Capstone Project: Sprint 2

CSDA1050, York University School of Continuing Studies

September 1, 2018

PulseInsights:

Claire Chong
Bryan Kuropatwa
Hardeep Mundair
Edward O'Neil

Contents

Business Understanding

PulseInsights

A newly formed start-up company at the intersection of healthcare and insurance, our focus is building big data solutions to showcase our capabilities to help healthcare insurance companies improve their decision making. As a small team focused on minimizing costs, we will rely on Medicare's publicly available healthcare provider data. The goal of this project is to showcase our solution to the Centers for Medicare & Medicaid Services (CMS), the administrator of Medicare, as a potential client.

Medicare

Medicare is a federal health insurance program in the US for people who are 65 years or older, certain younger people with disabilities, and people with end stage renal disease. The different parts of Medicare cover different services. Part A is hospital insurance, Part B is medical insurance, Part C covers health plans offered by private companies in contract with Medicare and Part D relates to prescription drug coverage.

Combining two trust funds through the US Treasury, the program cost $549 billion in 2011 and covers three different levels of clients: the pharmaceutical companies producing the drugs, the physicians prescribing them, and the individuals buying and using the drugs.

Medicare fraud costs taxpayers money from the US Treasury, diminishes the legitimacy of emerging universal health care in the USA, and costs man-hours (as well as profits) from medical-based organizations including hospitals, drug manufacturers, insurance providers, as well as small practice medical professionals. We aim to develop a tool that can identify fraudulent prescription transactions to combat Medicare fraud.

Business Problem Statement

Identify the existence of fraudulent prescriptions initiated by the physician, by the patient or by the physician and patient.

Analytical Solution

Find the prescriptions whose characteristics exhibit that of a fraudulent nature by applying descriptive analytical techniques. Apply machine learning using predictive modelling on known, past fraudulent prescriptions to identify whether other prescriptions are fraudulent as well.

Analytical Problem Objective

It is useful to determine if fraudulent prescriptions share common characteristics, originate from a certain healthcare speciality or have other distinguishing traits that can help CMS detect these offenders in advance. Advance detection will help CMS improve their decision-making process and accurately approve or reject a claim while providing them information about healthcare practitioners that may offend in the future. This could result in savings for them and revenue for us if they are our customer.

Data Understanding

The following 4 datasets were explored:

1. Medicare provider utilization and payment data (**Part D Prescriber**)
    - Sourced from Centers for Medicare and Medicaid Services[1]
    - Contains information about prescription drug events incurred by Medicare beneficiaries enrolled in the Medicare Part D prescription drug program.
    - A data file exists for each year from 2013 to 2016 inclusive and includes the following attributes:

npi	int64
nppes_provider_last_org_name	object
nppes_provider_first_name	object
nppes_provider_city	object
nppes_provider_state	object
specialty_description	object
description_flag	object
drug_name	object
generic_name	object
bene_count	float64
total_claim_count	int64
total_30_day_fill_count	float64
total_day_supply	int64
total_drug_cost	float64
bene_count_ge65	float64
bene_count_ge65_suppress_flag	object
total_claim_count_ge65	float64
ge65_suppress_flag	object
total_30_day_fill_count_ge65	float64
total_day_supply_ge65	float64
total_drug_cost_ge65	float64

2. List of Excluded Individuals and Entities (**LEIE**)
    - Sourced from The Officer of Inspector General, US Department of Health & Human Services[2]
    - The LEIE is administered by the Office of the Inspector General and contains a list of excluded individuals and entities suspected of fraudulent activity.
    - The dataset has 70,227 rows and 18 columns with the following attributes:

LASTNAME	object
FIRSTNAME	object
MIDNAME	object
BUSNAME	object
GENERAL	object
SPECIALTY	object
UPIN	object
NPI	int64
DOB	float64
ADDRESS	object
CITY	object
STATE	object
ZIP	int64
EXCLTYPE	object
EXCLDATE	int64
REINDATE	int64
WAIVERDATE	int64
WVRSTATE	object

3. **Payments** received by physicians
    - Sourced from Centers for Medicare and Medicaid Services Open Payments program[3]
    - Contains payments and other transfers of value made to physicians and teaching hospitals. There are 3 data files for each program year from 2013 to 2017 inclusive.
    - Each of the 3 datafiles is based on the type of payment: General payments, Research payments and Ownership/investment interests.
4. **FDA Drugs**

[1] https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html

[2] https://oig.hhs.gov/exclusions/exclusions_list.asp

[3] https://www.cms.gov/OpenPayments/Explore-the-Data/Dataset-Downloads.html

- Sourced from US Food and Drug Administration (FDA)[4]
- Contains 9 data files with information about prescription and over the counter human drugs and therapeutic biologicals currently approved for sale in the US.
- We focused on the Products file which contains 37,576 rows and 8 columns with the following attributes:

ApplNo	int64
ProductNo	int64
Form	object
Strength	object
ReferenceDrug	int64
DrugName	object
ActiveIngredient	object
ReferenceStandard	float64

- There is also a Marketing Status file which indicates status of drugs, i.e. whether they are prescription, over the counter, discontinued or none of the above.

Data Constraints/Issues

Some of the datasets had issues that prevented a "clean" relationship amongst the tables. The following table outlines the dataset issues identified:

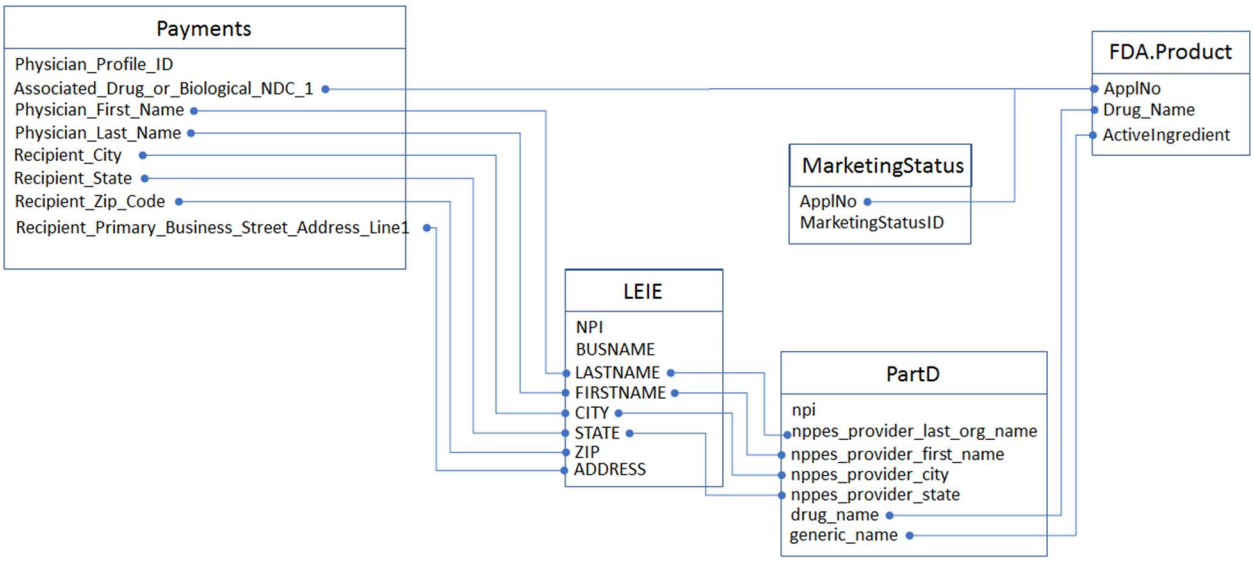
Dataset	Data Constraints/Issues
Part D Prescriber	● Does not contain a zip code attribute ● No explicit last name or entity name exists, the same field is shared ● No entity versus individual flag exists ● Does not contain a flag for excluded individuals or entities
LEIE	● Does contain the National Provider Identifier (NPI) attribute used to uniquely identify an entity or individual but is "0" for approximately 93% of the rows
Payments	● Does not contain an attribute for entity name, only individual fields for physician first and last name attributes exist
FDA Drugs - Products	● No issues

Entity Relationship Diagram

The data constraints and issues were analyzed to determine how the disparate datasets could be related in a meaningful way to allow different modelling techniques.

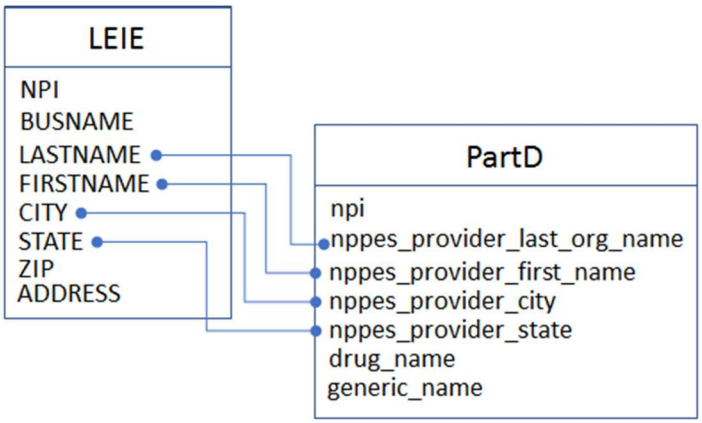
The illustration below depicts how the key relationships are established across the varying datasets:

[4] https://www.fda.gov/Drugs/InformationOnDrugs/ucm135821.htm



LEIE to PartD

This relationship depicts the prescriptions associated with excluded entities and individuals.



