Health Insurance Marketplace

Springboard Data Science Career Track Program
Capstone Project #1

Author:	*Ashish Mohan Sharma*
Reviewer:	*Srdjan*
Publish Date:	*02/27/2019*

Table of Contents

Introduction:

Health insurance matters to all of us. Most of the United States population have healthcare insurance but not till 2010. There were more than 16% people in United States who were not insured. On an average, US Citizens spends more than $1000 on monthly premiums for family coverage. Such a high premium leads people to not to opt for the Health Insurance which makes them vulnerable to financial breakdown in case of medical emergencies. To mitigate such problems, the **Patient Protection and Affordable Care Act** (**PPACA**), often shortened to the **Affordable Care Act** (**ACA**) or nicknamed **Obamacare**, was introduced. **ACA** became fully operational in 2014, consumers now have the option to purchase different government regulated health care plan that complies with the ACA.

In the United States, health insurance marketplaces, also called health exchanges, are organizations in each state through which people can purchase health insurance. It aims to promote individuals to compare and choose from a range of available health plans and select the one that is most suitable. In the meanwhile, the healthcare reform act proposed the “Meaningful Use” incentive. As a result, much of the data was made public online, and became very easy to access.

Given the massive database is available online, I can take full advantage of this resource to analyse and try to unravel the hidden invaluable information. The insurance companies always keep pricing decision in a black box, so by this study I will try to discover those patterns using numerous methodologies of Exploratory Data Analysis, Data Visualization and Predictive Analytics. I try to predict the rates of the plans for the next year and suggests the best plan to enroll in as per the individual needs.

Data Acquisition:

The Centers for Medicare & Medicaid Services (CMS) Center for Consumer Information and Insurance Oversight (CCIIO) is committed to increasing transparency in the Health Insurance Exchange. While health plan information including benefits, copayments, premiums, and geographic coverage is publically available on Healthcare.gov, CMS also publishes downloadable public use files so that researchers and other stakeholders can more easily access Exchange data.
The Health Insurance Exchange Public Use Files (Exchange PUFs) are available for plan years 2014 to 2019 to support timely benefit and rate analysis. The link to the dataset is below:

https://www.cms.gov/cciio/resources/data-resources/marketplace-puf.html
http://www.nber.org/data/cms-marketplace.html

The Exchange PUFs consist of ten separate files but for our study I will be concentrating on below files:

1. **Plan Attributes file** : Plan-level data on essential health benefits, coverage limits, and cost sharing
2. **Rate File** : Plan-level data on individual rates based on an eligible subscriber's age, tobacco use, and geographic location

3. **Benefits and Cost Sharing file**: Plan-level data on maximum out of pocket payments, deductibles, cost sharing, HSA eligibility, formulary ID, and other plan attributes

All other files like network coverage area, service area, and business rules tables mainly describe issuer-level data. Issuer here means insurance company available in the health insurance marketplace. Since the main goal of this project is to explore the landscape of each individual plan and predict rates for each plan, all issuer-level data was omitted from this study

The rate.csv contains plan-level data on individual rates based on an eligible subscriber's age and has over 12 million data entries. The BCS table contains plan-level data on essential health benefits, coverage limits, cost sharing (copay and coinsurance), and so on. The plan attributes table, as the name suggests, contains plan-level data that describes attributes of each individual plans, such as maximum out of pocket payments, deductibles, number of wellness program offered, and 174 other variables. Not all variables were used in this study, so I must manually select key variables based on our understanding of the healthcare industry.

Problem Statement:

There are number of plans available in the marketplace which are having different benefits and pricing for different states. It's a nightmare for the member to choose and buy the right kind of plan which can serve more benefits with lesser premium. So, my study will help in making the decision to choose the right plan for the individual and predict the plan rate in future.

During exploratory analysis, I will try to answer how do plan rates and benefits varies across states, how do plan benefits relate to plan rates, how do plan rates relate to age, how do plan vary across insurance network providers?

Data Wrangling:

Data in the healthcare domain are notorious for its dirtiness and untidiness. As Hadley Wickham has pointed out in his paper, "tidy data", it is essential to perform data wrangling to get it ready for meaningful analysis. Therefore, one of the goals for this study is to clean this enormous dirty and untidy dataset. There are many missing values, conflicting data entries, and inconsistent data types.

I have merged the data for different years and different types of data in one data frame, so that I can analyze all the data with the single source. It will require data merging and reshaping techniques (Split-Apply-Combine).

