Machine Learning Nanodegree

capstone project

ChandraKanth Vadla

February 1st, 2017

1. Definition

Project Overview

This project was picked up from kaggel's Prescription-based prediction https://www.kaggle.com/roamresearch/prescriptionbasedprediction. The goal is to predict a doctors attributes by looking at their prescription behavior. The dataset was provided by a company Roam Analytics. They work in machine learning applications for health care. The data itself is derived from U.S. Centers for Medicare and Medicaid Services (CMS) publicly released the prescriptions made under Medicare Part D in 2013. For each provider the dataset contains all the drugs they prescribed more than 10 times in that year.

Problem Statement

Develop a prediction model which when provided with a prescription of drugs the model should be able to predict with reasonable certainty about the specialty of the provider.

Metrics

here for a given new record we are trying to predict is the specialty for the provider this can be one of the many classes of the target variable (specialty). This is a multiclass classification problem. A multiclass classification problem is a problem of classifying an instance into one of the more than two classes of the target variable. Multiclass classification problem should not be confused with multilabel classification problem. The multiclass classification problem makes the assumption that each sample is assigned to one and only one class ie., for example a fruit can be either apple or an orange but not both.

We have nearly 240000 rows spanning 282 specialties that mean an average of 850 records per specialty can be expected but we see large number of specialties with very few records to make contribution for the learning hence we all those specialties which have less than 900 rows.

The scikit learn api that we use for measuring the prediction accuracy of the model we create is

```
sklearn.metrics.accuracy_score()
```

in principle, the accuracy_score is a simple technique to measure the percentage of the predictions that are made correctly on the test data.

2. Analysis

Data Exploration

Each record in the dataset is a JSON object with following schema

```
{
    'provider_variables':
        {
            'brand_name_rx_count': int,
            'gender': 'M' or 'F',
            'generic_rx_count': int,
            'region': 'South' or 'MidWest' or 'Northeast' or 'West',
            'settlement_type': 'non-urban' or 'urban'
            'specialty': str
            'years_practicing': int
        },
     'npi': str,
     'cms_prescription_counts':
        {
            `drug_name`: int,
            `drug_name`: int,
            ...
        }
}
```

Cms_prescription_counts lists all the drugs and the quantities prescribed. Npi stands for National Provider Index, this value is unique for each record. It can be considered as the unique identifier for the record. The provider_variable gives the information related to the doctor(provider) who issued the prescription. The sub-components of provider_variable and the type and purpose are self-explanatory from the above figure.

There are no duplicate records, this I verified by the fact that there are n repetitions of any npi values. And there are no missing data because there are no records with empty cms_prescription_counts.

In all we have 239930 records with 2397 unique drugs and 7 provider variables (`brand_name_rx_count, generic_rx_count, region, gender, settlement-type, specialty, years_practice`). Of these 7 provider variable any one of them or a combination of them or all of them together can be treated as the target variable. For the purpose of this project we consider only specialty as the target label. The 2397 unique drugs will become the feature columns for this dataframe created by reading this dataset.

```
In [13]: #parts1.append(' ')
         df_names_check = pd.DataFrame(parts1, columns=['titles'])
         counts = df_names_check.groupby(["titles"]).size()
         display (counts)
         print "total number of rows is:", len(df_names_check['titles'])
         print "cms_prescription_counts + provider_variables", counts['cm

         titles
         cms_prescription_counts    2397
         provider_variables            7
         dtype: int64

         total number of rows is: 2404
         cms prescription counts + provider variables 2404
```

Therefore, a total of 2397(features) +7(label)+1(npi) = 2405 columns are present in our dataframe.

```
In [3]: dataframe_main.shape
Out[3]: (239930, 2405)
```

The related code for data exploration and data visualization are available in the jupyter-notebook file data_exploration_visualization.ipynb.

The target variables and their relationships

The dataframe has 7 target variables

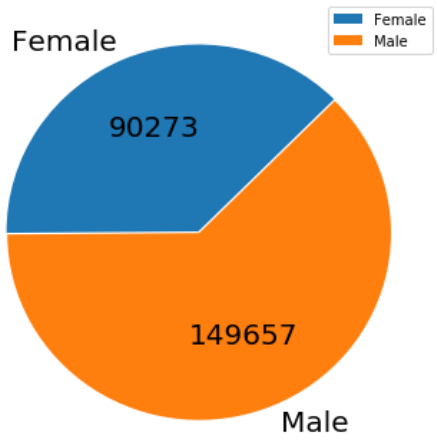
	brand_name_rx_count	gender	generic_rx_count	region	settlement_type	specialty	years_practicing
0	384	M	2287	South	non-urban	Nephrology	7
1	316	M	1035	West	non-urban	Nephrology	6
2	374	M	2452	Northeast:	urban	Gastroenterology	5
3	683	M	3462	Midwest	urban	Psychiatry	7
4	143	M	2300	Northeast:	urban	Psychiatry	7

The column npi does not provide any information hence can be dropped. Though the labels brand_name_rx_count, generic_rx_count, gender are not useful for the purpose of this project we hold them for any further refinements.

The gender variable

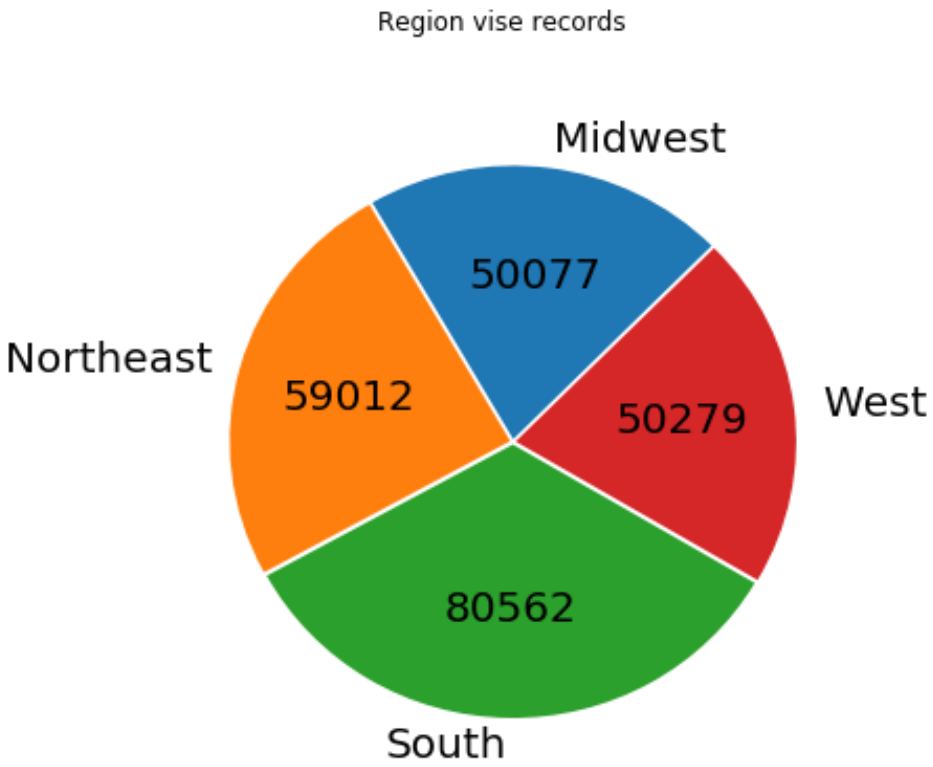
There is a near 50% increase in the prescription by male doctors compared to the female doctors

	gender	0
0	F	90273
1	M	149657



	0
count	2.000000
mean	119965.000000
std	41990.829094
min	90273.000000
25%	105119.000000
50%	119965.000000
75%	134811.000000
max	149657.000000