Although the datasets did share a unique NPI to match the excluded individuals (physicians) in the LEIE dataset with the prescriber information PartD dataset, the NPI was sparsely populated in the LEIE dataset. As a result, the following logic was considered to match individuals across the 2 datasets:

- First name
- Last name
- City
- State

Note that the actual join was done on only the first name, last name and city variables, because the others were found to be less clean.

As noted in the Data Constraints/Issues section, the PartD dataset does not include a zip code field or an explicit field to capture entity name.

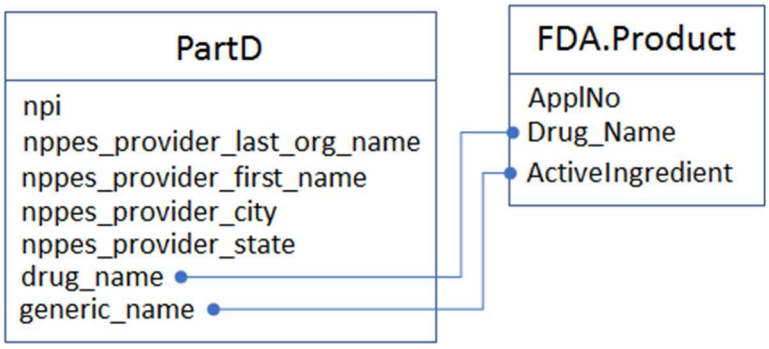
The absence of a zip code means a more coarse level relationship between the datasets with additional effort required to identify and eliminate incorrect matches.

The absence of an explicit field to capture entity name means additional logic is required to identify how the generic last/org name field should be used. We will model the data excluding entity name since we want to focus on sole practitioners, but if we were to incorporate entity name in the future we can apply the following logic:

Scenario	Entity/Individual
If first name is not blank and Last/Org is not blank	Individual
If first name is not blank and Last/Org is blank	Cannot determine, exclude record
If first name is blank and Last/Org is blank	Cannot determine, exclude record
If first name is blank and Last/Org is not blank	Entity

PartD to FDAProduct

This depicts the relationship between the prescription, drug name and active ingredient.

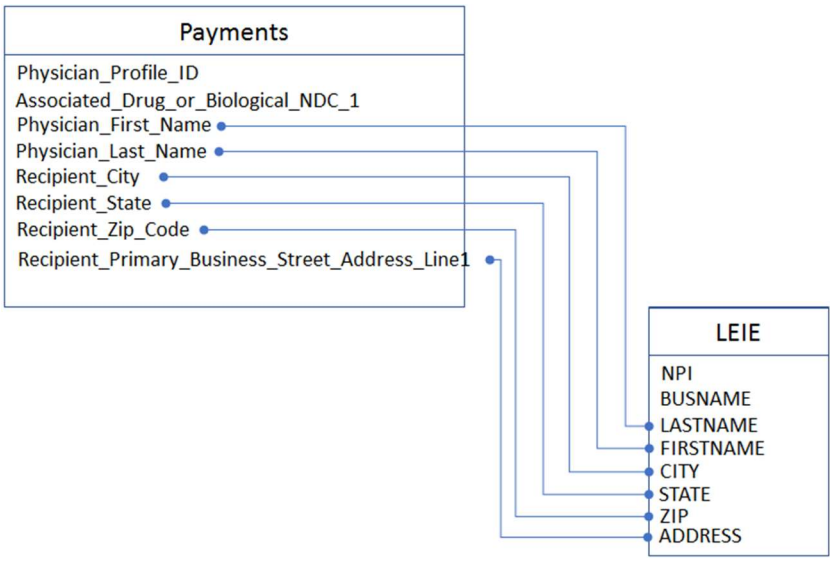


As a result, the following logic could be used to match drug information across the 2 datasets:

- Match on drug name for brand name
- Match on generic name for active ingredient

Payments to LEIE

This relationship depicts the payments associated with excluded entities and individuals.

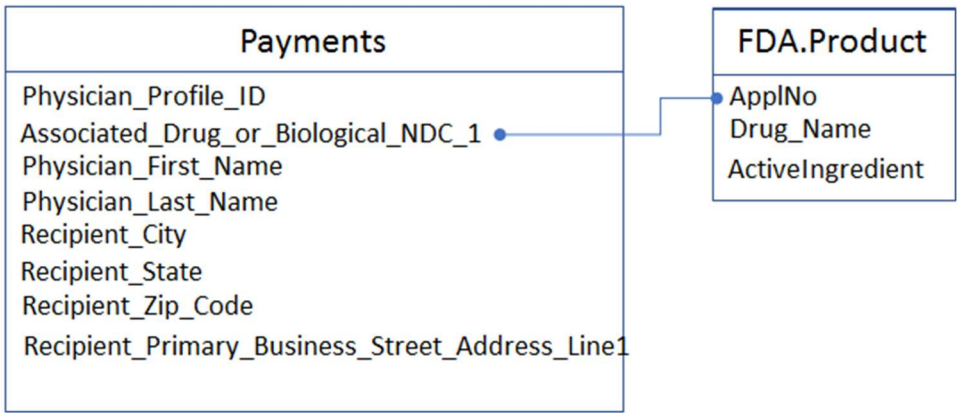


The datasets did not share a single unique identifier to match the excluded individuals (physicians) in the LEIE dataset with the physicians' payment data in the Payments dataset. As a result, the following logic could be used to match individuals across the 2 datasets:

- First name
- Last name
- City
- State
- Zip

Payment to FDAProduct

This depicts the relationship between the prescription payment and additional information about the drug prescribed and the manufacturers linked to a drug.

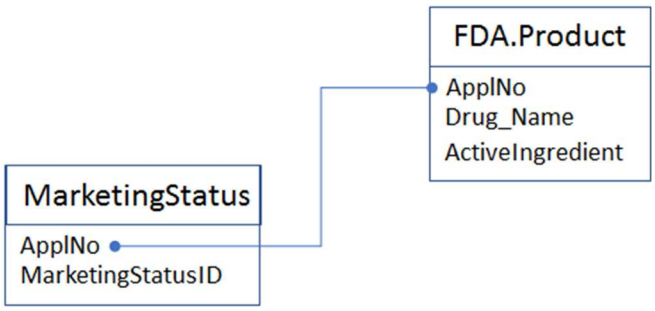


As a result, the following logic could be used to match drug information across the 2 datasets:

- Match on Associated Drug or Biological to Application Number of the manufacturer

Marketing Status to FDAProduct

This depicts the relationship between the prescribed product and an additional FDA table, MarketingStatus.txt, which links "ApplNo" to "MarketingStatusID". Anywhere in MarketingStatus.txt where the MarketingStatusID is 3, the product has been "discontinued" and would raise suspicion over prescriptions being handed out.



As a result, the following logic was used to match drug information across the 2 datasets:

- Match on ApplNo

Data Ingestion

We focused on the PartD, LEIE and FDAProduct data files as they have the core information we want to use for initial modelling.

1. Executed pre-processing on the FDA Drugs file by converting from a txt to csv file to facilitate loading into a Python dataframe.
2. Loaded the Part D Prescriber data file for 1 year into a Python dataframe.
3. Loaded the LEIE file into a Python dataframe.
4. Selected a subset of the LEIE file based on exclusion date from 2013 onwards to focus on recent exclusions that coincide with the time period covered by the PartD dataset. To do this we converted EXCLDATE into date to extract the year, and stored the subset of LEIE into *newleie*. We also dropped all rows where LASTNAME was missing.

```
#focus on LEIE since 2012
leie.year=pd.to_datetime(leie['EXCLDATE'].astype(str), format='%Y%m%d').dt.year
newleie=leie[leie.year>2012]
newleie=newleie.dropna(subset=['LASTNAME'])
```

5. Generated a new dataframe of the FDA Drugs file with unique drug names and dropped the duplicates.

```
#generate unique drug names
prtDdrugs=partD.drug_name.unique()
prtDdrugs.sort()
prtDdrugs

prtDdrugs_g=partD.generic_name.unique()
prtDdrugs_g.sort()
prtDdrugs_g
```

```
fdaDrugs=fda.DrugName.unique()

fda_1=fda[['DrugName','ActiveIngredient']]
fda_2=fda_1.drop_duplicates(keep=False)
```

6. We use Part D file as our base file since it has all the prescription information. We will join all other relevant datasets to this file with a "left join".

- Merged the Part D file with the FDA drug file using DrugName as the key on which to join and created a new merged file called *partD_drug*.
- Brought in only DrugName and ActiveIngredient from the FDA drug file since these are the attributes of interest that we want to use as predictors of fraud.

```
#Merge prescription and active ingredient
partD_drug = pd.merge(partD, fda_2[['DrugName', 'ActiveIngredient']], how='left', left_on='drug_name', right_on='DrugName')
```

Excluded Practitioners

7. Created a unique key on which to join the LEIE and Part D prescriber datasets and then merged the *partD_drug* dataset with *newleie* to create a newly merged dataset including data from Part D, LEIE and FDA drugs all in one dataframe.

```
#make merge keys for prescrptions and banned subscriber list
partD_drug["key"]=partD_drug["nppes_provider_last_org_name"]+partD_drug["nppes_provider_first_name"]+partD_drug["nppes_provider_city"]
newleie["key"]=newleie["LASTNAME"]+ newleie["FIRSTNAME"]+ newleie["CITY"]


#merge
merge_partD=pd.merge(partD_drug, newleie, on='key', how='left')
```

8. Repeated the above steps for each yearly Part D prescriber data file. The result is a dataset for each year of all drug data associated with individuals on the excluded list (individuals who have committed healthcare fraud in the past), called *PartD_Prescriber_PUF_NPI_Drug_*year*_badpeoplemerge*.