There are few data columns which were not captured from the year 2017 onwards which make me to read the manual so that I can identify all the common Columns in all the years which helps me to merge them together.

I have read all the 3 files from the website for all the year 2014-2019 and merge them separately. The major problem with he files like Plan rates and Benefits and Cost sharing, is they are very huge and takes time to process. So, I have kept them separately and wrangle them individually.

In all files I must deal with all kind of Data cleaning process and Imputing methodologies to make it clean enough to do Visualization and Analysis. The common to deal with are:

- **Missing data (NaN and Spaces)** :

  To handle them, it was a logical imputation rather than just deleting them.
  There are numerous ways that you can identify whether the dataset has Missing data or not.
  I have used the below way to do it:
  1. DataFrame.info() will give you an idea how many NaNs are present in the dataframe.
  2. After that, I have used .isna() fucntion to get the data of all the missing values in the dataframe

  To fill the data with the logical data I have used the below methods:

  1. *fillna()* method: There are lot of options available with the fillna() method to deal with the missing values.

     For example: *fillna(0)* will replace all the NaN values with 0(zero) 2.

  2. *dropna()* method: to drop the entire data point if there are missing values in all the columns.

- **Outliers :**
  For such kind of data, I majorly deleted them as they can interfere with the Algorithm and can mislead the prediction algorithm. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results.
- **Duplicate Rows:**
  Removing duplicates is necessary as it can lead the model to overfit and will work badly on the unseen data

There are columns in the data frames which has floating numbers representing the amount with “$” sign as prefix. For my analysis I had to change the column as numeric so that I can perform mathematical calculation on the columns. To convert these kinds of values to numerical values I must get rid of the “$” sign.

To do this I have used replace function on data frame column name and then pandas *to_numeric* function to convert the object datatype into numeric datatype.

Similar kind of conversion I did on Age column of the rate data frame, which has the values as object datatype and with the text values as “0-20” where I have to get rid of “-“ and covert it to numerical values.

Also, *RatingAreaId* column has the similar values, like “Rating Area 10”, in these types of values I must get rid of space and then convert these areas into numbers. As these data will require in doing prediction of rates so I need to convert them in numeric quantity.

Exploratory Analysis

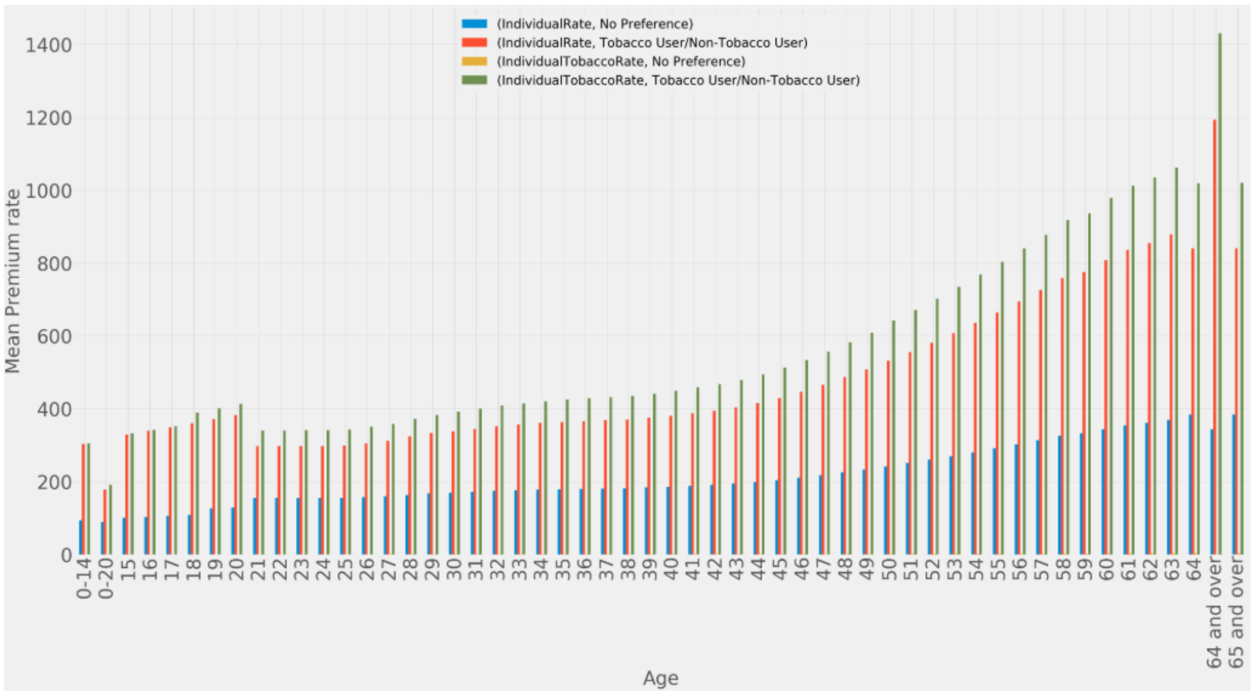
Below are few analyses that was done to understand the behavior of data points.