The region variable

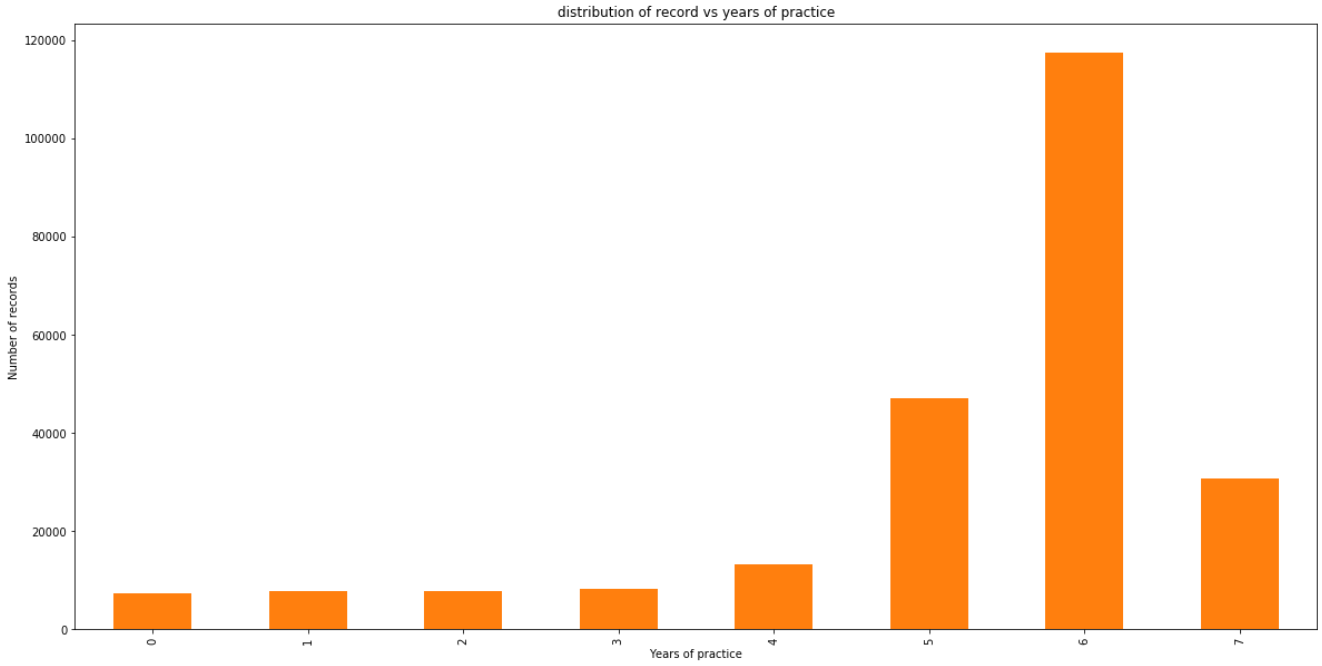


	record_count
count	4.000000
mean	59982.500000
std	14337.997687
min	50077.000000
25%	50228.500000
50%	54645.500000
75%	64399.500000
max	80562.000000

This likely traces to a systematic bias in the CMS dataset: it's a record of Medicare payments and hence mostly records data on older patients, who are over-represented in the South, especially in Florida.

Years of practice

Here is the distribution of the years of practice of the providers

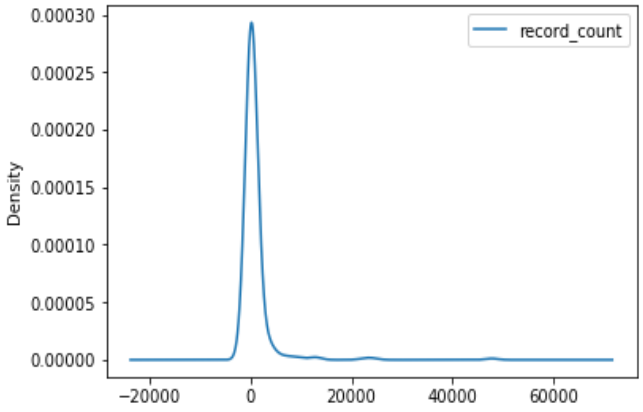


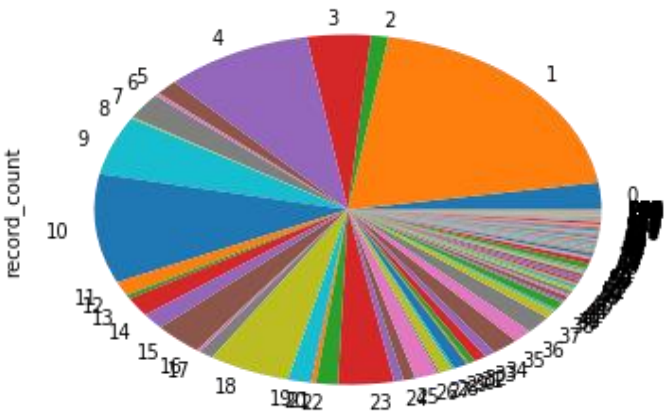
The specialty variable

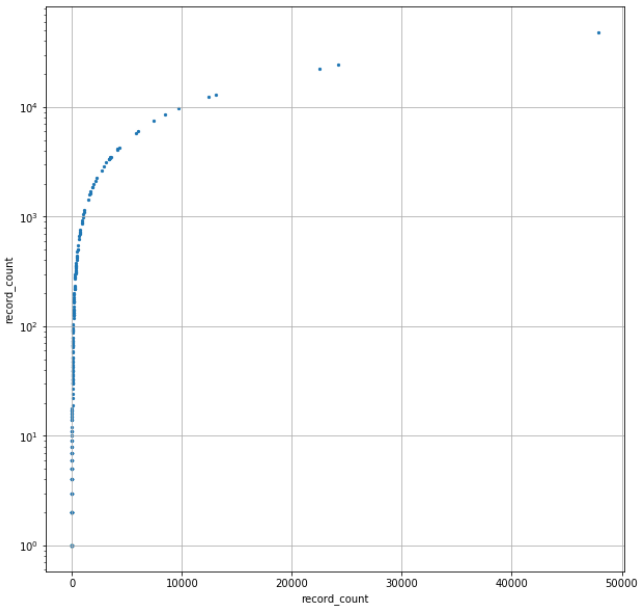
Of the 282 specialties a large number of them have very few records to make any unbiased analysis.

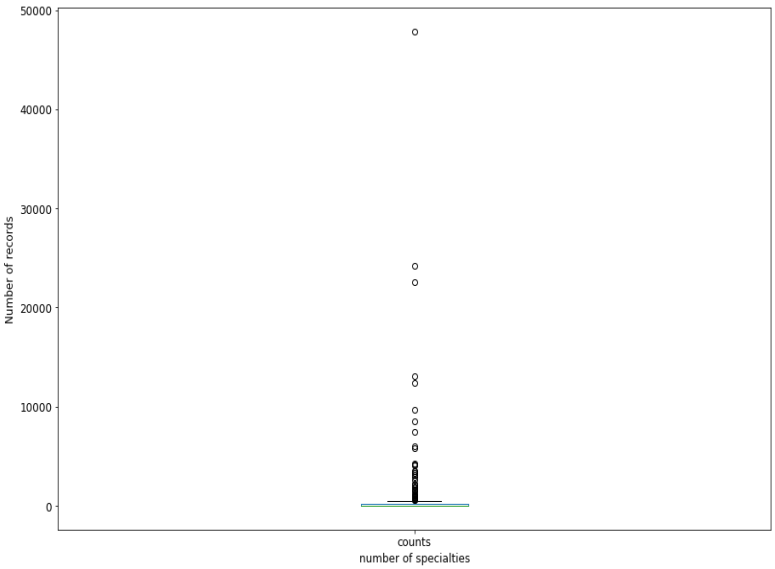
```
number of specialties with rows fewer than
100 are                  189
200 are                  21
300 are                  10
400 are                  11
400 to 1000 are          18
1000 to 2000 are         10
2000 to 3000 are         4
3000 to 4000 are         6
4000 to 5000 are         3


greater than 5000        10
```









In fact a closer inspection revels that nearly 100 specializations have less than 10 records. With 239930 record over 282 specialties would mean an average of 800 records per specialty. But some specialties have a large number of record like general medicine has 47000 records while many specialties have less than couple of hundreds of records. This is a highly imbalanced data. Hence from here on we consider only those specialties that have more than 1000 records.

	record_count
count	282.000000
mean	850.815603
std	3762.523437
min	1.000000
25%	3.000000
50%	23.000000
75%	214.000000
max	47836.000000

