Project report:

- Problem statement:

Create a complaint resolution workflow by classifying consumer finance complaints into 12 pre-defined classes. The data can be downloaded from data.gov. We can assume each new complaint is assigned to one and only one category.

Finance complaint resolution department is the client. As consumer complaints comes in we can assign the best suitable category to it and add it to appropriate person's queue.

- Data set description, and data wrangling process:

Data set is published at Data.Gov by Consumer Financial Protection Bureau in comma separated value format. Given dataset has 18 columns

''Date received', 'Product', 'Sub-product', 'Issue', 'Sub-issue', 'Consumer complaint narrative', 'Company public response', 'Company', 'State', 'ZIP code', 'Tags', 'Consumer consent provided?', 'Submitted via', 'Date sent to company', 'Company response to consumer', 'Timely response?', 'Consumer disputed?', 'Complaint ID'

But for this project we need only two columns product and consumer complaint narrative.

Product is our target variable and consumer complaint narrative is input.

I removed missing values in consumer complaint narrative and converted categorical column product into integer.

We cannot process narrative text as is. Most of the ML algorithms expect numerical feature values of fixed size. Most common approach is to use the bag of words model. We add weight to each word based on it's occurrence (frequency). We will consider unigrams (single word) and bigrams (pair of words. Most common words that follows). We remove stop words like "a", "the", "an"

Along with dataset we can use census data and geo location data to see which locations and demographics are facing maximum issues.

- Initial findings:

Data set is imbalance. Maximum complains are related to Mortgage, Credit reporting, credit repair services, or other personal consumer reports and Debt collection. It becomes difficult to implement ML models with such dataset. If not configured correctly models tend to give lots of false positives.

- In-depth analysis:

As mentioned earlier we are working with consumer complaint narrative and products column only.

Let's work on Product column first. There are 12 categories of products.