1. **Rate vs Age vs Tobacco**:

   Here I study he behavior of Plan rates with respect to the age for Tobacco/Non-Tobacco users.

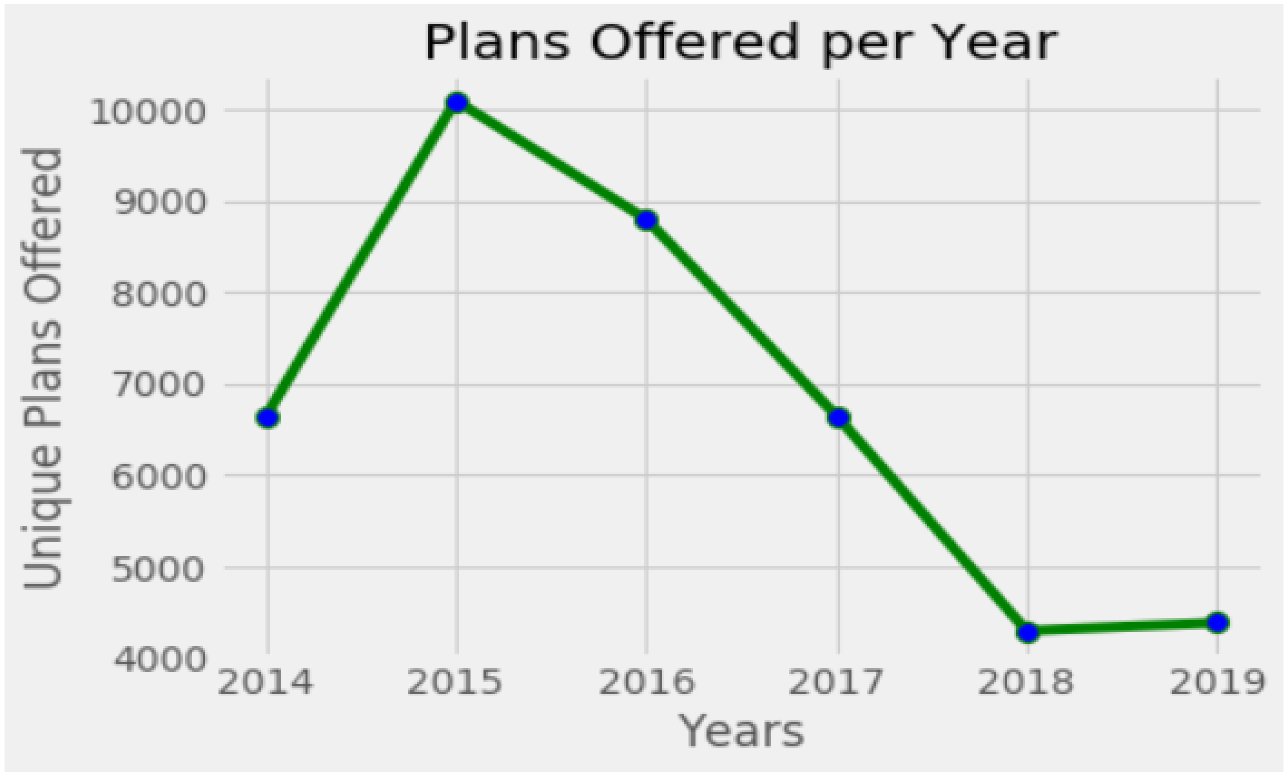
   The grey line is almost the double of the red line in the below graph. It suggests that **as age increases the rate difference will be higher between Tobacco user Individual rate and Tobacco user Individual tobacco rate.** The premium rate for Tobacco user is more than double of Non Tobacco user. This is a good incentive for living a healthy life.

   This clearly suggests that these 2 populations are different and could be an outlier for each other. While applying Machine learning algorithm we have to keep this in consideration.