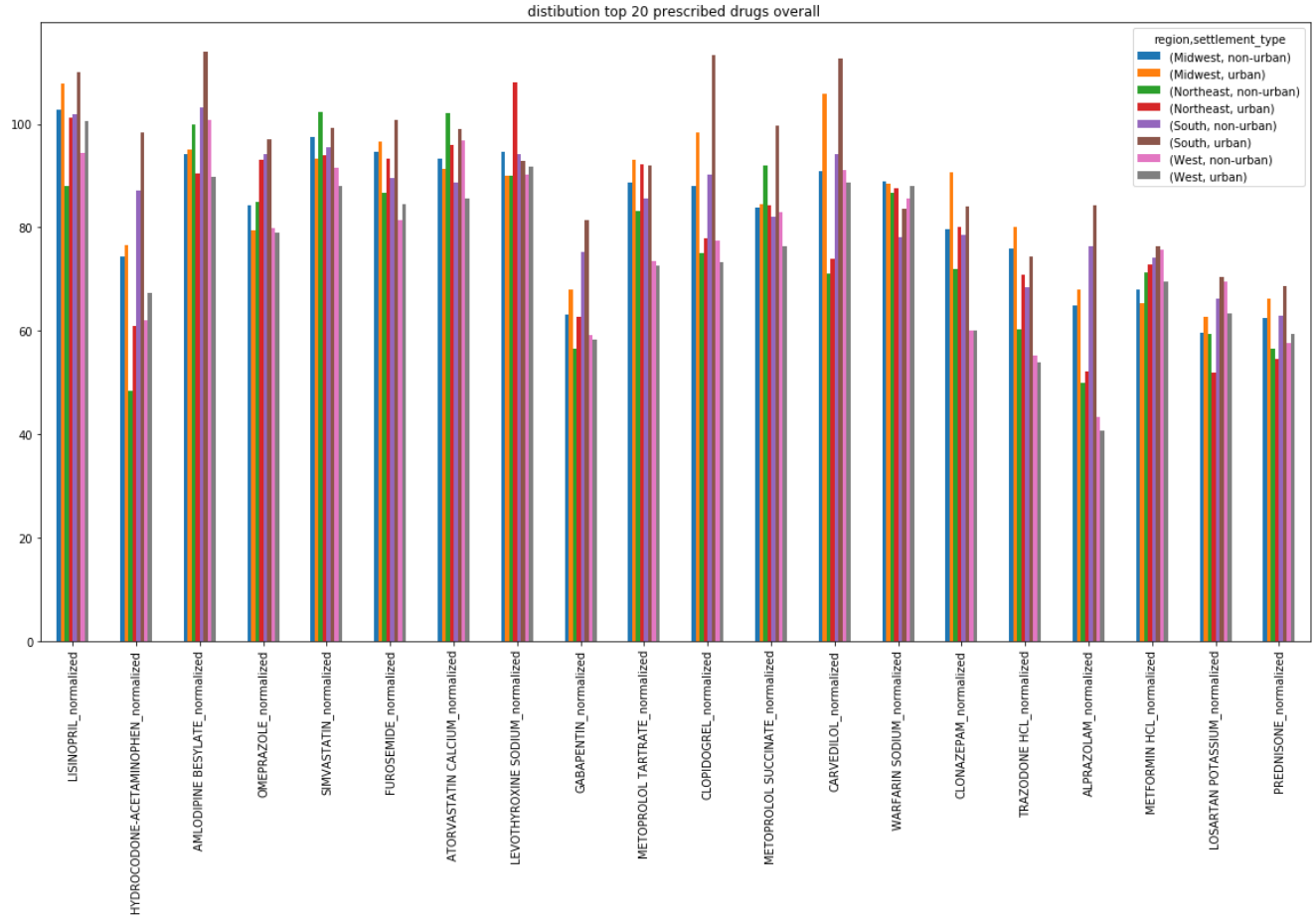
Exploratory Visualization

Different set of drugs might be prescribed by different specialties, which means drugs that are used to treat more commonly occurring health issues might have a significant representation in the dataset also drugs that might be prescribed by multiple specialty also will also have high representation.

Therefore, the investigation to check if there is any preference towards some drugs compared to others needs to be done at two levels

1) A holistic investigation of drug prescription pattern across all specialty

   This chart show distribution of top 20 drugs prescribed in the dataset.



A visual inspection of the above chart easily shows that there are no peaks that are at-least greater than 20% or lesser than 20% of the mean of that column. Hence here nothing suggests that there is a bias towards or against some drugs in certain regions/settlement-types. Therefore it would be a good idea to make the same investigation individual specialty level.

2) A deeper investigation at individual specialty levels

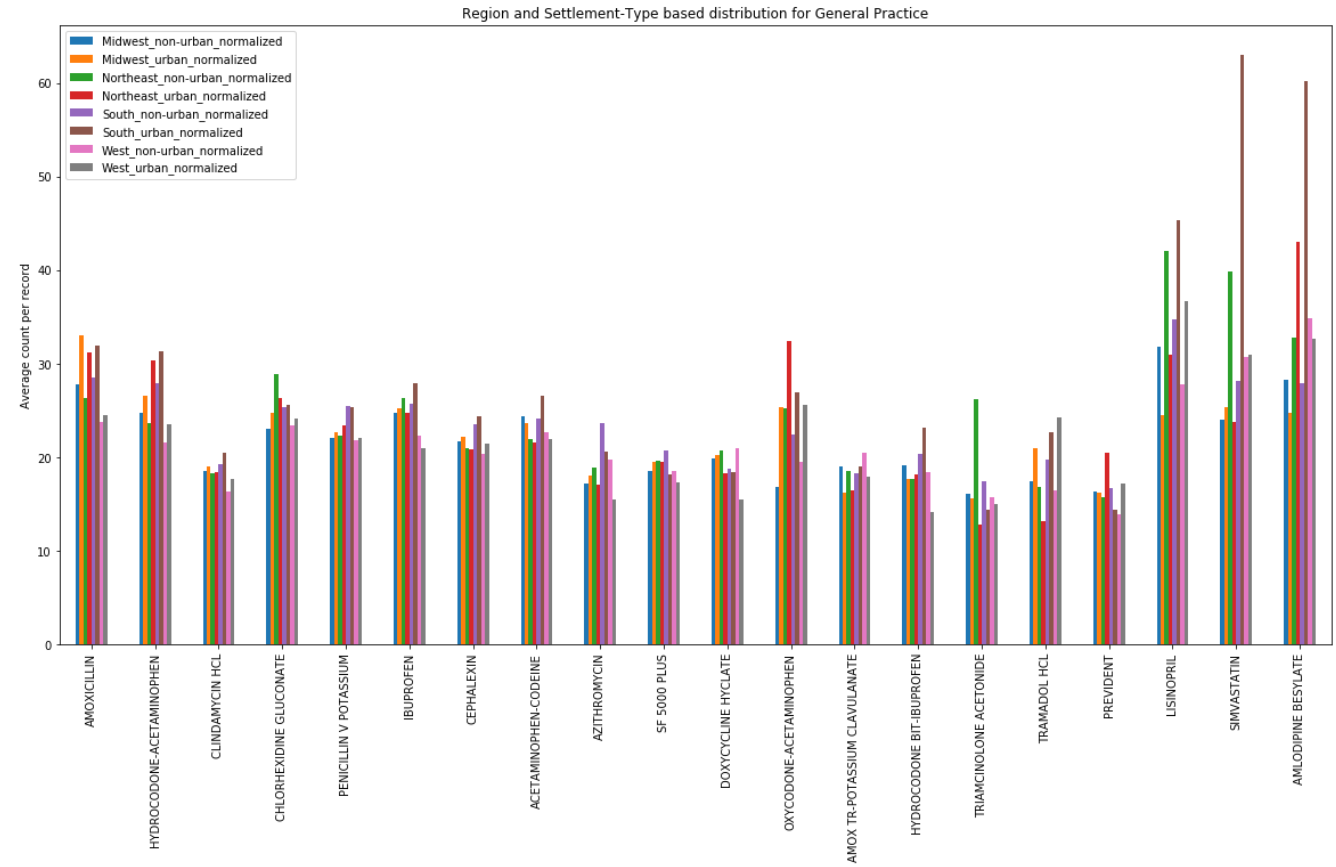
For this study, we make a three-way analysis

1) region-based (appendix I)
2) settlement type based (appendix II)
3) a mixture of both region and settlement type