Non-Excluded Practitioners

9. Similarly, we run a loop to create a non-excluded practitioner file. In order to have a balanced dataset for training the model, we include the same number of good practitioners as bad practitioners (based on determination above) for a given year, with all prescriptions from each in the respective *goodpeoplemerge* and *badpeoplemerge* files for each year.

```
#load data
partDBad=pd.read_csv(fname3, sep=',')
partDall=pd.read_csv(fname2, sep=',')
#how many unique bad prescribers
totaluniqueBad=partDBad.npi.nunique()
#find good prescribers
good=partDall[partDall["EXCLDATE"].isnull()]
uniqueGood=good.npi.unique()
totaluniqueGood=good.npi.nunique()
#create an index of good subscribers the same lenght as the bad ones
random_index = np.random.choice(uniqueGood, totaluniqueBad)
#find all prescriptions from good subscribers
partDGood=partDall[partDall.npi.isin(random_index)]
partDGood.to_csv(fname4)
```

10. We combine the *goodpeoplemerge* and *badpeoplemerge* files from each year and add a column representing whether they are “good” or not (1, 0). That is, not being on the LEIE is “good”, being on the LEIE is “bad”; this is our training variable for the model. We also add a year column to identify from which year the data relates.

```
#load data
partDBad=pd.read_csv(fname3, sep=',')
partDGood=pd.read_csv(fname4, sep=',')

partDBad["good"]=0
partDGood["good"]=1

partDBad["infoYear"]=x
partDGood["infoYear"]=x

frames = [partDBad,partDGood]

result = pd.concat(frames)

#result.groupby(['key', 'generic_name'])['total_claim_count'].agg('sum')
frames = [superframe,result]

superframe=pd.concat(frames)

superframe.to_csv('superframe.csv')
```

11. The resulting combined datafile, called *superframe.csv,* covers 2013 to 2016 and has 108769 rows x 47 columns. It is read into a Python dataframe called *superframe* for further exploration and analysis.

```
superframe.shape
```

```
(108769, 47)
```

12. The number of instances associated with "good" practitioners is 58726 (53%). The number associated with "bad" practitioners is 50043 (46%).

```
#Target variable
freq_table = pd.Series(superframe['good']).value_counts()
print(freq_table)
```

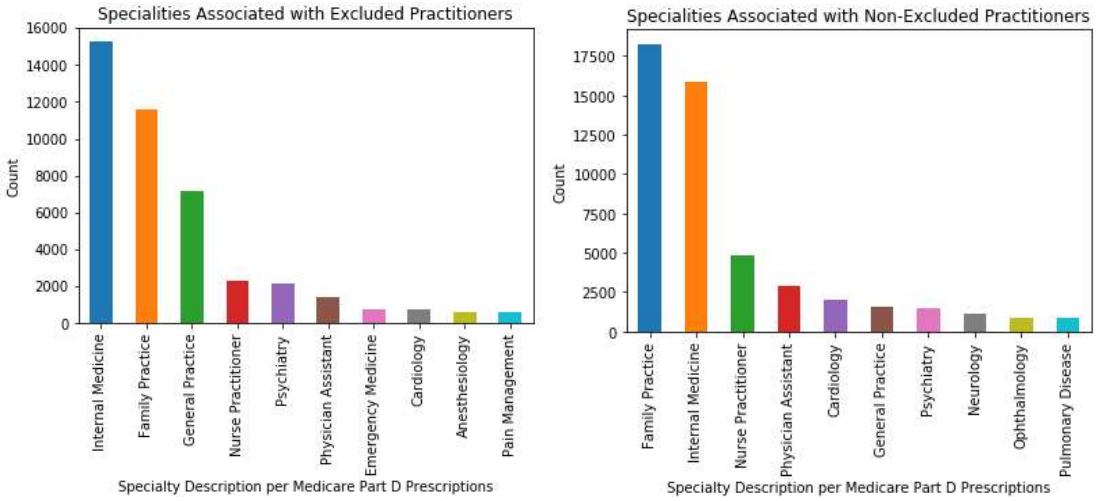
```
1    58726
0    50043
Name: good, dtype: int64
```

```
1    0.539915
0    0.460085
Name: good, dtype: float64
```

Data Exploration and Preparation

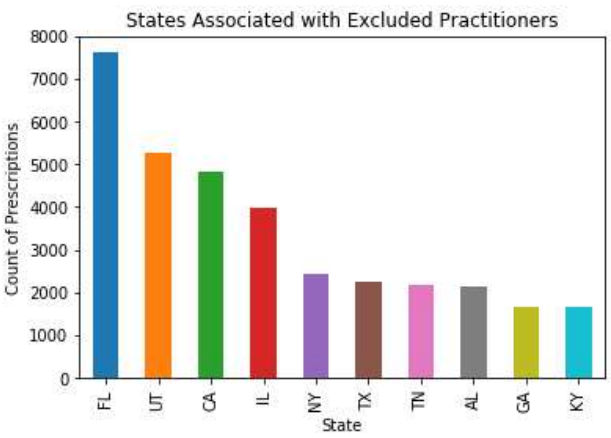
Categorical Variables

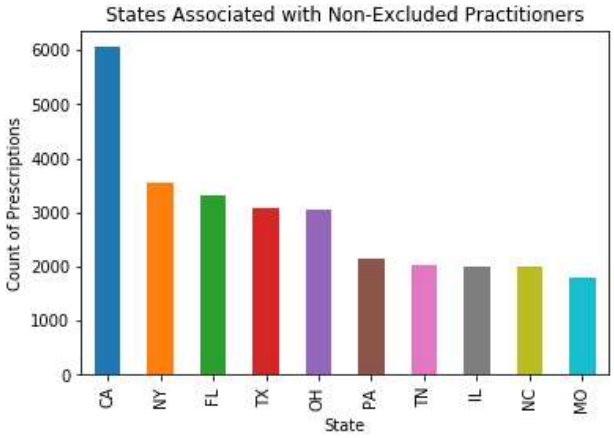
1. Explore counts of specialty areas associated with the "good" and "bad" practitioners.



- Internal Medicine is the most popular specialty amongst prescriptions written by excluded practitioners and the second most popular amongst non-excluded practitioners. Internal medicine is a broad-based specialty dedicated to treating adults. Because it's a very general category, there may be more opportunity for fraudulent practices to go undetected.

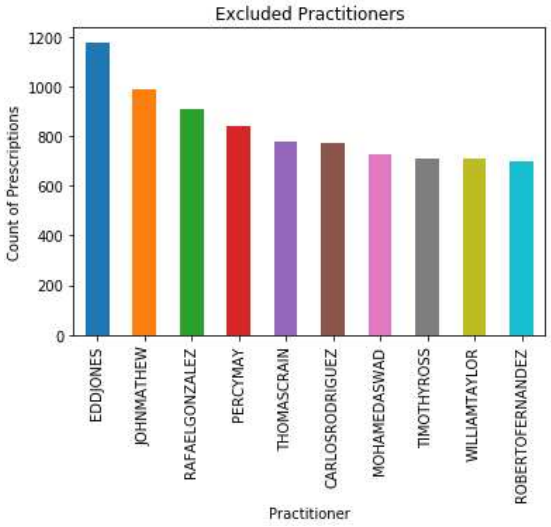
2. Explored the US states where "good" and "bad" practitioners are located.

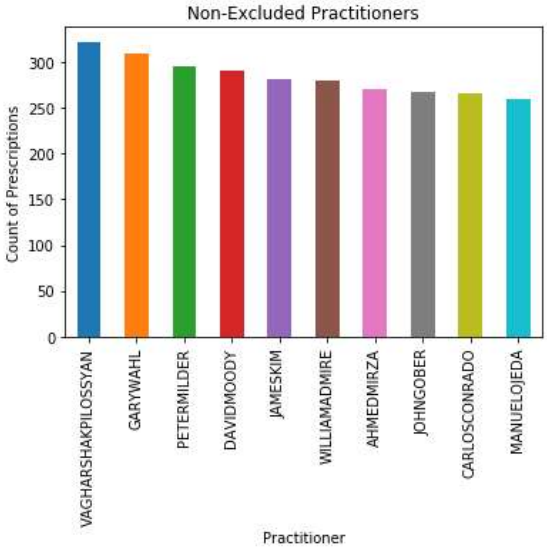




- Florida is the most popular state amongst prescriptions written by excluded practitioners. Perhaps this is because there are a lot of seniors in Florida as it is considered a winter home for many.
- California is the most popular state amongst prescriptions written by non-excluded practitioners
- Florida, California and New York are in the top 5 for both views

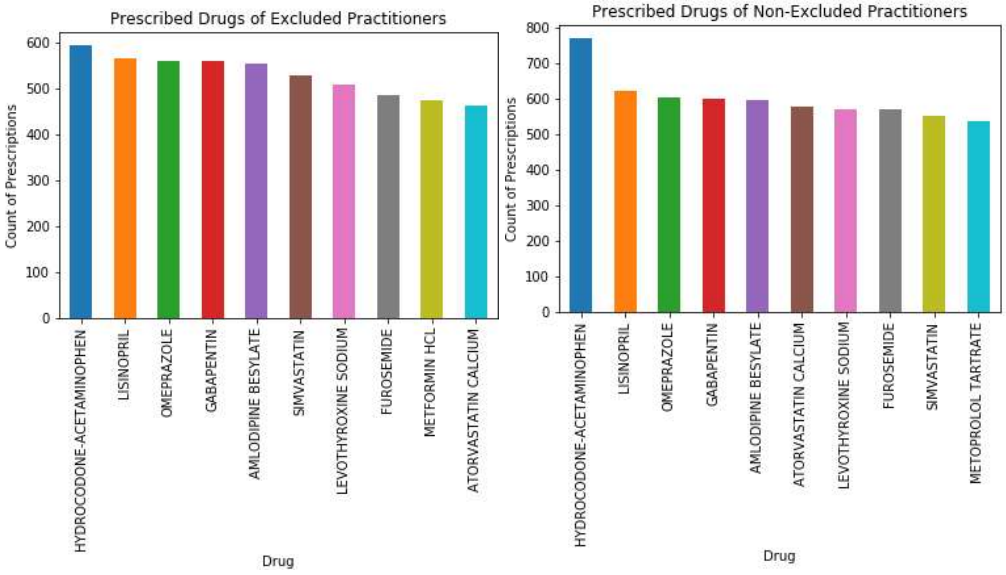
3. Concatenated first and last name and visualized the frequency by practitioner to get an idea of the magnitude of transactions that practitioners who have committed fraud have done compared to those who haven't committed fraud.