2. **How are plans being offered in the subsequent years ?**

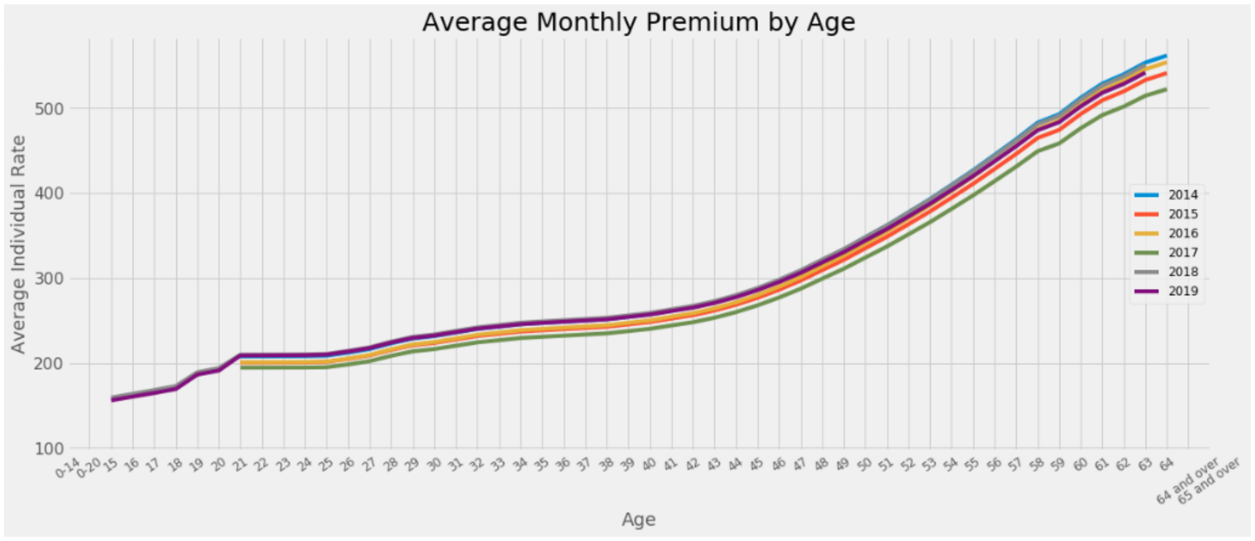
There was a huge change in the number of plans offered in the marketplace from the time of its inception. The below graph suggests the same story.



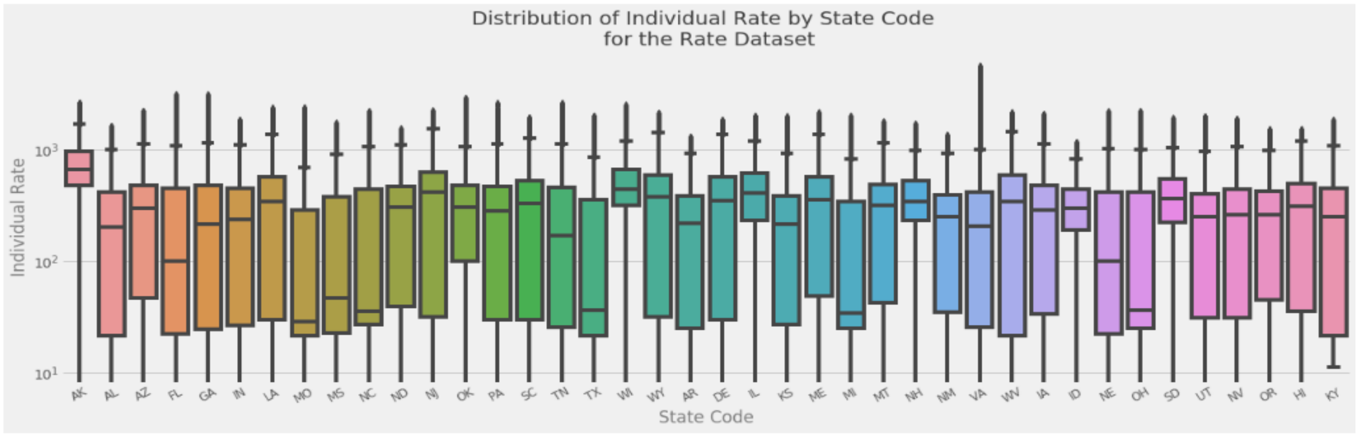
3. <u>Individual rates variations with Ages in all years</u>

In the year 2014 to 2017, there was one group of 0-20 age. But from 2018 the age group is reduced to 0-14 and Kids older than 14 are covered differently. Also we can see that upper limit of age is changed from '64 and Above' to '65 and above' after 2018.