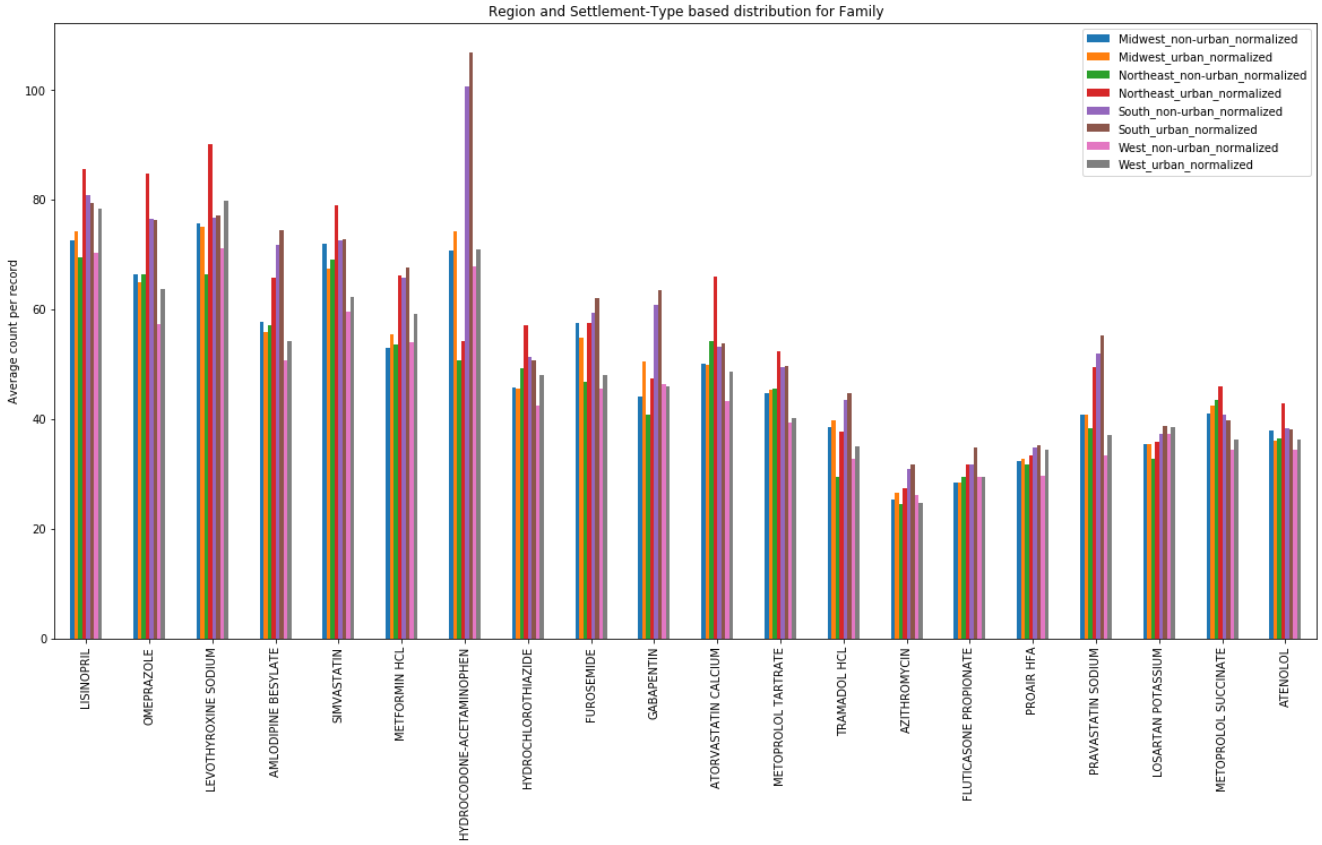
The method to achieve this is to retrieve all the rows of a specialty this is done in the function ABC in the file dataExplorationAndAnalysis.ipynb . from the new data frame

only keep those columns which have any valuable information that is remove all the columns which have only in Nan values. Now check how individual drugs are represented in in this new data set and if there are any drugs that are having very few values we can ignore them as they do not to contribute much information hence we consider only those columns which have any significant information. Next, we group these results by settlement type and/or region and get a cumulative sum for each column. when we plot these values, it is possible that some drugs stand out clearly this might be because there are a large number of entries for these drugs in that particular settlement type and/or region compared to other settlement types or regions this might happen due to the population density of an area likes City vs rural. To overcome this effect, we normalize drug prescription count record for each column. this gives a better estimation of the prescription pattern. if there are any significant Towers compared to their neighbors in the same category that implies there is a bias towards prescription of those drug in that settlement-type or region.

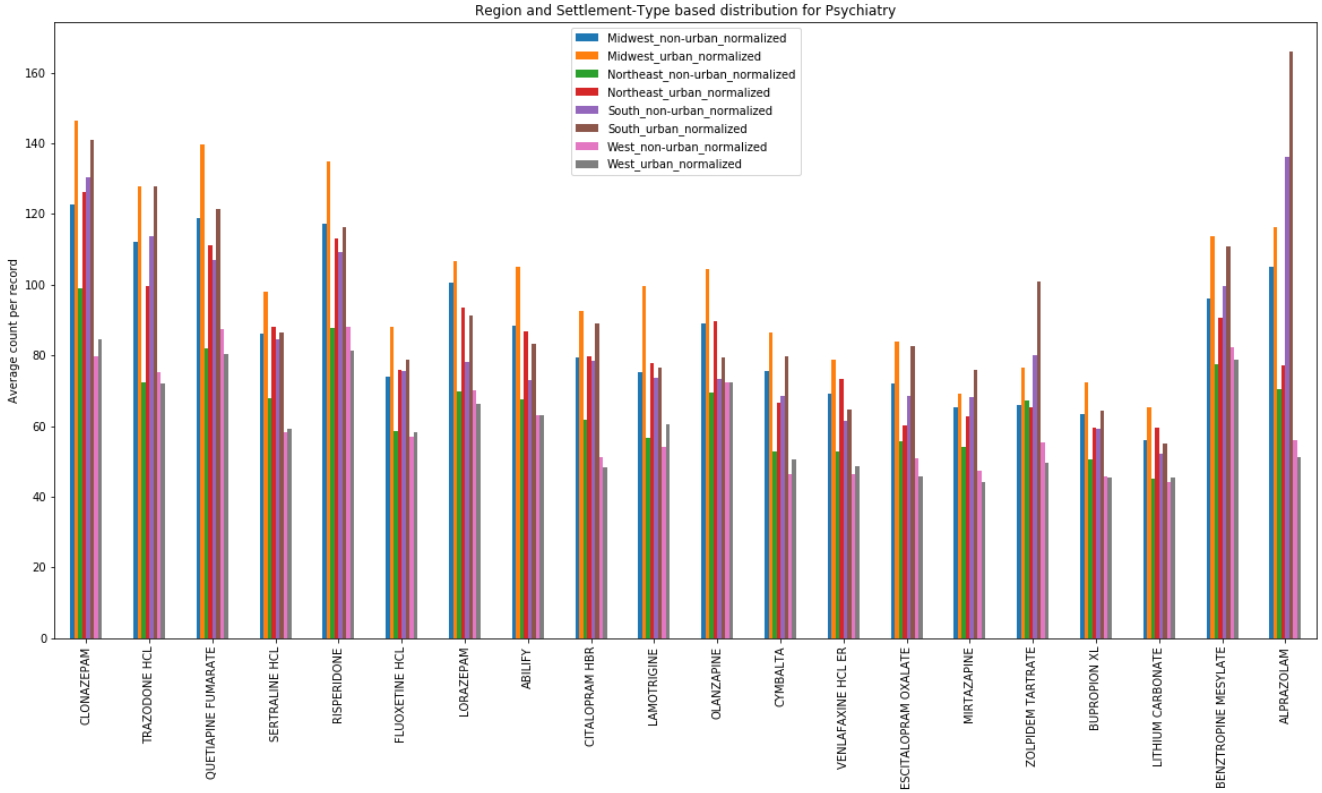
For the project we examine only top 10 specialties with highest records.



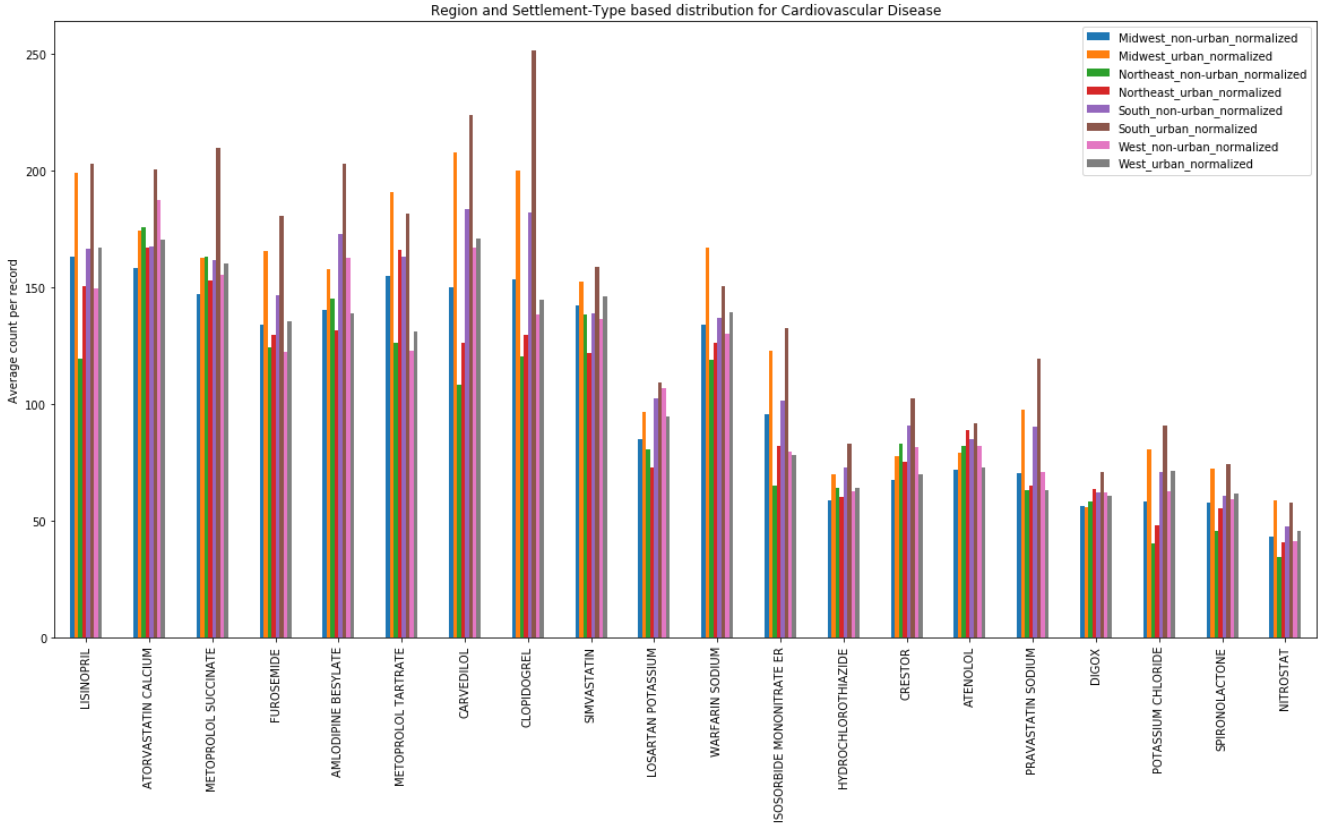
For the drugs SIMVASTAIN and AMLODIPINE-BESYLATE, the south urban really stands out very prominently. These drugs are primarily used to treat cholesterol related problems. This might suggest that a lot of obese people are around south urban areas. The magnitude of the difference seems to ring alarm bells.



