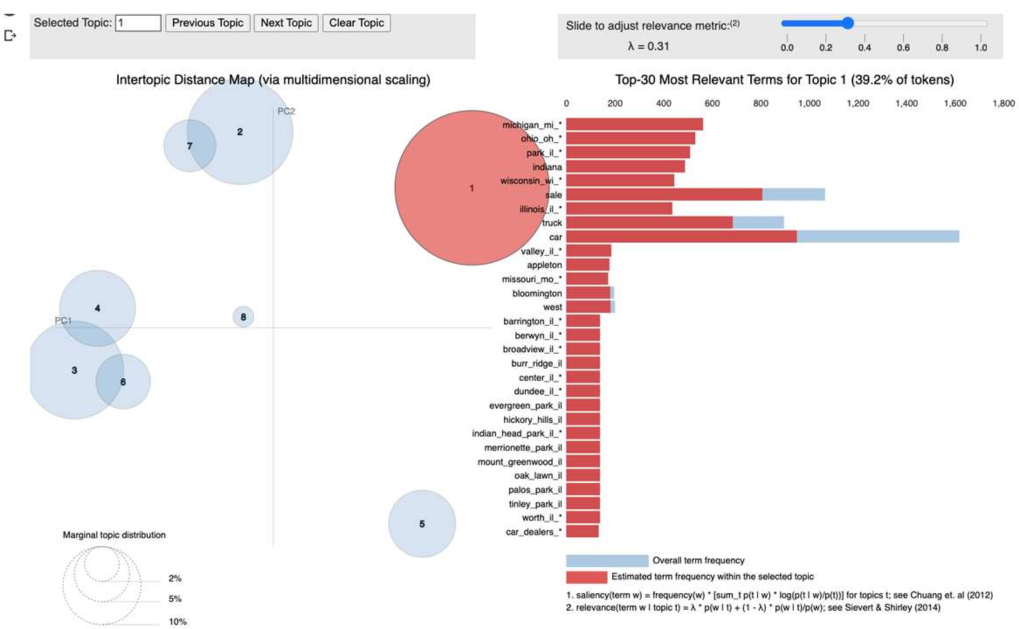
```
bigrams[:10]

['spoilerrear spoilerside',
 'legible fashion',
 'conspicuous legible',
 'xv crosstrek',
 'representation expressed',
 'existence ownership',
 'document preparation',
 'posted accurate',
 'brakesfog lightsintermittent',
 'mirrorsother featuresnavigation']
```

Trigram:

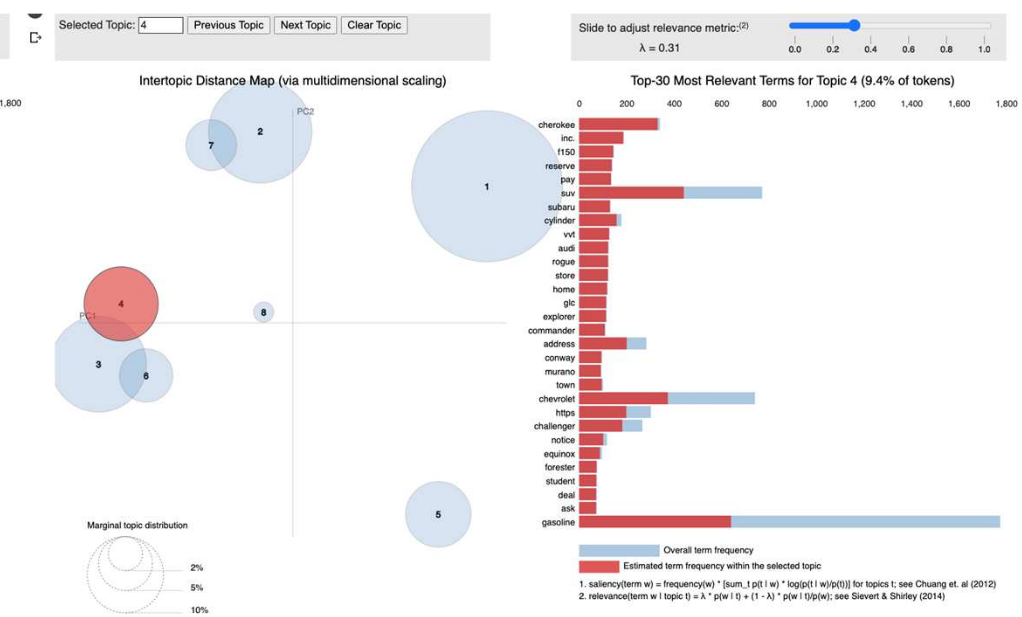
```
trigrams[:10]

['_bad_credit_ok _bad_credit_ok _bad_credit_ok',
 'santa fe sport',
 'first time buyer',
 'old mill creek',
 'country club hills',
 'waukegan auto auction',
 'champaign urbana illinois',
 'south dakota sd',
 'actual low miles',
 'south bend michiana']
```