Average premium mostly remains same throughout the years. Premium increases with increase in Age, which is normal in health Insurance industry.



4. **Distribution of individual rate across all the states**.



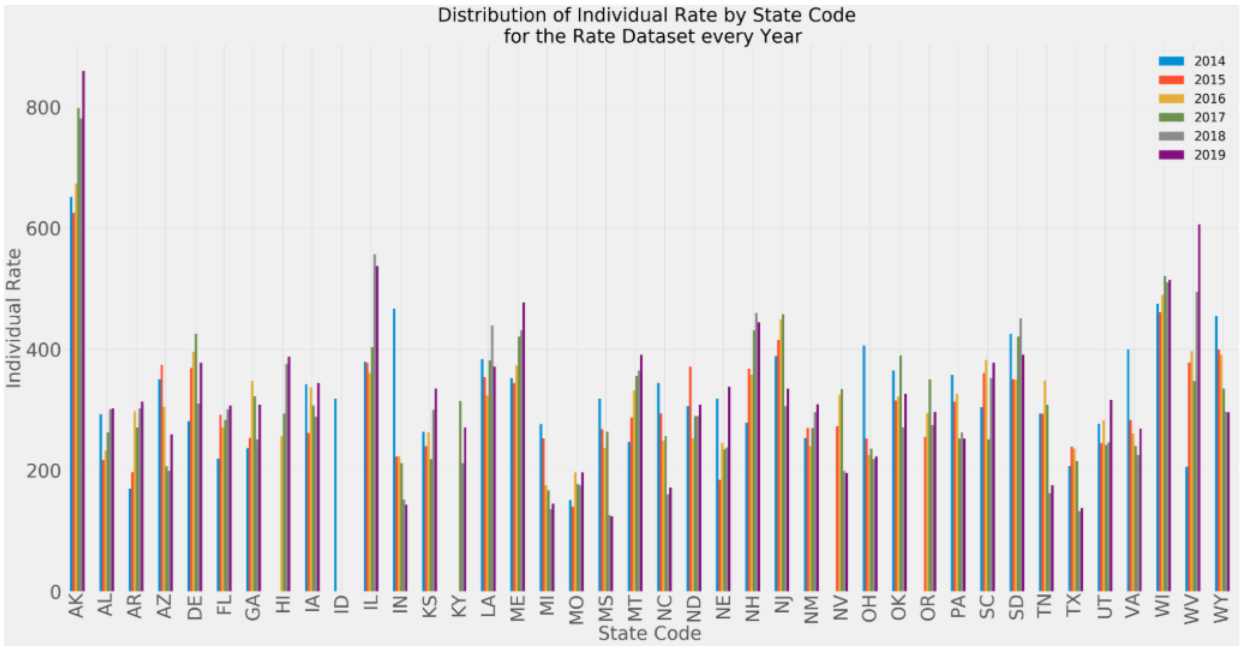
Observations from the above graph:

a) Out of 50 states only 40 states have participated in this exchange program
b) There are big differences in insurance premiums among the states. It's clear that the individual rates in Alaska and Illinois are very high.
c) The Median of Alaska is also very high, which means that there are no Plans which comes cheaper in this state
d) Tennessee and Texas are the more reasonable states with regards to Individual Plans.