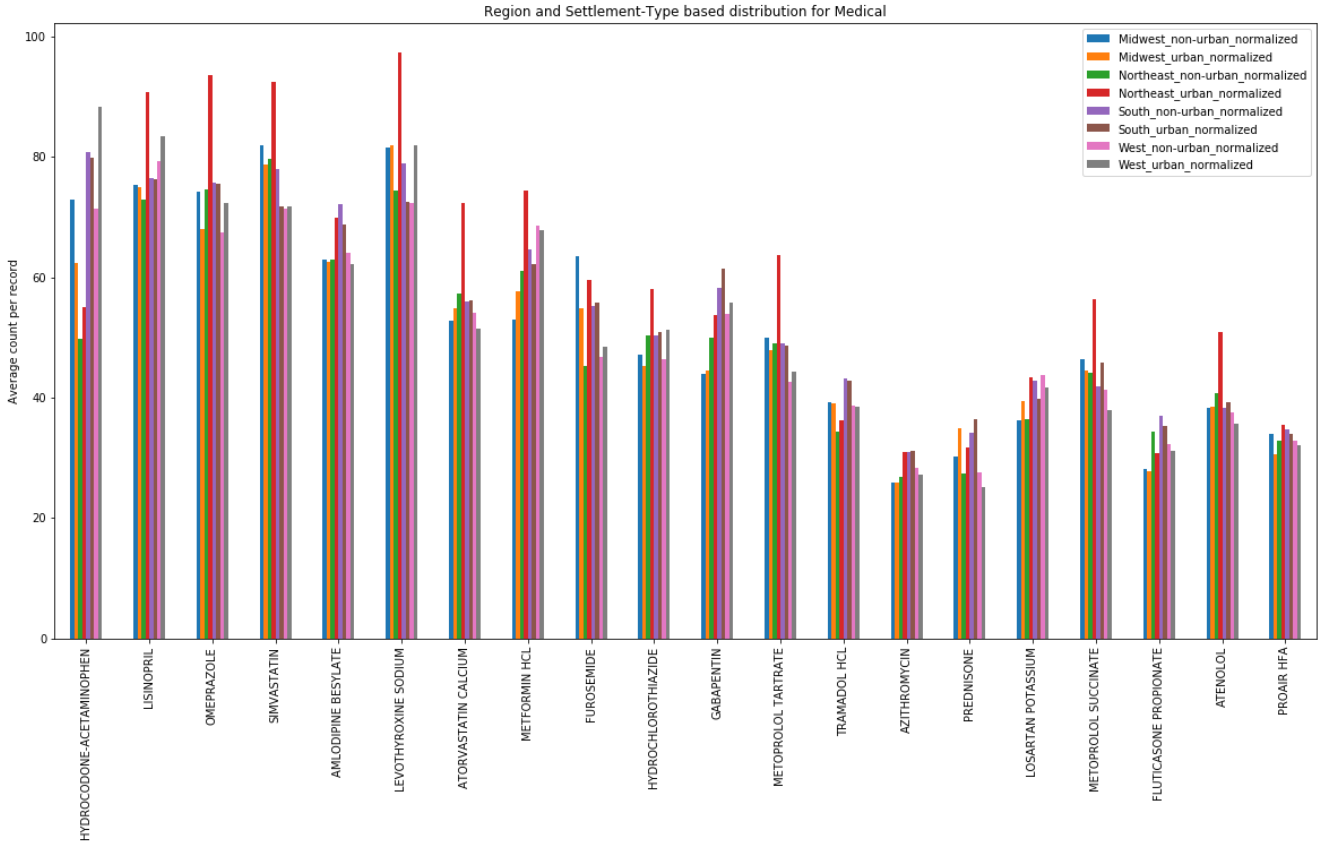
Here we can see HYDROCODONE-ACETAMINOPHEN dominates in both urban and non-urban areas of south region. A simple google search for this drugs shows that it is used to relieve moderate to severe pain. It is clear that Family doctors of south region are prescribing more pain-relieving drugs than other regions. This makes a case for further investigation for possible explanations, like was there any epidemic in that region.



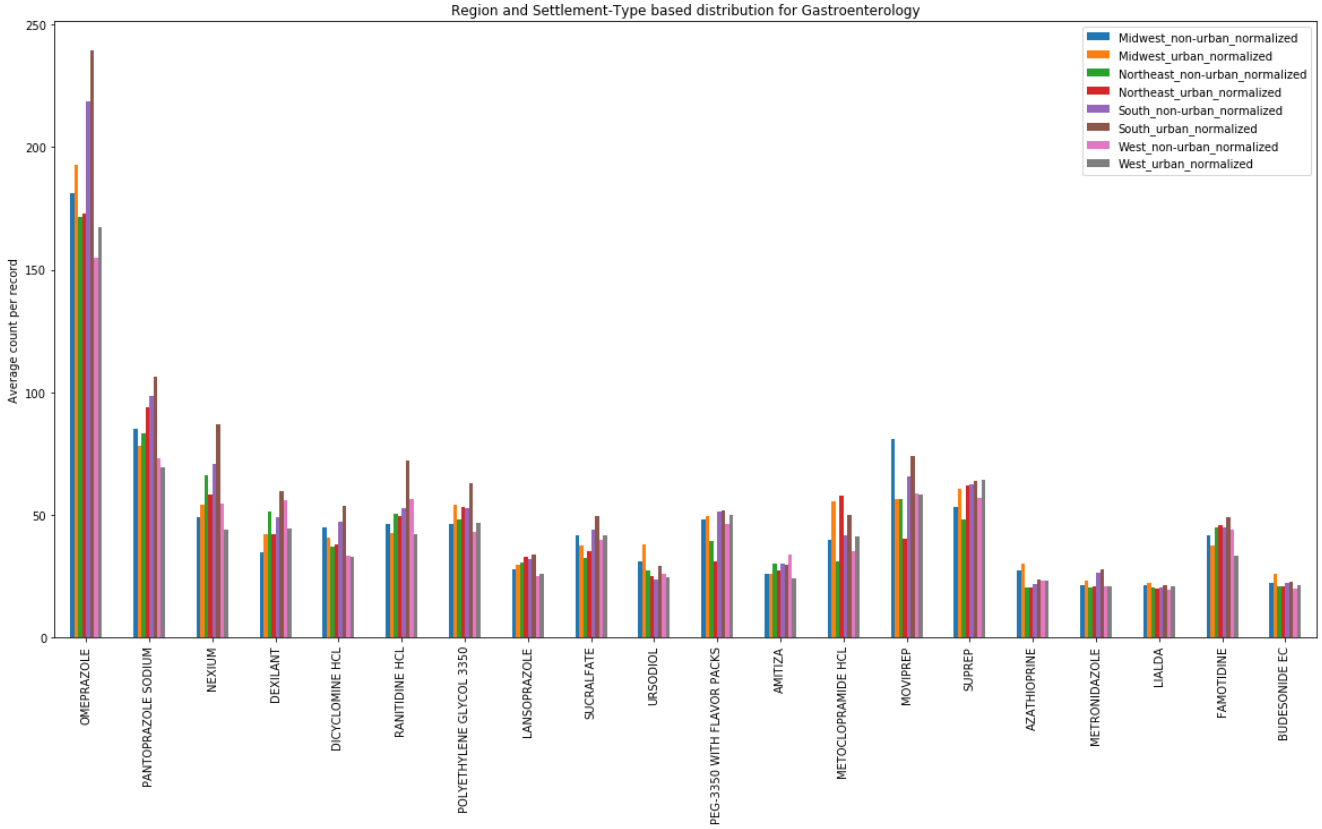
For psychiatry, can see that Midwest urban stands out generally across all the drugs. Perhaps this might be indicative of widespread psychiatry problems in this area.