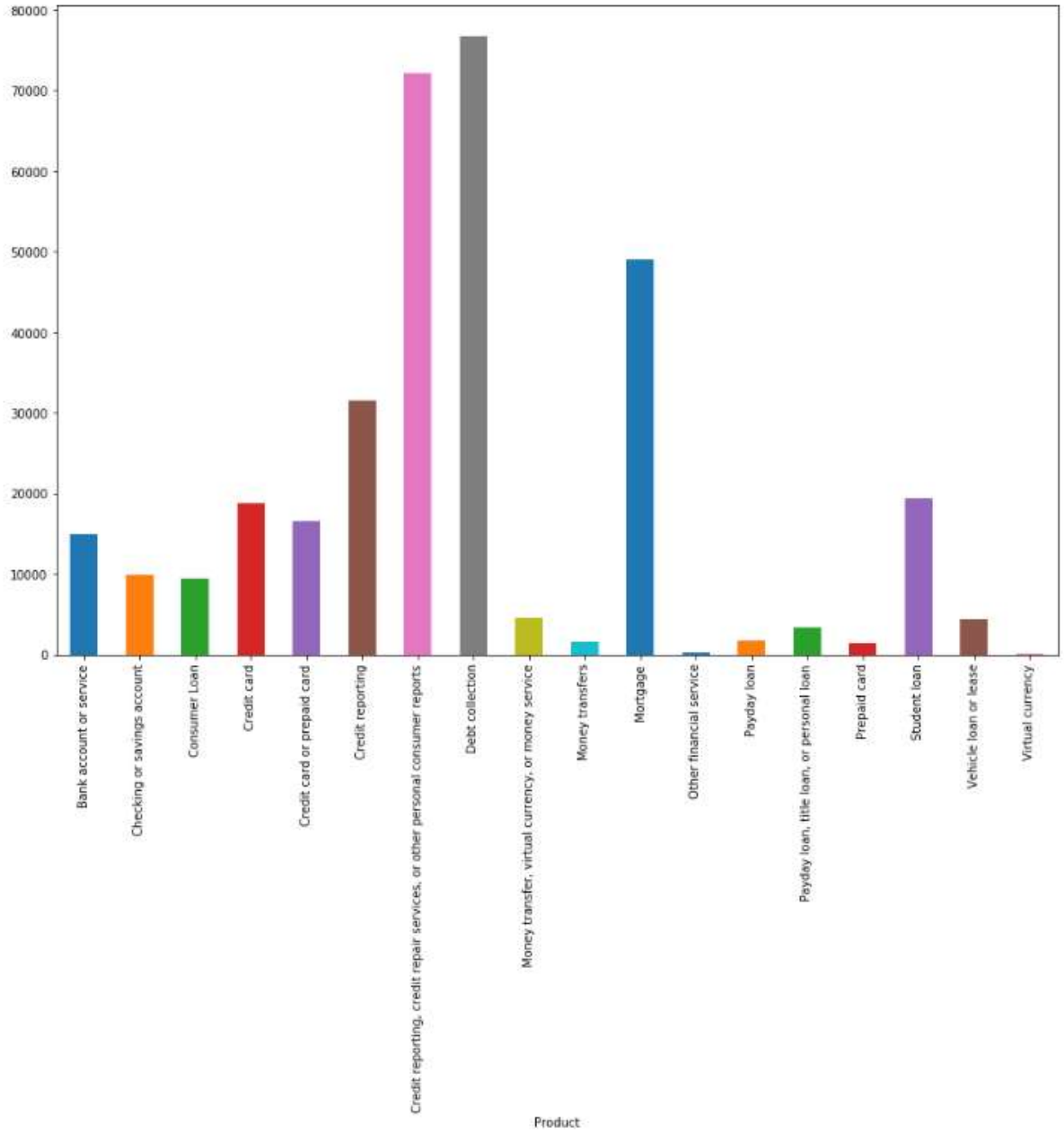
Product	Counts	Percent of Total
Virtual currency	16	0.004766742
Other financial service	292	0.086993049
Prepaid card	1450	0.431986033
Money transfers	1497	0.445988339
Payday loan	1747	0.52046869
Payday loan, title loan, or personal loan	3335	0.993567877
Vehicle loan or lease	4380	1.304895742
Money transfer, virtual currency, or money service	4468	1.331112826
Consumer Loan	9473	2.822209445
Checking or savings account	9826	2.9273757
Bank account or service	14885	4.434560074
Credit card or prepaid card	16489	4.912426004
Credit card	18838	5.612243378
Student loan	19385	5.775206385
Credit reporting	31588	9.410741258
Mortgage	49047	14.61215102
Credit reporting, credit repair services, or other personal consumer reports	72176	21.50277514
Debt collection	76767	22.8705323
Total	**335659**	**100**

After removing missing data, I encoded product column as in integer to make it categorical variable. I created two mapping dictionaries –

1) category_id_dataset – mapping between encoded category id and actual description

2) category_to_id_dataset - reverse mapping of category_id_dataset ie. Description to id mapping

More than 50% of the complaints are related to Mortgage, Credit reporting, cr edit repair services, or other personal consumer reports, and Debt collection.



Moving on to descriptive column, Consumer complaint narrative.
For ML algorithm to understand the contains of the column I have to convert this column in some kind of numerical format. Most common approach is to use bag of words model. In this model a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

Eg.

```
(1) John likes to watch movies. Mary likes movies too.
(2) John also likes to watch football games.
```

Based on these two text documents, a list constructed as follows for each document:

```
"John","likes","to","watch","movies","Mary","likes","movies","too"
"John","also","likes","to","watch","football","games"
```

Representing each bag-of-words as

```
BoW1 = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1};
BoW2 =
{"John":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1};
```

Each key is the word, and each value is the number of occurrences of that word in the given text document.
The order of elements is free.

There are couple of ways to implement this model:

1) `sklearn.feature_extraction.text.TfidfVectorizer` to calculate a `tf-idfvector` for each of consumer complaint narratives
2) Use `word_tokenize` from `nltk` library

To make ML process more efficient I took these steps:

1) Remove stop words:
    a. Generally, stop words does not add any value in ML and can skew the results as they have high frequency.