5. **Mean Individual rate per state code per year :**

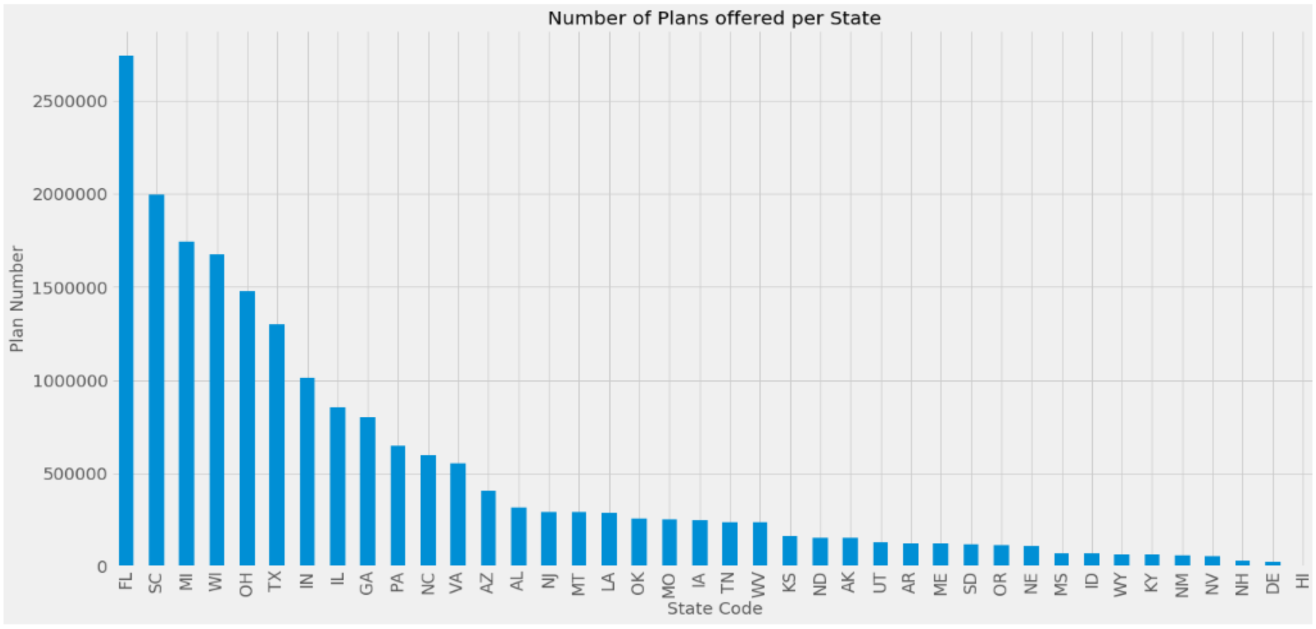
It also tell the same story

a) Alaska is always a very expensive state.
b) Illinois is again expensive as compare to other states
c) Wyoming is becoming expensive in 2019.



6. **Number of Plans per state**

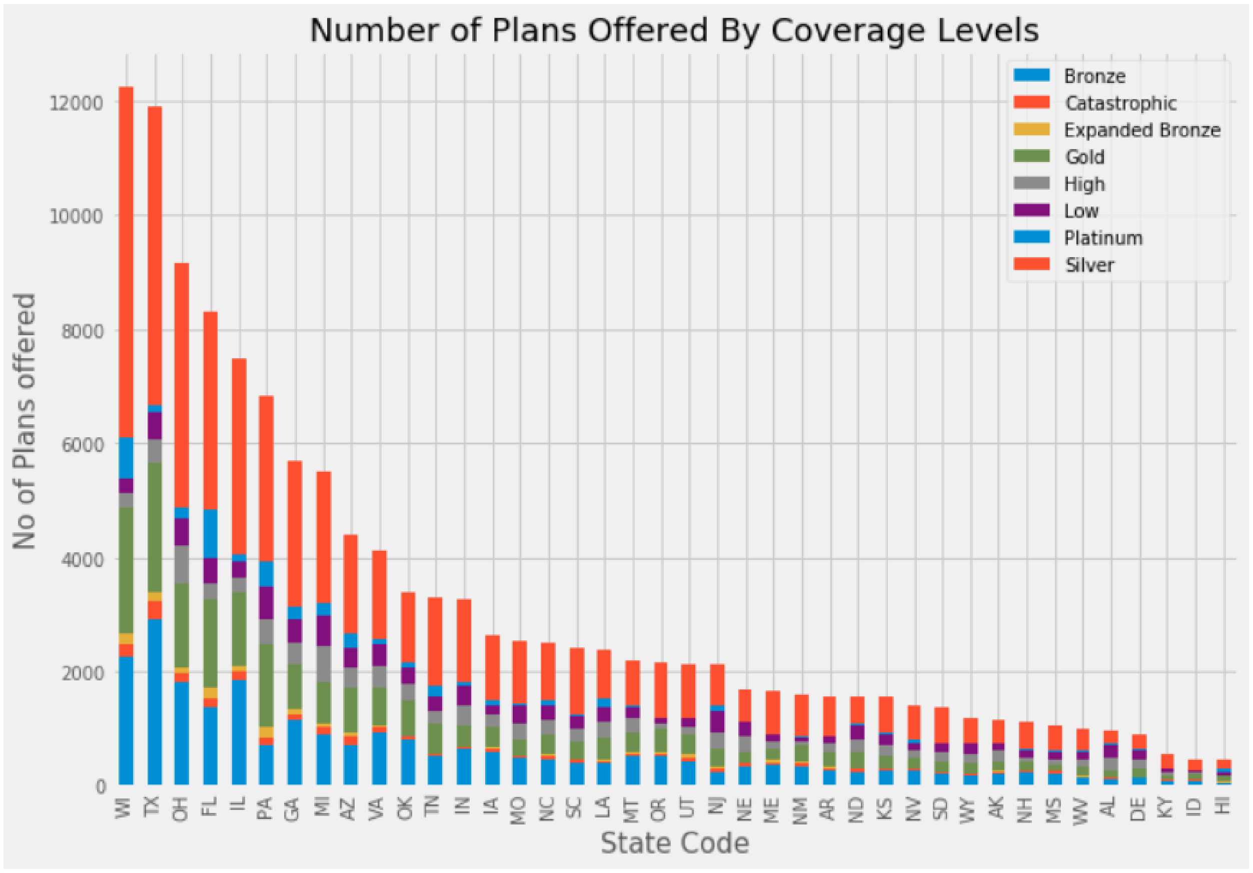
The above analysis allows us to think why few states are so expensive than others. Lets check the below graph to answer it. I have checked the number of plans offered in each year, and it gives us a very interesting story. There is huge difference among the states in plan offerings.