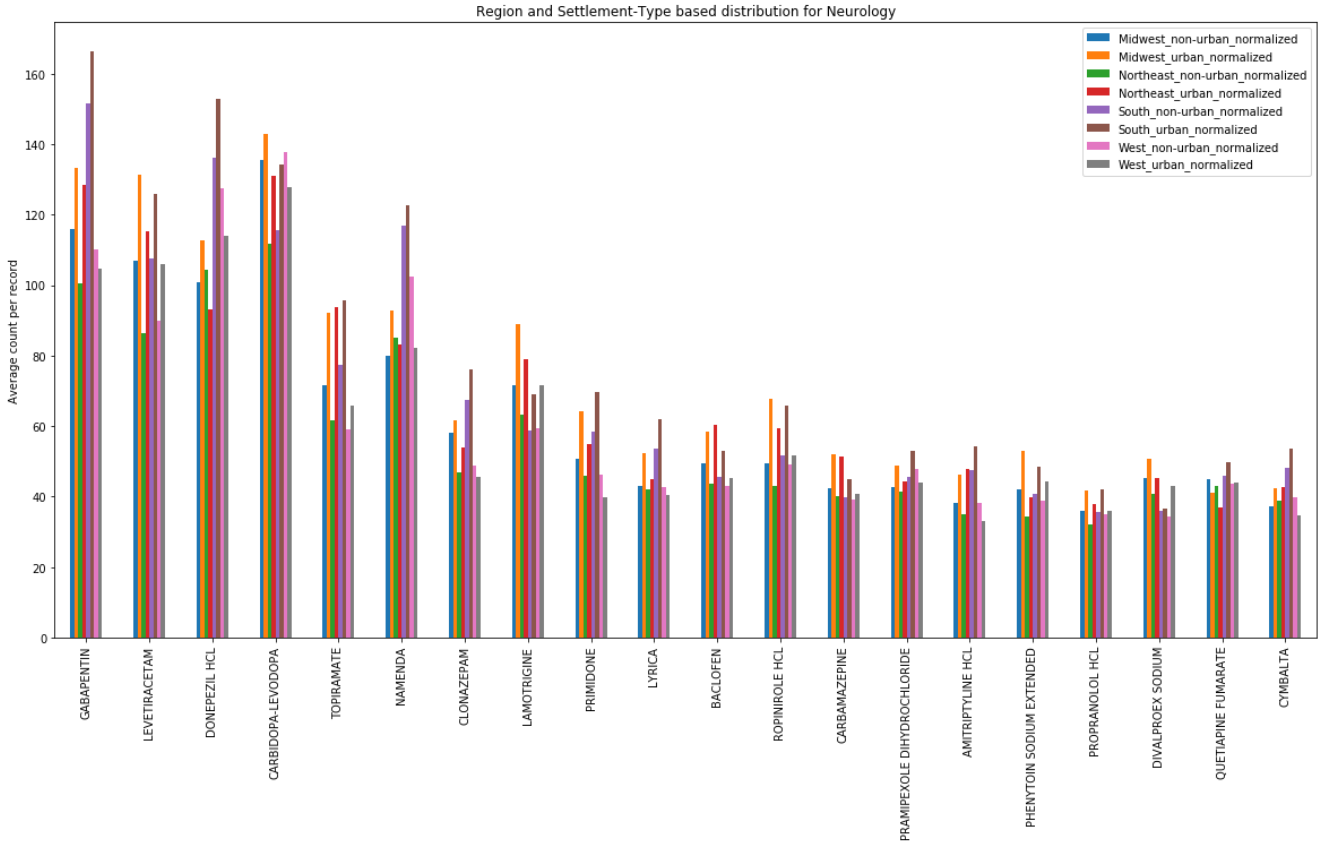
CLOPIDOGREL does look to standout but the values on y-axis shows it's only a some 10% rise and that's not alarming.



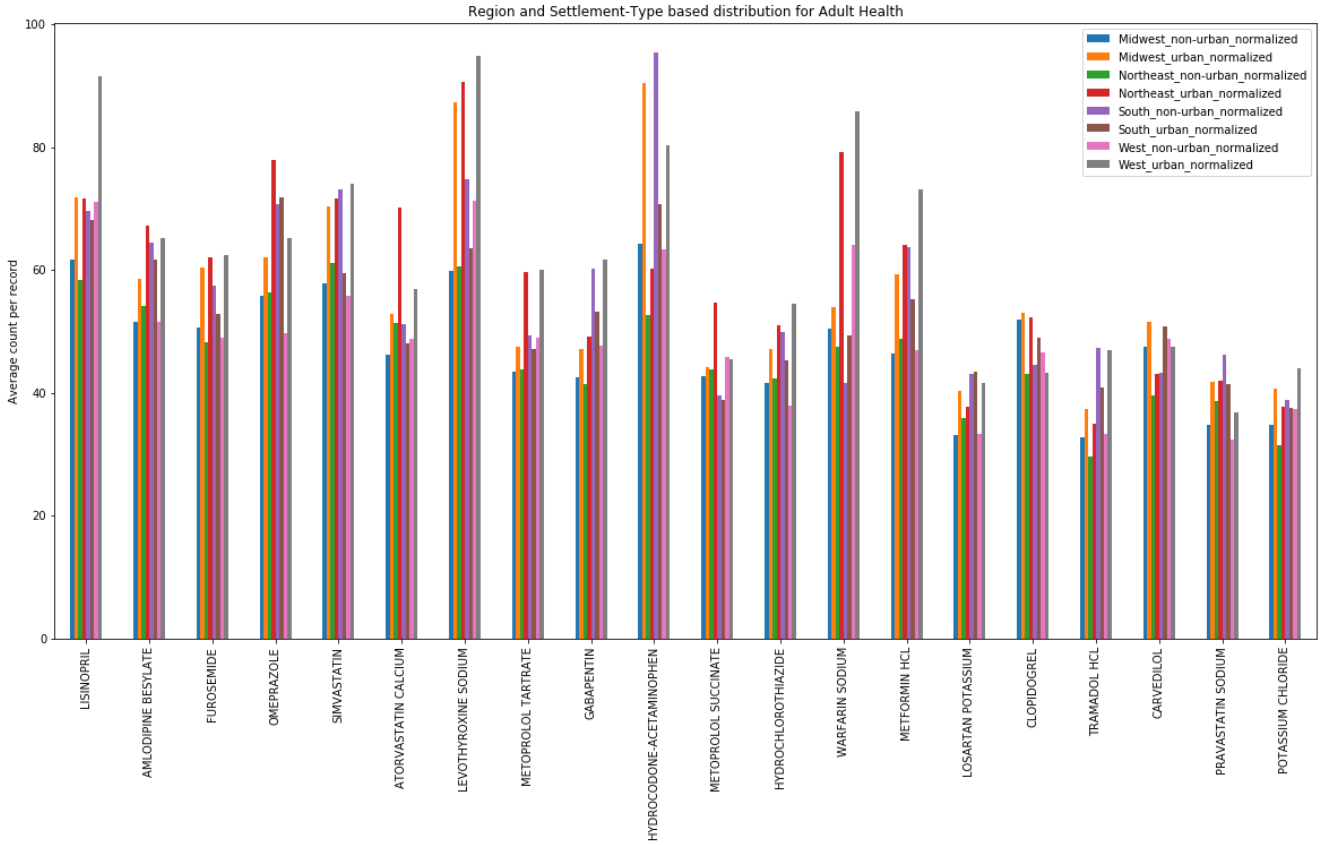
Though Northeast Urban slightly seems to domination across most drugs it does not make much of a case