IDENTIFYING LEGITIMATE LISTINGS

Topic 1: Locations Reported in Listings



Topic 4: Specific Car Models and Brands

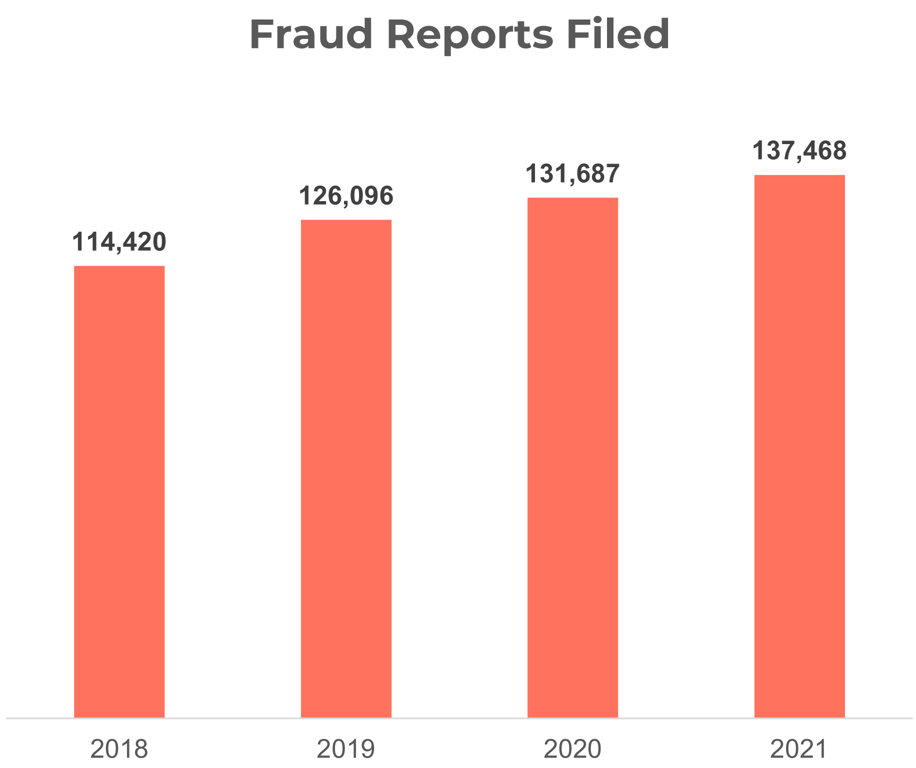


NEXT STEPS

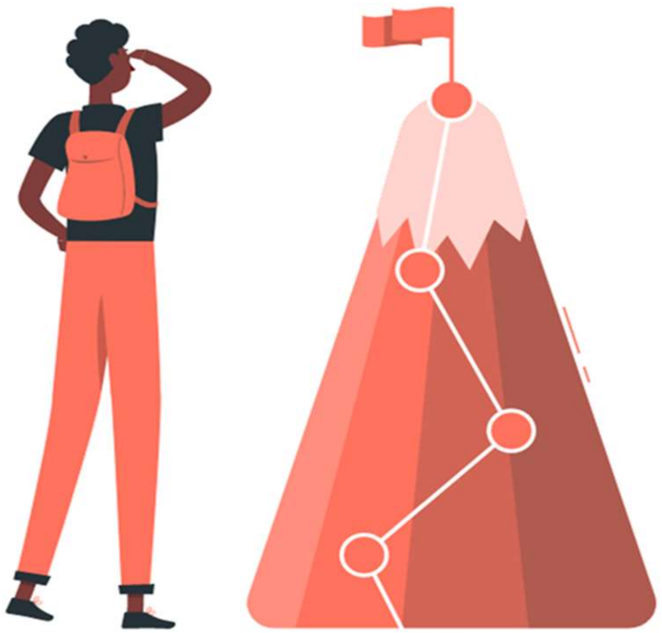
- For future work, we plan to extend the experimental dataset.
- Since this is a highly imbalanced classification problem, if we randomly pick a sample of instances for judgment, there are very few scam instances in the sample.
- To overcome this, we plan to use other advanced models to pick instances that are likely to be scam for judgment.

RECOMMENDATIONS AND CONCLUSION

- FTC reported that ~2.8 million people were scammed in the year 2021 alone.[1]
- Auto related scams also amounted to ~137k reports being filed.[1]
- This brings in a need for Craigslist to not only host credible sellers but also regularly scrutinize the ads posted on the platform.



1. Consumer Sentinel Report : FTC



THANK YOU