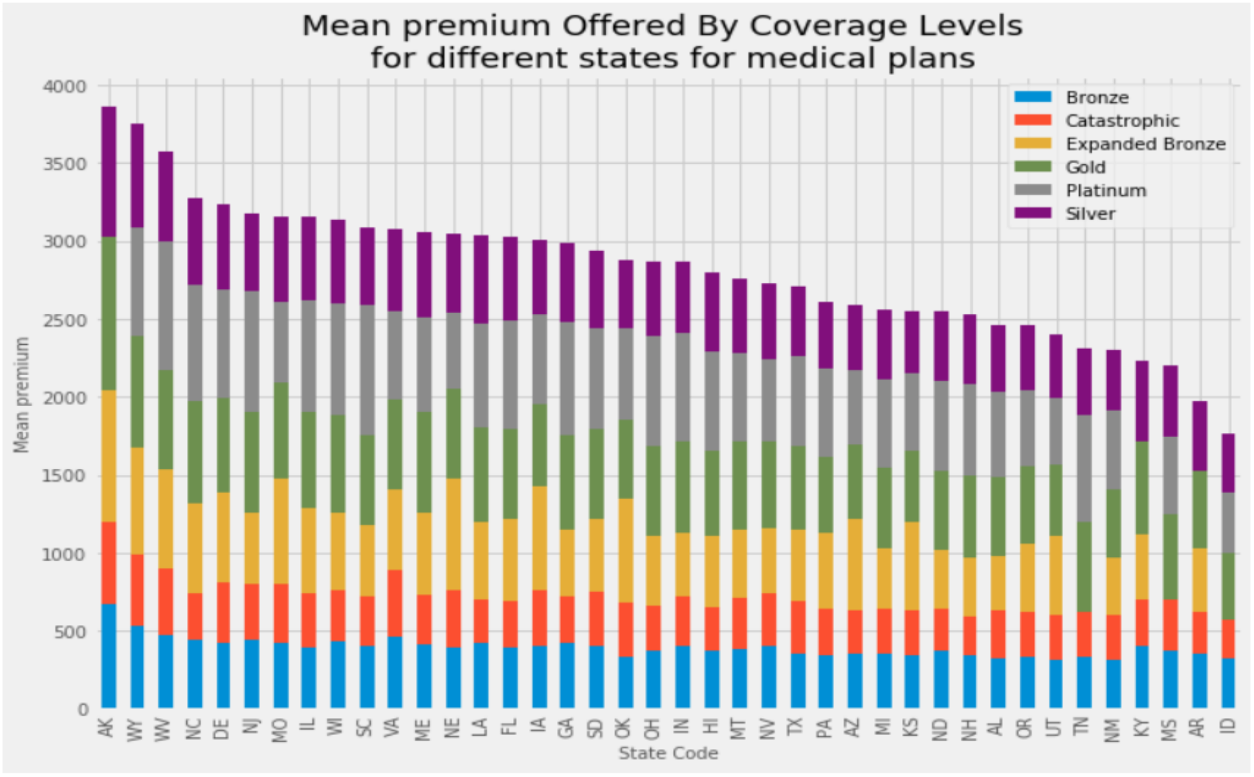
a) Florida has maximum numbers of plans offerings for the population
b) Alaska, Illinois and Wyoming have very less offerings for the plan.