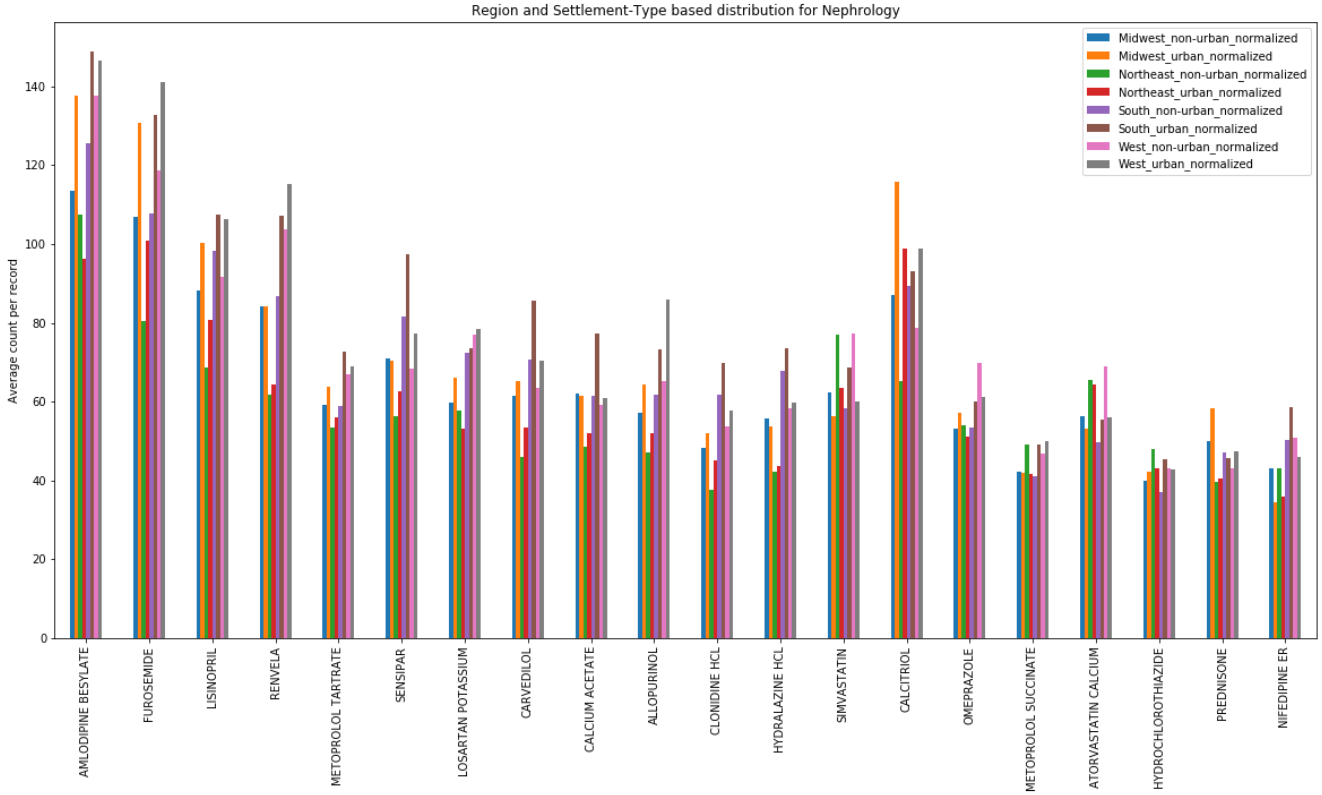
Though south region dominates for the OMEPRAZOLE, the peak doesn’t look alarming



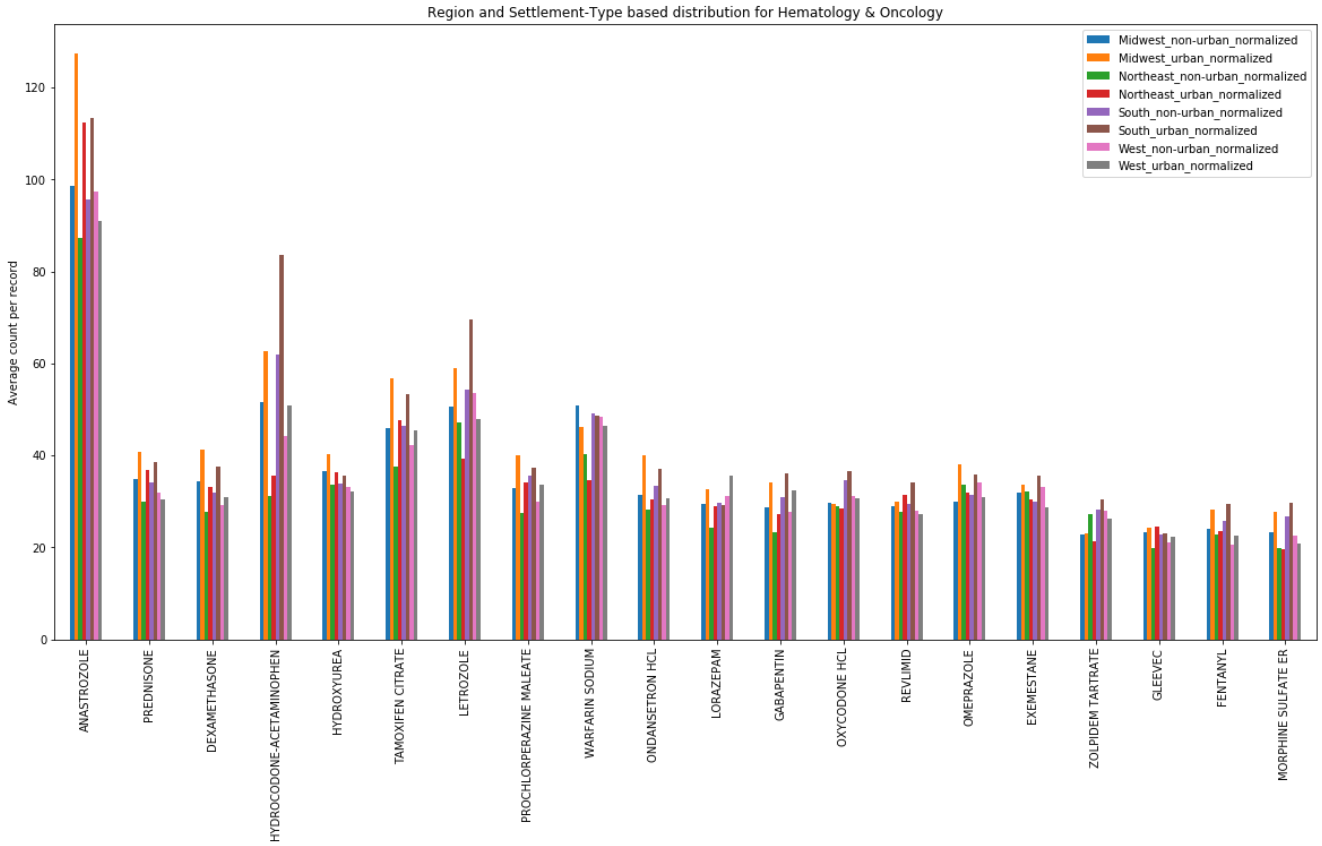
A similar kind of representation by all regions can be seen across all drugs.



For LISINOPRIL, the west urban tower really stands out contrasting to others. LISINOPRIL is user to treat hypertension. This might be an indicative that people in west urban are subjected to more tension than people of other areas. Coincidentally this is also the area with more computer professionals. So work life imbalance can be a possible cause for this peak.



A similar kind of representation by all regions can be seen across all drugs.



but in the graphs, that are plotted we cannot see any significantly tall Towers which are at least 20% bigger than the neighbor hence we can conclude that there is no significant Biased towards prescription of some drugs over others.

Algorithms and Techniques

The classifier we use to solve this problem is xgboost. It is a free and open source software library available under the permissive apache2 license. It is a very fast algorithm for classification and regression. Unlike the traditional algorithms which are inherently sequential in nature xgboost can be parallelized and run in a distributed environment taking advantage of the multi core architecture of the newer hardware. xgboost is very good with structured datasets for classification and regression predictive modelling xgboost internally uses gradient boosting decision tree algorithm. Boosting is an ensemble technique where new models are added to the existing models to correct the errors. Gradient boosting uses of gradient descent algorithm to minimize the losses when adding new models.

Some key features xgboost are

a. Parallelization: can use all CPU cores during training for tree construction
b. Stochastic: gradient boosting the model can sub-sample at rows, columns and columns per split levels
c. Sparse awareness: the model is very good with automatically handling the missing data

Why XGBoost?

We have a large data set with very high dimensionality and it is highly sparse in nature. The above mentioned three key features of the xgboost are the primary reasons for our choice of this model. we have a sparse dataset with high dimensionality so we want a model that can handle sparse data and it should take advantage of the multicore architecture of the hardware. With a large number of feature columns that we have it is a sensible to go with a tree based techniques that supports column level subsampling.