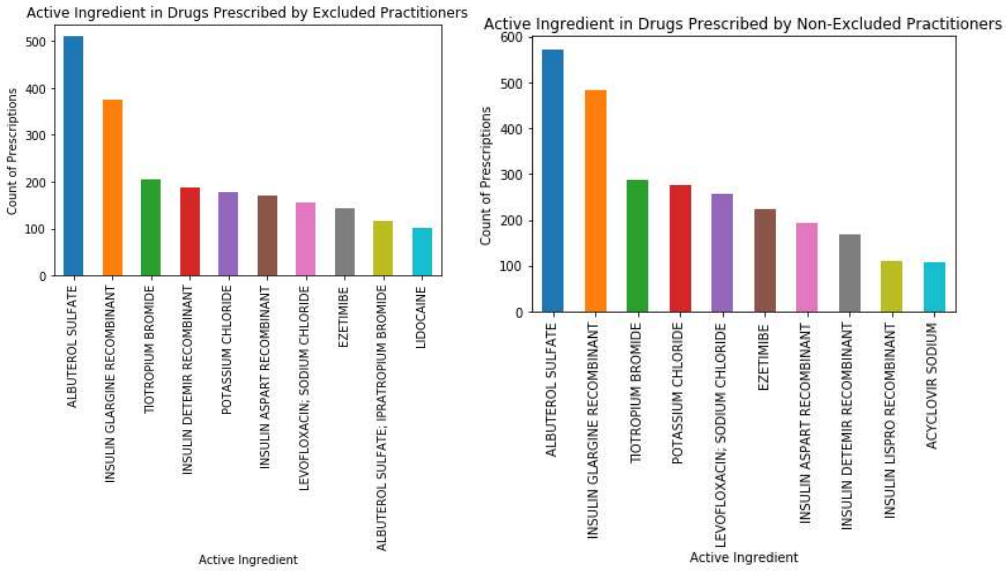
- EDD Jones wrote approximately 20% more prescriptions than the second ranked excluded practitioner
- The 7th to 10th top ranked excluded practitioners wrote approximately the same amount of prescriptions
- Nominal variance of prescriptions written by the top and 10th ranked non-excluded practitioners

4. Frequency by drug name



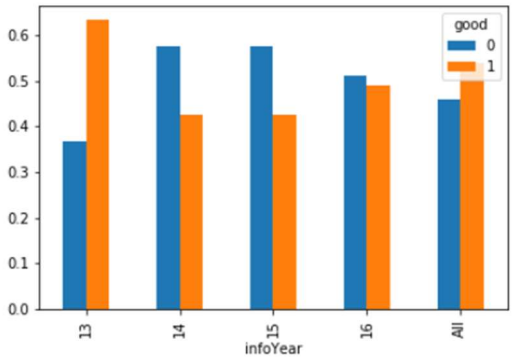
- The top 10 drugs prescribed by excluded practitioners have similar frequencies
- 7 of the top 10 drugs are the same for both excluded and non-excluded practitioners indicating perhaps there is no pattern between the prescribed drug and incidence of fraud.

5. Frequency by active ingredient



- Excluded and non-excluded practitioners prescribe drugs with 2 active ingredients more frequently than the rest
- The top 3 drugs that excluded and non-excluded practitioners prescribe have the same top 3 active ingredients indicating not much differentiation is evident between the two groups.

6. Percentage of drug transactions associated with “good” and “bad” practitioners over the time period of coverage shows a greater proportion that are associated with “good” practitioners in 2013 but all other years have a greater proportion related to “bad” practitioners.



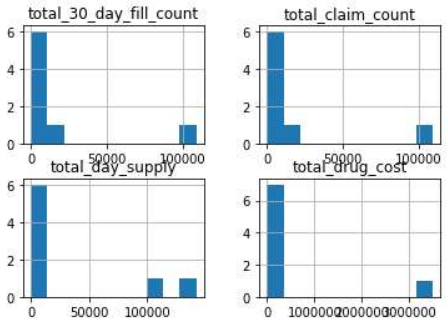
Numeric Variables

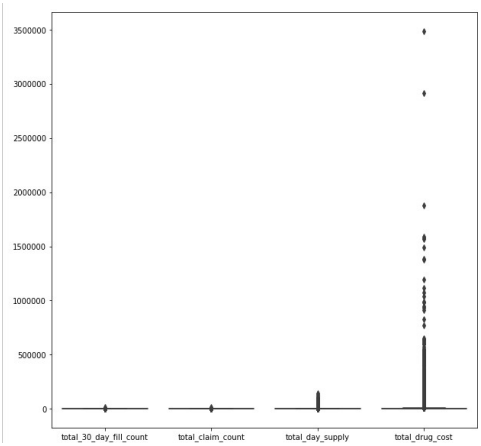
7. Explore the statistics behind selected numeric attributes:

	total_30_day_fill_count	total_claim_count	total_day_supply	total_drug_cost
Min.	11.00	11.00	9.00	0.00
1st Qu.	18.00	15.00	443.00	273.00
Median	32.00	25.00	857.00	740.00
Mean	72.81	55.69	2026.00	4389.00
3rd Qu.	69.00	52.00	1920.00	2629.00
Max.	13545.00	13499.00	142293.00	3487406.00

- The monthly prescriptions *(total_30_day_fill_count)* has a slightly larger Median/Mean than Medicare claims by the patients *(total_claim_count)*. This is reasonable as coverage varies but shouldn’t exceed monthly prescriptions.
- Daily supply (*total_day_supply)* is pills taken during a 30-day period. The data appears to have massive outliers which could indicate suspicious cases.

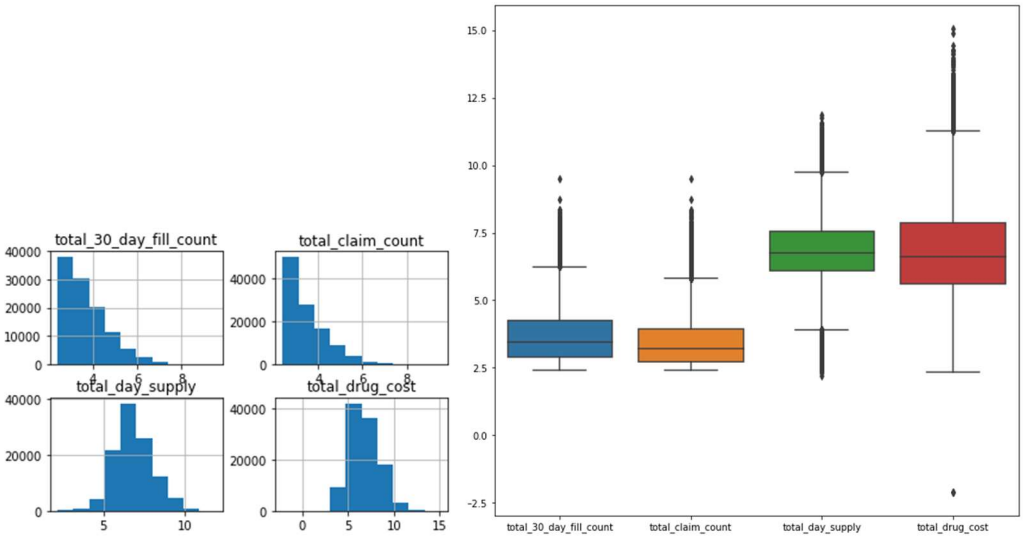
8. Histograms and box plots of the selected numeric attributes from *superframe* confirm existence of outliers.