2) ngrams:
    a. With ngram technique I identified most commonly used words, words that occur in pairs (correlated). This is also useful to identify negations. Using “Not” can change the meaning of the sentence.
3) Steming:
    a. Derive the base work from variant form of the work.
       Eg. Playing -> play

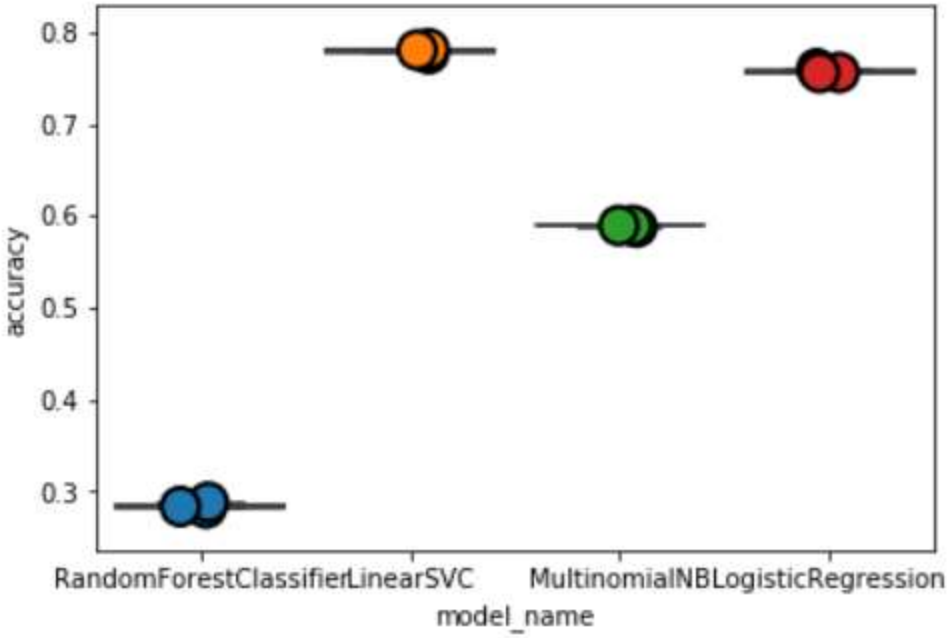
I used `sklearn.feature_selection.chi2` to find the words that are the most correlated with each of the products.

Eg.

```
# 'Bank account or service':
  . Most correlated unigrams:
. dir. angrily
  . Most correlated bigrams:
. hold week. live outside
# 'Checking or savings account':
  . Most correlated unigrams:
. tablet. inexperience
  . Most correlated bigrams:
. xxxx duty. company advising
# 'Consumer Loan':
  . Most correlated unigrams:
. wont. siding
  . Most correlated bigrams:
. requested taken. identity security
```