Benchmark

A simple GaussianNB() classifer was used to set a benchmark and the accuracy we obtained is 0.183314

3. Methodology

Data Preprocessing

Data preprocessing is done in the data-preprocess.ipynb file. The original dataset is fairly well organized and consistent as mentioned earlier there are no missing values or duplicate records the data needed very few tweaks like commas at the end of record. As the Jason schema is nested in structure the json_normalize() api of json library creates column names “cms_prescription_counts.” prefixed to the actual drug name and “provider_variables.” prefixed to the label name like specialty. we can rename the column names by removing these common prefixes this is done by the function column_name_prefix_remover(). The column ‘npi’is a unique identifier to the records and does not contribute any information so we remove it. Using the function check_for_column_name_repetitions() we confirm that there are no empty string names and there are no duplicate column names.

Implementation

From the dataframe_main dataset remove the columns npi, brand_name_rx_count, gender, generic_rx_count, region, settlement_type, years_practicing. Next pop out the target variable speciality into “label”. we can see that there are no missing values in the target label and next quick look at the target variable shows that it is multiclass in nature. All the class names as string type and well defined.

The xgboost model expects label to be of numeric form. Hence we use a labelencoder() to convert all the string classes to numerical classes. Label encoder takes a 1-d array argument and returns another 1-d array as numerical representation of the in classes. These is the reason to choose labelencoder over other popular encoders like onehotencoder, that the labelencoder produces a 1-d array as output rather than a multicolumn output by onehotencoder.

New we use train_test_split from sklearn.model_selection.train_test_split to the features dataset and the target label are split into 80%-20% for training and testing datasets(X,train, y_train, X_test, y_true)

we can directly use the new training feature set(X_train) and training label column(y_train) to train our xgboost . On making such an attempt to it was immediately

clear that this technique will need immense computational resources and time to train the model. Therefore, we need to reduce the number of a features in the training data set. We discarded these training and testing datasets and labels went back to the dataframe_main.

A popular feature reduction technique is Principal component analysis (PCA). PCA is used for dimensionality reduction by projecting the data to a lower dimensional space. That is PCA produces a sequence of new features called components these components are arranged in the magnitude of the variance they capture from the different dimension of the original data.

For the dataset we have, PCA does not suit well because

PC at does not handle Nan values

1. To handle these Nan values if we replace them with zeros, we are artificially introducing a large variance and
2. We can replace Nan values with a mean value of the column but even then we will still influenced the first dimension of the PCA.

Hence an alternative to PCA has to be explored

In a different approach if we can imagine our data set as a bag-of-words from Natural Language processing(NLP).

1. each row represents a document
2. columns represent the words
3. row entries for the columns represent the frequency of that word in that document.

Now we can easily handle the Nan values by simply replacing them with zeros yet to do not face any side effects. Next we pass this dataset to TfidfTransformer() . A Tfidf Transformer transforms count Matrix into a normalized tf-fidf representation this output is a sparse matrix.

This sparse Matrix can be fed into a Non-Negative Matrix factorization algorithm like sklearn.decomposition.NMF. The NMF algorithm essentially finds 2 non negative matrix(W,N) whose product approximate the non-negative input matrix X. This technique is generally used in dimensionality reduction or topic extraction. This way we reduce the number of input features from 2397 to 40. The reason to select 40 is explained in the next section Refinement.

Now that we have reduced the feature set we can easily train xgboost model the parameters used for model are also explained in the next section. once the model is it trained on the training dataset(X_train). The model is used to predict the testing dataset x_test. The results of the prediction are captured in y_pred and compared to y_true using accuracy_score() method from sklearn.metrics.accuracy_score.

Refinement

This part of the program is implemented in the refinements.py file. This takes a lot of time to run hence it is not a good idea to implement it in the Jupiter notebook because due to any reason if the browser web page is refreshed all the computational results would be lost.
There are three import important components here

1. strtifiedKfold
2. gradesearchCV
3. NMF

StarifiedKfold is a cross validator used in the grid search CV for model validation. In principle it works by making sub-samples of the input data called folds and except for one all the folds are used to train and the remaining 1 fold is used to test. This process is repeated with each fold as a test fold. That means if nsplits is 10, then there will be 10 folds and 10 iterations of training and testing inside the cross validator.
GridSearchCV takes a param-grid as an Input and exhaustively tries all possible combinations of parameters with the model to search for the best estimate. Therefore if the param-grid has three parameters each with 5 possible values then the grid search CV will run the model with 5^3 = 125 combinations of parameters.
NMF as told earlier is used for feature reduction but we don't know beforehand what would be the good feature set sizes for the model therefore we have to iteratively try the grid search with different features set sizes produced by NMF.
The results of the trial runs of the grid search CV over different features set sizes are listed below

Trial with top 5 specialties with most records. X_train shape 157621,2141

NMF Features	Colsample_by_level	Scale_pos_weigh	Accuracy_score
50	0.7	Default	0.817573
100	0.5	Default	0.815480
150	0.6	Default	0.814401
200	0.7	Default	0.813957
250	0.6	Default	0.813259
300	0.7	Default	0.812181

350	0.7	Default	0.811229
400	0.6	Default	0.810278
450	0.5	Default	0.809960

Trial with top 3 specialties with most records. X_train shape 120132, 1871

NMF Feature Components	Colsample_by_level	Scale_pos_weigh	Accuracy_score
50	0.4	0.1	0.85528
100	1.0	0.1	0.853706
150	0.7	0.1	0.851917
200	1.0	0.1	0.850127
250	0.6	0.1	0.850002
300	0.3	0.1	0.849919

Some of the obvious observations are

1. scale_pos_weight which is learning rate is always 0.1for best results and
2. colSample_by_level value is one of 0.5,0.6 or 0.7 for best estimator

hence, we take these parameters to better tune our final model.

4. Results

Model Evaluation and Validation

In the refinement section we have mentioned that the best looking values for the colsample_by_level might be one of 0.5, 0.6 or 0.7. So I went ahead and tried the xgboost model with all three colSample_by_level values.

sensitivity analysis: From the results shown in table-2 of the refinements section accuracy score was over 0.85. This was with a training dataset that had only three classes in the target label. And in the table-1 the accuracy score was little over 0.81 and this dataset had 5 classes. And with the final dataset of 35 classes, covering almost all the important classes, an accuracy score of 0.69 is very good considering the fact that GaussianNB classifier gave a score of just0.018 on the same dataset.

The final parameters for the model are

colSample_by_level=0.6 and Scale_pos_weigh = 0.1

Justification

Following are the results for the benchmark score(gaussianNB) and the final model xgboost

	Accuracy score
GaussianNB	0.183314
XGboost	0.697158

We can clearly see that our choice of the model that uses decision trees for column sampling and gradient boosting for ensemble was in the right direction. Club that with NMF for feature selection has really given us a tremendous increase in the accuracy score.

5. Conclusion

Reflection

In this project, we first tweaked the raw data set so that it is easily compatible with the Json libraries. Next the data set was read into a pandas dataframe and then we did some cleanup to remove the prefixes in the column names and we checked the for any duplicate records there were no duplicate records and duplicate column names all the column names were unique.

Next, we did some Data exploration and analysis by plotting graphs of top ten drugs prescribed among the top ten specialties according to the region and settlement types. We did Discover a couple of interesting and Peculiar facts.

Next, we created a benchmark score using Gaussian NB. Then we proceeded to create a predictive model using xgboost this model was it trained with a dataset of all the specialties that has at least 900 records. we had" specialty" as a multi class target label with 35 classes.

Then we observed that PCA was not conducive for our data set to reduce the features. We had to explore other ways and technique. What I came up was to use the bag-of-words technique of NLP.

We treated the dataset as a bag-of-words and replaced all Nan's with 0's and processed the dataset with Tfidf() and then used NMF to extract reduced feature set to train the xgboost model. The final model was evaluated using accuracy score.

One of the interesting aspect of that was seen in this project is the necessity to reduce the feature set and PCA was not helpful due to the nature of the data set. I had to explore other means and the bag-of-words technique, usually used for natural language processing, had come in handy.

Some of the difficulties are ram and computing power requirements. The data set in its raw format was only 32 MB but when it is loaded into a dataframe it occupied nearly 9 GB ram and during this loading process it hit as high as 14 GB this definitely could not be handled by my personal laptop of 8GB RAM. Earlier I explore the option with breaking the raw files into smaller sizes and then reading and appending individual files into Sparse dataframe. This is work initially but then there was no room to do any further operations.

This made me explore the cloud based options like Microsoft Azure. With the 1 month free trail subscription provided Microsoft I could manage a Ubuntu Virtual Machine with a ram of 28Gb and 4 core processor. Another thing I got to figure out was how to make jupyter-notebook run on the server and access it remotely.

Improvement

one of the improvement that can be easily extended with the predictive model that was created in this project is multicolumn target, that is instead of only 'specialty' we could have had more columns like 'years_practice' , 'gender' , 'region' and 'settlement-type' as target variables.
Another in improvement which I think is possible is to use neural networks for predictive analysis rather than I believe neural networks will give us even better results.