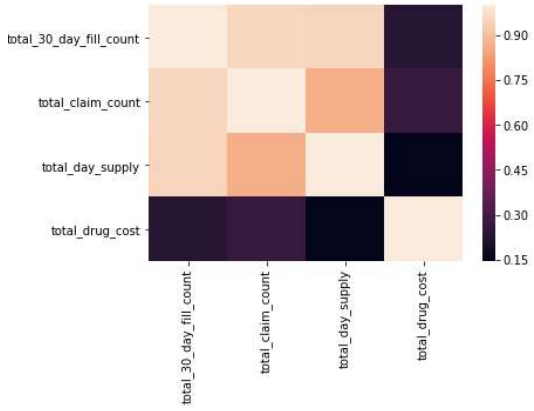
9. Perform a log transformation to normalize the distributions to enable a better view of the plots since it is difficult to gain insight from the above due to the presence of extreme outliers.

num_cols =
np.log(num_cols[['total_30_day_fill_count','total_claim_count','total_day_supply','total_drug_cost']].replace(0, np.nan))

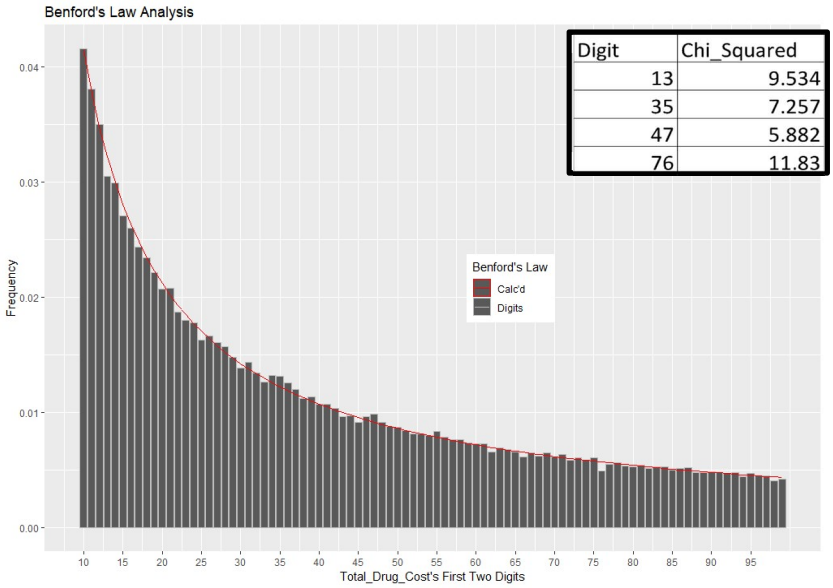


10. We will conclude on whether outlier treatment (if any) is needed prior to modelling.

11. Correlation plot of the selected numeric attributes shows
    - the total_30_day_fill_count is highly positively correlated with total_day_supply and total_claim_count
    - there is a small positive correlation between total_30_day_fill_count and total_drug_cost



12. Benford's Law: In fraud detection, unusual activity can sometimes be spotted by financial records' first digits NOT following the calculated log distribution – which is a mathematical probability rule summing the frequency of 1 occurring in $10, $15, $11, $23 etc. We explore the drug cost attribute in our *superframe* dataset by applying Benford's Law and plotting the results:



The plot shows that anomalies exist at very high chi values for drug cost amounts whose first two digits are 13, 35, 47, and 76 which means that Benford's Law can assist in finding anomalies.

13. A total of 13 columns in *superframe* were removed:
    - The attribute *bene_count* was not included as the column was not consistently populated
    - The following 7 attributes were not included as the columns are a subset of the entire data and only specific to patients 65 and older:
        - bene_count_ge65
        - bene_count_ge65_suppress_flag
        - total_claim_count_ge65
        - ge65_suppress_flag
        - total_30_day_fill_count_ge65
        - total_day_supply_ge65
        - total_drug_cost_ge65
    - Duplicated columns were removed (arising from the LEIE and PartD merge):
        - NPI
        - LASTNAME
        - FIRSTNAME
        - CITY
        - STATE

14. Remaining columns were aggregated to produce a final dataset for modelling.
    - One potential way a fraudster could try to mask their actions is by prescribing many types of opioids in smaller quantities. To unmask these cases, prescriptions were aggregated for each provider according to the generic name of the prescribed drug. This was done using a *groupby* statement. Given that several drugs were collapsed into a single generic name, it was necessary to *sum* dispensing-related info for each of them. Also, basic practitioner-level data had to be repopulated in the dataframe (i.e., name, state, etc). This was done by

copying the *first* instance of this data for every entry of the practitioner, as specified in the following dictionary file.

#column aggregation types

```
colAgg={'ADDRESS':'first',
        'ActiveIngredient':'first',
        'BUSNAME':'first',
        'DOB':'first',
        'DrugName':'first',
        'EXCLDATE':'first',
        'EXCLTYPE':'first',
        'GENERAL':'first',
        'REINDATE':'first',
        'WAIVERDATE':'first',
        'WVRSTATE':'first',
        'ZIP':'first',
        'description_flag':'first',
        'drug_name':'first',
        'generic_name':'first',
        'good':'first',
        'infoYear':'first',
        'key':'first',
        'npi':'first',
        'nppes_provider_city':'first',
        'nppes_provider_first_name':'first',
        'nppes_provider_last_org_name':'first',
        'nppes_provider_state':'first',
        'specialty_description':'first',
        'total_30_day_fill_count':'sum',
        'total_claim_count':'sum',
        'total_day_supply':'sum',
        'total_drug_cost':'sum',
        }
```

- The aggregation was then completed, grouping by practitioner (npi) and generic name, referencing the dictionary to determine how to treat other columns.

```
df_new = superframe.groupby(['npi', 'generic_name']).aggregate(colAgg)#.reindex(columns=superframe.columns)
```

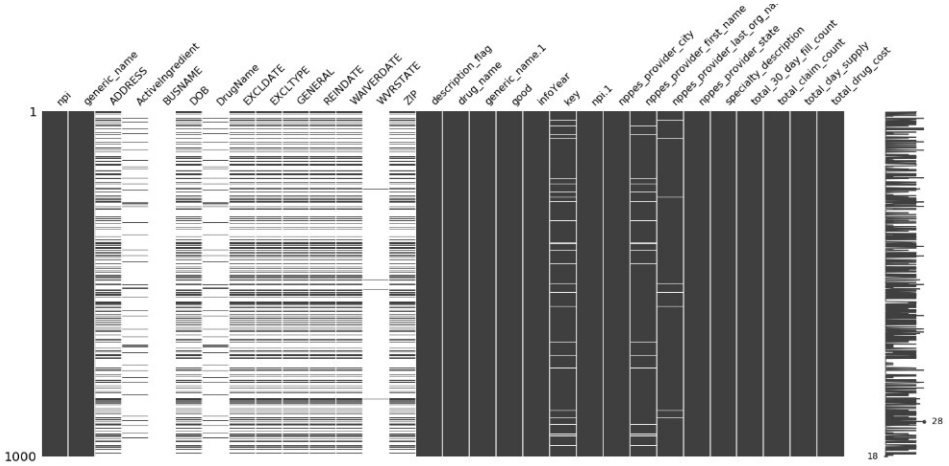
This new dataframe now has 78231 rows and 30 columns.

Missing Value Treatment

15. The missing values calculation and visualization of a sample of 1000 instances indicates several missing values. For example, the features brought over from the LEIE data file have missing values together which makes sense given these values would only exist for fraudulent prescriptions and therefore would be missing for those instances that are not fraudulent.

```
df_new.isnull().sum()
```

npi	0
generic_name	0
ADDRESS	54526
ActiveIngredient	72404
BUSNAME	78231
DOB	54526
DrugName	72404
EXCLDATE	54526
EXCLTYPE	54526
GENERAL	54526
REINDATE	54526
WAIVERDATE	54526
WVRSTATE	77964
ZIP	54526
description_flag	0
drug_name	0
generic_name.1	0
good	0
infoYear	0
key	3011
npi.1	0
nppes_provider_city	3
nppes_provider_first_name	2305
nppes_provider_last_org_name	703
nppes_provider_state	0
specialty_description	0
total_30_day_fill_count	0
total_claim_count	0
total_day_supply	0
total_drug_cost	0
dtype: int64	



We treat the missing values iteratively:

a. Attribute “key” was used to join the files, so if there are missing values in “key” that means we could not join the data, so we want to remove those rows. Also, the missing “key” values represents 3011/78231 = 3.8% of the data which is a small enough percentage that removing them will not reduce the dataset significantly. The number of instances reduced from 78231 to 75220.
b. Remove DrugName since it’s a duplicate of drug_name and has missing values (whereas drug_name does not).
c. Attribute npi.1 is a duplicate of npi (arising from the aggregation) so remove npi.1
d. Attribute generic_name.1 is a duplicate of generic_name (arising from the aggregation) so remove generic_name.1
e. REINDATE has either nulls or zeros in 100% of the instances, so remove REINDATE. (54526+20694)/75220=100%

```
df_new1['REINDATE'].isnull().sum()
```

```
54526
```

```
(df_new1['REINDATE'] == 0).sum(axis=0)
```

```
20694
```

f. WAIVERDATE has either nulls or zeros for almost all the instances, so remove WAIVERDATE. (54526+20427)/75220 =99.7%

```
df_new1['WAIVERDATE'].isnull().sum()
```

```
54526
```

```
(df_new1['WAIVERDATE'] == 0).sum(axis=0)
```

```
20427
```

g. WVRSTATE and BUSNAME are very sparse. The entire column of BUSNAME is missing so we should remove it. WVRSTATE has 99.6% missing values, so we should remove it. Both are not needed for modelling and they came from the LEIE data meaning they are directly related to our target variable and would make our model look accurate but would not perform accurately in the real world.
h. Remove ADDRESS since we have the zip code attribute and both have the same number of missing values.
i. Sole purpose of the attribute “key” was to join the files so we can drop this column from the dataset since we have no further use for it.
j. Remove ‘description_flag’ as this indicates the source of the specialty description which is not relevant for our purpose.
k. Remove additional variables directly related to the LEIE dataset since keeping them will result in data leakage when predicting our target variable of “good” or not. These include DOB, EXCLDATE, EXCLTYPE, GENERAL and ZIP.
l. Remove 'nppes_provider_first_name', and 'nppes_provider_last_org_name' which would discriminate our model and identify fraudsters by unique name. We want to classify fraudsters based on their characteristics and not by name.

```
df_new2=df_new1.drop(['DrugName','npi.1','generic_name.1','REINDATE','WAIVERDATE','WVRSTATE', 'BUSNAME',
                      'ADDRESS','key','DOB','EXCLDATE','EXCLTYPE','GENERAL','ZIP','description_flag',
                      'nppes_provider_first_name','nppes_provider_last_org_name'],axis='columns')
```