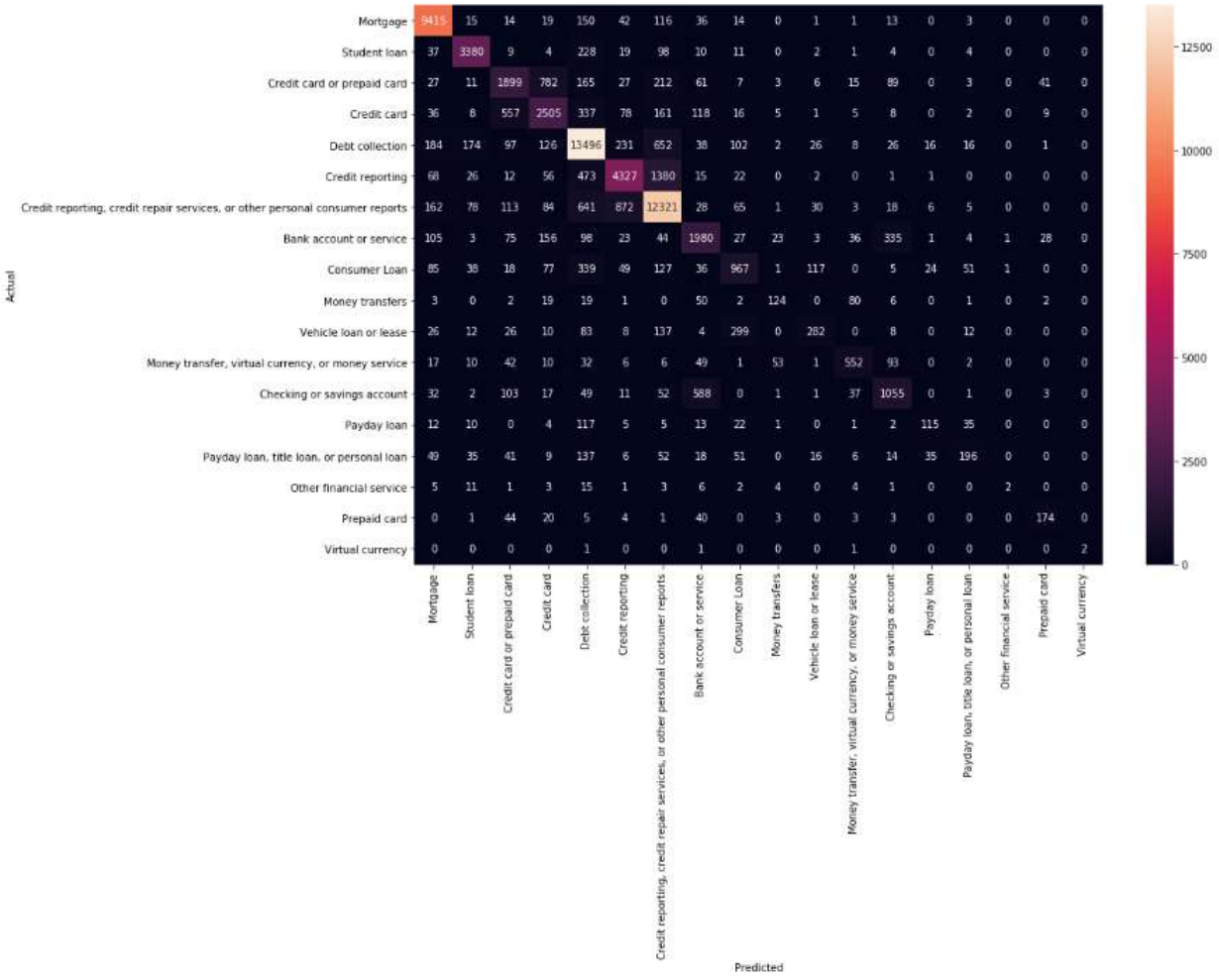
After all these transformations I used `TfidfVectorizer` to create sparse matrix X and y is category id.

I have everything to try multiple machine learning algorithms on this dataset. Using cross validation with 3 folds I ran RandomForestClassifier, LinearSVC, MultinomialNB (Naive Bayes Classifier) and LogisticRegression.



With 77% accuracy, LinearSVC model performed the best of out all.

I split the data into Train (80%) and test (20%) to test the model and print confusion matrix.



As expected most of the predictions are correct. There are few misclassifications as well. I believe that it because some complaints can be classified in multiple categories.
Here is classification report:

	precision	recall	f1-score	support
Student loan	0.92	0.96	0.94	9839
Credit card or prepaid card	0.89	0.89	0.89	3807
Mortgage	0.62	0.57	0.59	3348
Credit reporting	0.64	0.65	0.65	3846
Credit reporting, credit repair services, or other personal consumer reports	0.82	0.89	0.85	15195
Debt collection	0.76	0.68	0.72	6383
Vehicle loan or lease	0.80	0.85	0.83	14427
Money transfer, virtual currency, or money service	0.64	0.67	0.66	2942
Checking or savings account	0.60	0.50	0.55	1935
Payday loan, title loan, or personal loan	0.56	0.40	0.47	309
Consumer Loan	0.58	0.31	0.40	907
Bank account or service	0.73	0.63	0.68	874
Credit card	0.63	0.54	0.58	1952
Money transfers	0.58	0.34	0.43	342
Prepaid card	0.59	0.29	0.39	665
Payday loan	0.50	0.03	0.06	58
Other financial service	0.67	0.58	0.63	298
Virtual currency	1.00	0.40	0.57	5
avg / total	0.78	0.79	0.78	67132

Conclusion:
With 77% accuracy we can assign correct product category to a consumer complaint.

Future Development:

If I have more resources and machine power then I can implement Gridsearch, steming and multi label classification.