16. After dropping the above attributes, check what remains missing in df_new2:

```
df_new2.isnull().sum()
```

```
npi                          0
generic_name                 0
ActiveIngredient         69576
drug_name                    0
good                         0
infoYear                     0
nppes_provider_city          0
nppes_provider_state         0
specialty_description        0
total_30_day_fill_count      0
total_claim_count            0
total_day_supply             0
total_drug_cost              0
dtype: int64
```

17. We will keep ActiveIngredient's missing values for now as they will be treated via label encoding.

18. The dataframe, after treating the above missing values, has 75220 rows and 13 columns. The proportion of good vs bad transactions is 72% and 28%. The proportion is important since we want to ensure we have enough "bad" transactions on which to train the model, since our objective is to identify the fraudulent prescriptions. Currently our dataset is imbalanced, so we will balance it later before modelling.

```
df_new2.shape
```

```
(75220, 13)
```

```
1    54526
0    20694
Name: good, dtype: int64
```

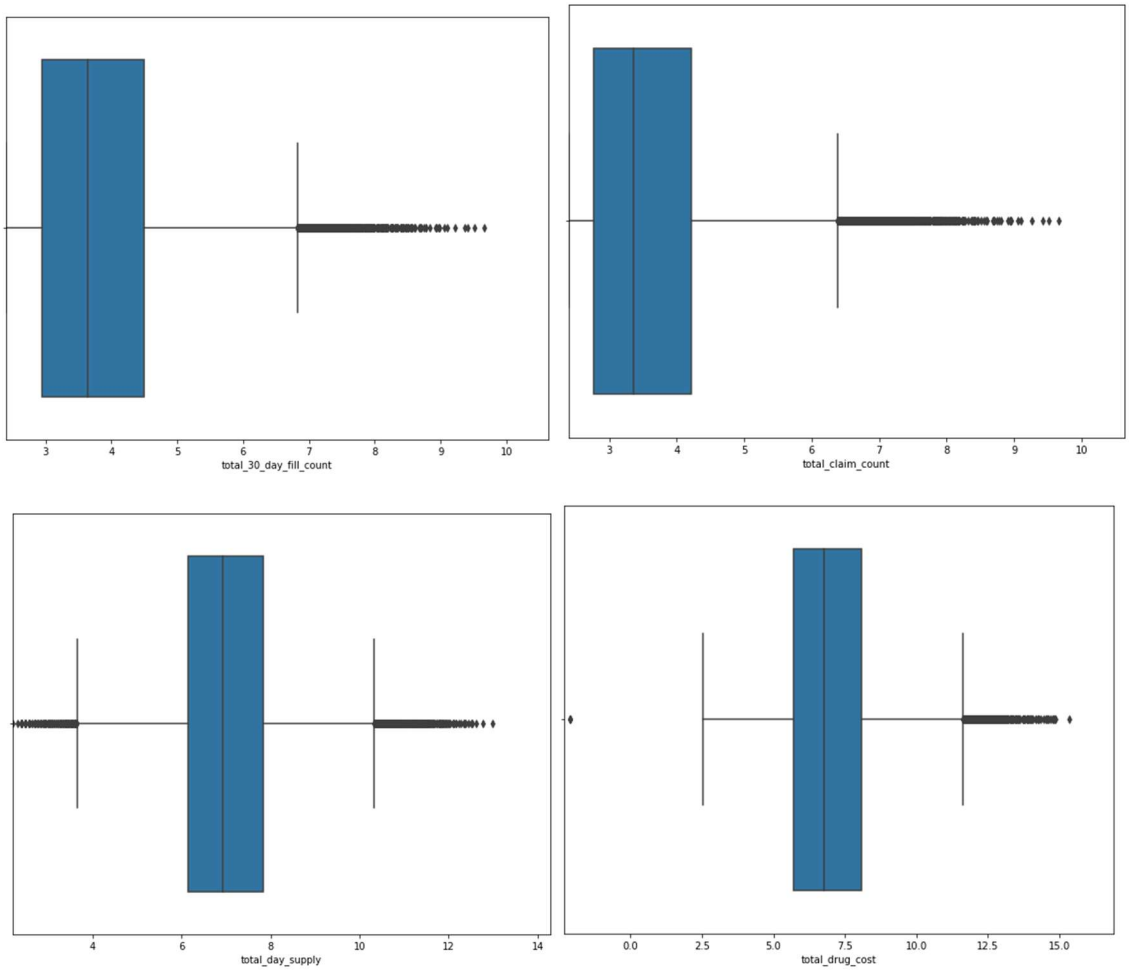
```
1    0.724887
0    0.275113
Name: good, dtype: float64
```

Outlier Detection

19. Stats on df_new2 shows outliers in the selected attributes:

	total_30_day_fill_count	total_claim_count	total_day_supply	total_drug_cost
Min	11.00	11.00	9.00	0.00
1st Qu	19.00	15.00	450.00	287.00
Median	37.00	28.00	990.00	846.00
Mean	101.20	77.43	2816.00	6102.00
3rd Qu	88.10	66.00	2460.00	3136.00
Max	15757.60	15745.00	434728.00	4664741.00

To enable a better view of the box plots, we perform log transformation to get the following:



Since the purpose of our analytical tool is to detect anomalies (i.e. fraudulent prescriptions), outliers should not be removed from the data, as they could represent the anomalies that we're looking for. We will discuss log transformations further in the Modelling section.

Encoding Categorical Variables

To make our target variable more intuitive we attach a fraud label where those transactions previously labelled 0 meaning 'not good' is labelled 1 for 'fraud' and 0 for 'not fraud' and we drop the 'good' column. This change will make it easier to interpret the results of our classification models.

```
#change the target variable so that fraud = 1 and non-fraud = 0
df_new2['fraud']=(df_new2['good']-1)*-1
```

Decision tree and logistic regression algorithms require categorical variables to be converted into numeric values before the algorithms can process them. Our latest dataframe has many variables of "object" data type which must be encoded or factorized into numeric values.

```
npi                        int64
generic_name              object
ActiveIngredient          object
drug_name                 object
infoYear                   int64
nppes_provider_city       object
nppes_provider_state      object
specialty_description     object
total_30_day_fill_count  float64
total_claim_count          int64
total_day_supply           int64
total_drug_cost          float64
fraud                      int64
dtype: object
```

We make a copy of the df_new2 dataframe for encoding and call it encoded_df. We identify the categorical columns and perform factorization on them to assign numeric values to each unique value within.

```
cat_cols = encoded_df.select_dtypes(['object'])
cat_cols.columns
```

```
Index(['generic_name', 'ActiveIngredient', 'drug_name', 'nppes_provider_city',
       'nppes_provider_state', 'specialty_description'],
      dtype='object')
```

```
#factorize the categorical vars
label_mapping = {}

for c in cat_cols:
    encoded_df[c], label_mapping[c] = pd.factorize(encoded_df[c])
```

We balance the dataset by duplicating the number of fraudulent transactions

```
#make balanced dataset
freq_fraud1=encoded_df[encoded_df.fraud==1]
freq_fraud0=encoded_df[encoded_df.fraud==0]
freq_fraud0=freq_fraud0.iloc[:41388,]
freq_fraud1=pd.concat([freq_fraud1,freq_fraud1])
encoded_df=pd.concat([freq_fraud0,freq_fraud1])
```

An excerpt of the dataframe shows text values have been replaced by numeric values as a result of the encoding, and the number of rows has increased.

```
encoded_df.head()
```

	npi	generic_name	ActiveIngredient	drug_name	infoYear	nppes_provider_city	nppes_provider_state	specialty_description	total_30_day_fill_count
0	1003001868	0	-1	0	13	0	0	0	34.0
6	1003076159	1	-1	1	13	1	1	1	12.0
7	1003076159	2	-1	2	13	1	1	1	18.0
8	1003076159	3	-1	3	13	1	1	1	18.0
9	1003076159	4	-1	4	13	1	1	1	12.4

The dimensions of encoded_df now has 82,776 columns with 13 rows. Recall our original 'fraud' population was 20,694. If doubled we get 41,388 and we want equal number of non-fraudulent transactions so our new population is 41,388*2=82,776.

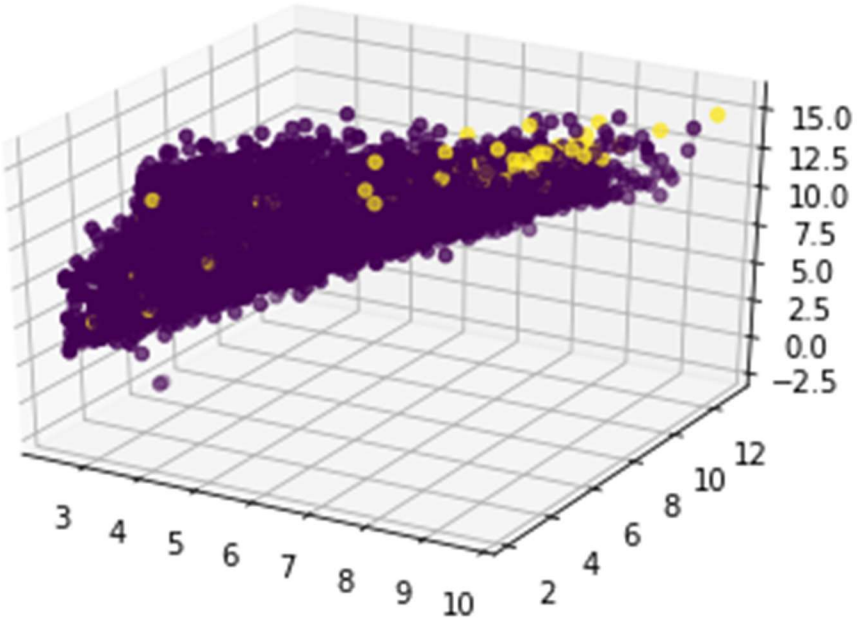
We use encoded_df for modelling.

Modelling

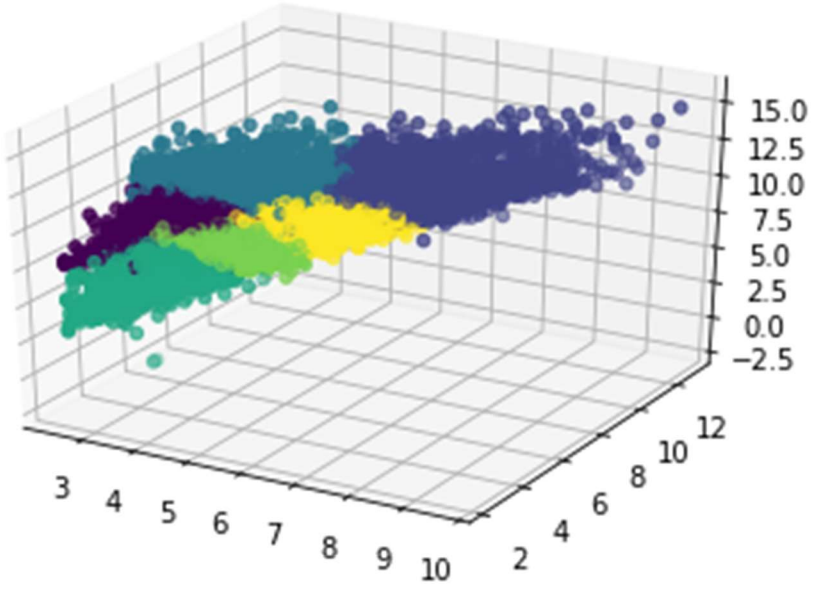
Clustering Model

The concept behind K-means is that we tell the algorithm we want the data to be divided into a specific number of groups that we define. The goal of K-means clustering is to divide the data in such a way that the sum of squares from data points to the assigned cluster centres is minimized.

We used K-means clustering to see if opioid prescriptions could be identified in a data-driven manner. Opioids were identified by creating a flag column in the data table that searched for generic and drug name values that matched those listed on an FDA risk evaluation list focusing on opioids. The upper plot displays prescription claims data on 3 axes: total claim count, total day supply and total drug cost. Opioid drugs are colored yellow, and non-opioid drugs are colored purple. This figure reveals that opioids are high on all three variables of interest as compared to other drugs.



Comparison of this labelled image to an image color coded using K-means clustering (below) reveals that one of the clusters has reasonably good overlap with the opioid drugs (indicated in blue). This suggests that opioids can be distinguished from non-opioid drugs in a data-driven manner, providing some evidence that prescribing patterns of these drugs are unusual and could imply fraud.



Classification Models

Logistic Regression

Logistic regression is used to predict a binary, categorical variable outcome (fraud or not) given a set of independent variables.

The Y variable (fraud/not fraud) indicates the class relative to the observation – this is the target value to be predicted.

We create an input dataframe by dropping the target variable 'fraud' from encoded_df as well as 'npi' which is a unique identifier for the practitioner that is not necessary for modelling. We also create a target variable 'fraud' to feed the model.

```
inputvar=encoded_df.drop(['fraud','npi'], axis=1) targetvar=encoded_df['fraud']
```

We split the data into 70% train and 30% test and apply cross validation which helps to ensure the best fitting of the model by maximizing the use of the available data for training and testing.

```
X_train, X_test, y_train, y_test = cross_validation.train_test_split(inputvar, targetvar, test_size=0.3)
```

We run logistic regression on the training data and predict using the test data:

```
LRmodel = LogisticRegression()
LRmodel.fit(X_train,y_train) #use same train-test split as in decision tree
```

```
LRpred = LRmodel.predict(X_test)
```

We get the following results from logistic regression:

```
print(accuracy_score(y_test,LRpred))
```

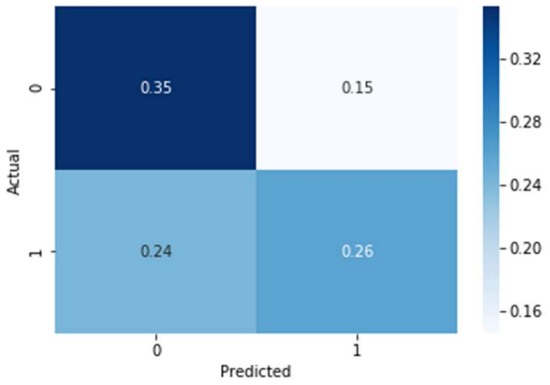
```
0.6129746708009504
```

```
print(confusion_matrix(y_test,LRpred))
```

```
[[8771 3615]
 [5996 6451]]
```

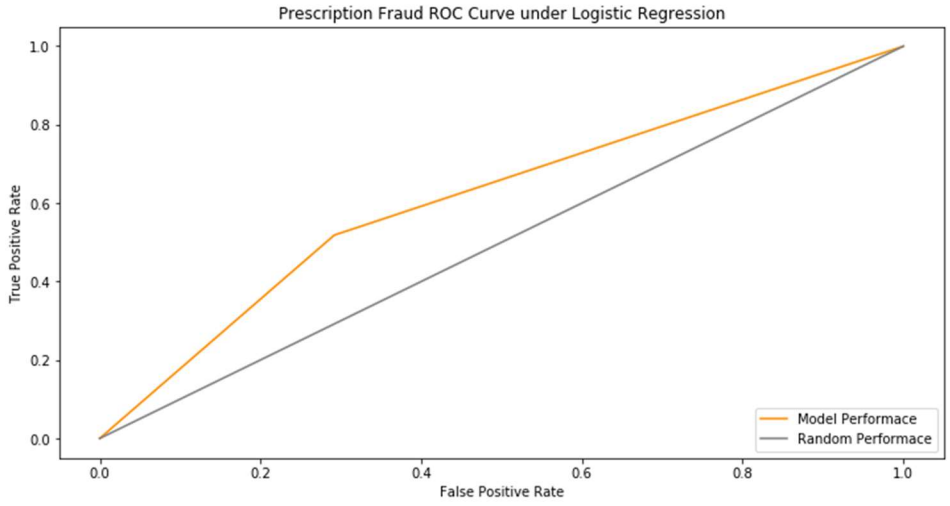
```
print(classification_report(y_test,LRpred))
```

	precision	recall	f1-score	support
0	0.59	0.71	0.65	12386
1	0.64	0.52	0.57	12447
avg / total	0.62	0.61	0.61	24833



Accuracy is 61% and precision and recall on fraud are 64% and 52% respectively. 24% false negative rate means 24% of fraudulent transactions would go undetected with this model.

We plot the ROC curve, a common graph to visualize the performance of a binary classifier, by plotting the True Positive Rate (y-axis) against the False Positive Rate (x-axis). We also compute the AUC which is commonly used to summarize a classifier's performance in a single value. With logistic regression, ROC and AUC are not very good as performance is slightly better than random.



As discussed previously, the numeric variables have outliers. We try the logistic regression model with log-transformed numeric attributes to see if model performance improves.

```
#Log transform the numeric variables
inputvar2[['total_30_day_fill_count','total_claim_count','total_day_supply',
           'total_drug_cost']]=np.log(inputvar2[['total_30_day_fill_count','total_claim_count',
                                                 'total_day_supply','total_drug_cost']])
```

We use inputvar2 as the input variables for the model. Target variable is the same.

```
X_train2, X_test2, y_train2, y_test2 = cross_validation.train_test_split(inputvar2, targetvar, test_size=0.3)
```

```
LRmodel2 = LogisticRegression()
LRmodel2.fit(X_train2,y_train2)
```

Accuracy increases to 65% (from 61% without log transformations) so log transforming the numeric attributes does not make a significant difference.

```
print(accuracy_score(y_test2,LRpred2))
```

```
0.6503442999234889
```

Using the "drugstandards" library, generic names and brand name drugs can be grouped by a single chemical name further reducing the parameters. Similarly, a "brand" column can be made by assigning "True" if Drug_name is NOT equal to generic_name. Replacing drug_name and generic_name with these two columns further yields an accuracy value of 0.68.

Decision Tree

The Decision Tree algorithm works by making splits that best separates (discriminates) the data for the classes or predictions being made (fraud or not) by learning simple decision rules inferred from the inputs. This type of model is suitable for our analytical problem since we want to discriminate between "fraud" vs "not fraud" transactions to enable prediction of fraud on unseen data.

Due to the number of features and number of unique values in some of the features, we define a maximum depth to our decision tree of 5 to prevent overfitting of the training data. On our test data, our model achieved 72% accuracy (9449+8512)/24833.

```
print(accuracy_score(y_test,dtree_prediction))
```

```
0.7232714533080981
```

The confusion matrix indicates that out of 24833 predictions (test data):

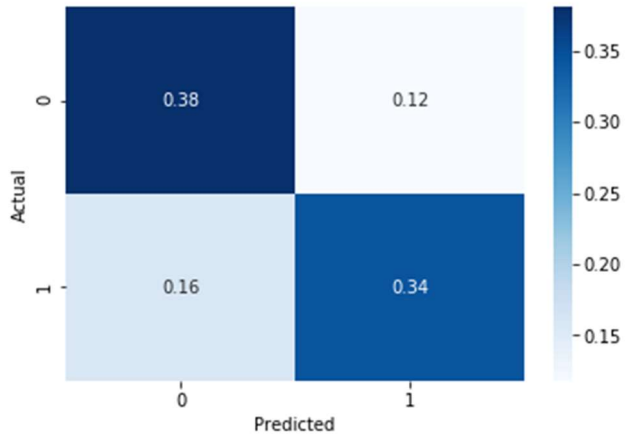
- 2937+8512=11449 transactions were predicted as fraud and 9449+3935=13384 transactions were predicted as not fraud
- in reality, 3935+8512=12447 were actually fraud and 9449+2937=12386 were actually not fraud
- 34% of the test set was correctly predicted to be fraud while 38% was correctly predicted not to be fraud (somewhat equal proportions given our dataset was evenly balanced).
- 12% transactions were predicted as fraud when they actually were not (false positive) and 16% transactions were predicted as not fraud when they actually were (false negative)
- The precision on fraud is 8512/(8512+2937)=74% while recall on fraud is 8512/(8512+3935)=68%

```
print(confusion_matrix(y_test,dtree_prediction))
```

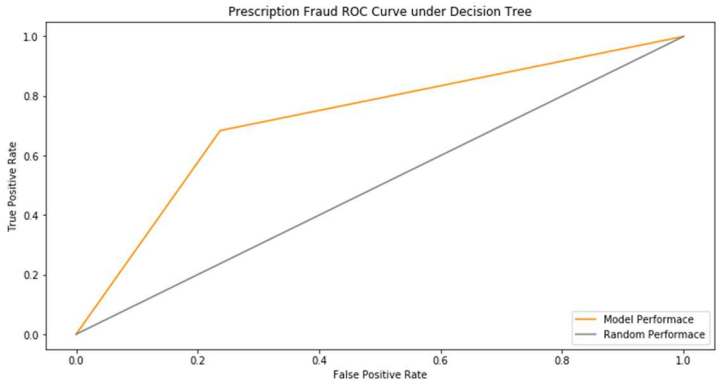
```
[[9449 2937]
 [3935 8512]]
```

```
print(classification_report(y_test,dtree_prediction))
```

	precision	recall	f1-score	support
0	0.71	0.76	0.73	12386
1	0.74	0.68	0.71	12447
avg / total	0.72	0.72	0.72	24833



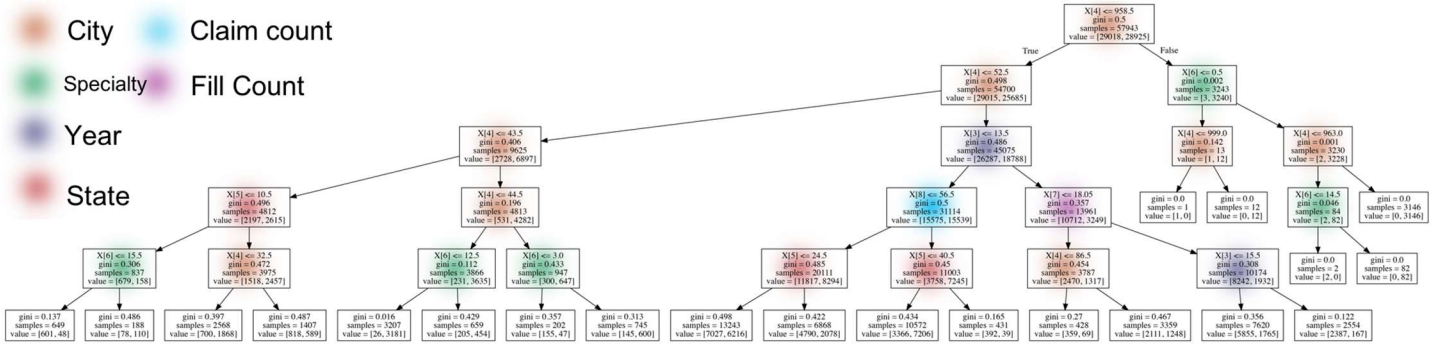
We can see that the ROC and AUC indicates our model performs significantly better than random. We also computed the ROC and AUC score on train data and get a similar score confirming that our model has not been overfitted.




```
roc_auc_score(y_test,dtree_prediction)
```

```
0.7233685034136212
```

Visual representation of the decision tree:



An examination of the decision tree reveals city as an important factor. If city is greater than 958.5 (based on the encoded numeric values we assigned to it earlier), only 3 of 3243 prescriptions are non-fraudulent. Inspections of these cities revealed that they were smaller cities with few medical providers. Another notable aspect is the Specialty decision node right off of the root node. Careful inspection of this node indicates that it only distinguishes one specialty which is dentistry, indicating that this specialty was the primary driver of fraud of this subset of prescriptions.

Inspecting the leaves, we see that the 5th from the left captures a large amount of fraud (26 legitimate vs 3181 fraud). We can see that the combination of city criteria uniquely specifies city 44, which is Miami. Moving down to the next decision node, we see that if Specialty is less than 12.5, a group that includes dentists and pain management specialists, there is a high rate of fraud.

State is also an important consideration. The State node on the left of the decision tree indicates that for some states, including Florida, North Carolina, Montana, and Colorado, a subset of cities in those states were found to have 62% fraud rate.

Further analysis will be needed in a future phase to understand the socioeconomic factors that these cities and states with higher levels of fraud have in common, but these insights highlight the most influential aspects of our model with respect to fraud detection performance.

Recommendations

Decision tree is the best supervised machine learning algorithm to predict fraudulent prescriptions as all evaluation metrics showed better performance compared to logistic regression (with and without log transformation).

K-means clustering is not recommended since it has two disadvantages: the number of groups have to be defined upfront, and the results are not consistent or repeatable over multiple runs of the model. We would not want to rely on a model whose results are not consistent in detecting fraud as the consequences of misclassifying fraud would be great.

Evaluation

We set out to build an analytical tool to identify the existence of fraudulent prescriptions. We were able to meet the original business objective by:

- joining various disparate data sets containing prescription information, excluded practitioners information, and drug information
- flagging prescription transactions that are fraudulent using excluded practitioners' information
- predicting transactions that are fraudulent based on existing fraudulent transactions

We were more interested in the prescriptions that were fraudulent as opposed to not fraudulent and focused on the resulting true positive rate. We determined that Decision Tree is the best model to use for our tool based on the performance metrics discussed above.

Deployment Plan

Further analysis will be required to determine the best deployment model for our tool. Consideration must be taken to fully understand end user behaviour and requirements. Possible options include:

- Real-time application program interface (API) service
- Batch API service
- Commercial Off the Shelf product requiring installation on the client side
- Webpage to facilitate business-to-business servicing

Ultimately, we would implement the Medicare fraud detecting analytical tool as a real-time product with the goal to monetize the fraud detection insights.

Our model trained on a balanced dataset of 50% fraudulent transactions and 50% not fraudulent. In reality, the number of fraudulent transactions is a small fraction of the total transactions so most of the time, our tool will predict that the transaction is not fraudulent. This fact will need to be kept in mind when deploying the tool to real users.

Opportunities for Improvement in a Future Phase

As with any data analytics project, there is always opportunity to try new things and make improvements. If we have the opportunity to work further with our customers on refining the tool, we would consider taking the following actions to improve our data and model performance and to expand the capabilities of the tool:

- Incorporate entity name when joining LEIE raw data file to Part D raw data file to see how joining the files on more/different features might impact the model outcome
- Incorporate Payments from Pharmaceuticals datasets to see if this financial information will alter the model performance and whether new/different insights can be gained
- Incorporate the other FDA product files (recall there were 9 and we used 1)
- Label our target variable from the outset as 1 = fraud and 0 = not fraud, rather than “good” and “not good” to improve efficiency of analysis
- Reduce the number of predictor variables input to the model (currently used 13) to achieve more optimal model performance and combat the Curse of Dimensionality which refers to the fact that as the number of features grows, the amount of data needed to generalize accurately grows exponentially.[5]
    - We can do additional research to better understand the extent of influence each feature has on identifying fraud to better discern whether it should be included as a model input.
        - For example, drug names are somewhat arbitrary and don’t prove a strong influence. We could consider joining the ATC (anatomical therapeutic chemical) information where each drug is assigned an ID based on chemical structure.

[5] https://www.kdnuggets.com/2017/04/must-know-curse-dimensionality.html

After factorization, this will likely prove effective for a hierarchical clustering algorithm.

- Reduce the number of classes when factorizing, ex. ActiveIngredient has 194 different names; other variables have 1000+ different names - this would assist in potentially reducing the number of splits in the decision tree
- Incorporate the opioid clustering results as another input that could differentiate between fraudulent and non-fraudulent prescriptions
- Include additional public domain datasets and/or initiate dialogue with various industry players to share and to consent use of non-public domain data
    - incorporate geolocation data to observe any trends based on geography using zip code or address
    - incorporate socioeconomic information about the practitioners such as income, etc.
- Develop a library of fraud detection patterns customized by industry player and services

References

Centers for Medicare and Medicaid Services - Part D Prescriber data

https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html

Centers for Medicare and Medicaid Services - Open Payments

https://www.cms.gov/OpenPayments/Explore-the-Data/Dataset-Downloads.html

Drugs@FDA database

https://www.fda.gov/Drugs/InformationOnDrugs/ucm135821.htm

LEIE Downloadable Databases

https://oig.hhs.gov/exclusions/exclusions_list.asp

Journal of Accountancy article on Benford's Law

https://www.journalofaccountancy.com/issues/1999/may/nigrini.html

BMC Bioinformatics, Drug clustering

https://bmcbioinformatics.biomedcentral.com/track/pdf/10.1186/s12859-018-2123-4

List of Extended-Release and Long-Acting Opioid Products Required to Have an Opioid REMS

https://www.fda.gov/downloads/Drugs/DrugSafety/InformationbyDrugClass/UCM348818.pdf