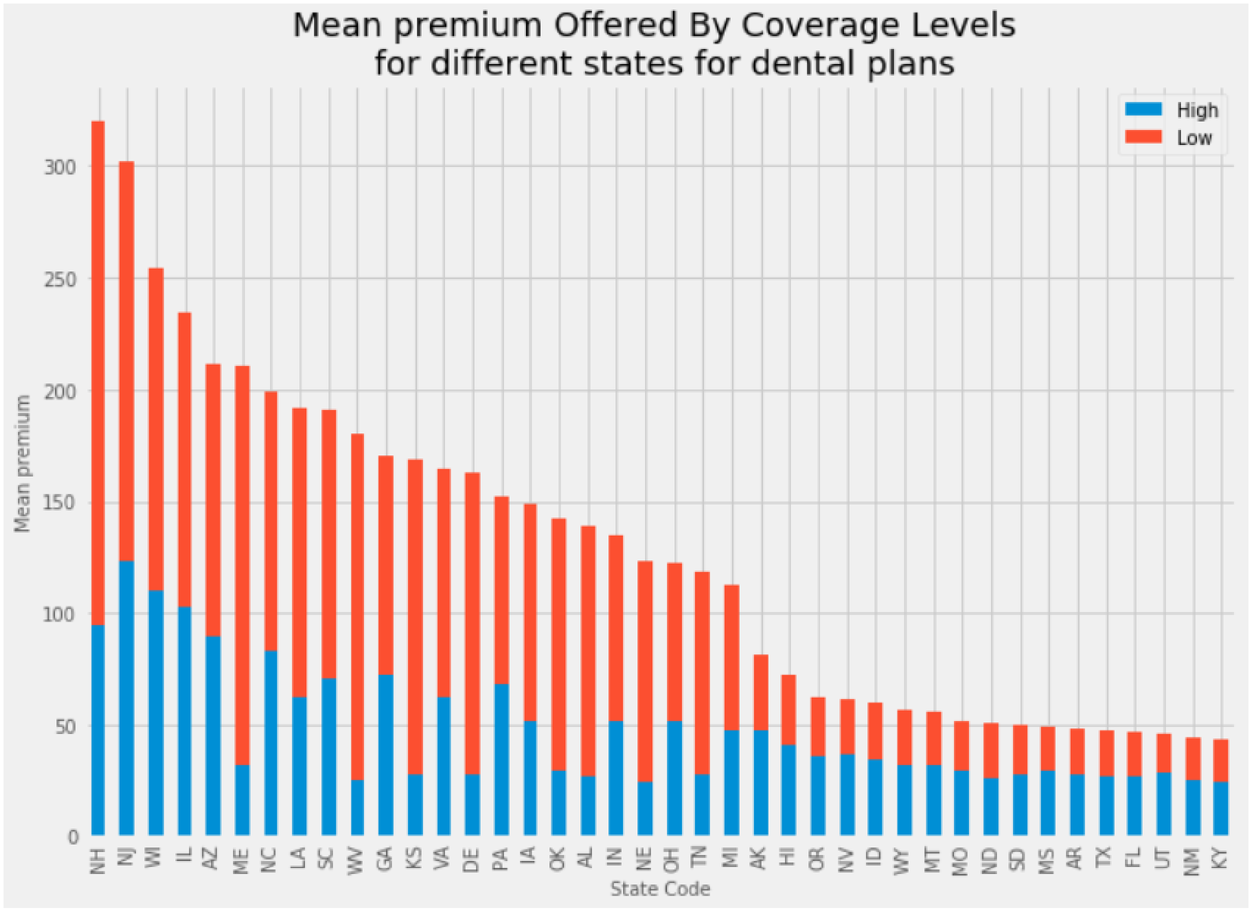
c) We can see some negative co-relation with Number of plans offered with monthly premium in a State

7. **Number of Plans offered by coverage level**
There is a difference in offerings of the plans in different states but the ratio of plans in different categories remains same. As per the below graph, all states offer Bronze category plans the most. Bronze covers all the mandatory benefits as suggested by the ACA act but doesn't provide whole lot of benefits. So, it is a cheap plan with all but few benefits.



8. **Mean premium offered by Coverage Levels per state**
Now I have combined the plan and rate files together based on Plan ID, state code and Business year. We have the indicator in the plan file which suggests whether the plan is Dental or Medical. So, I have analyzed them differently to see how their rate distribution is among all the states.

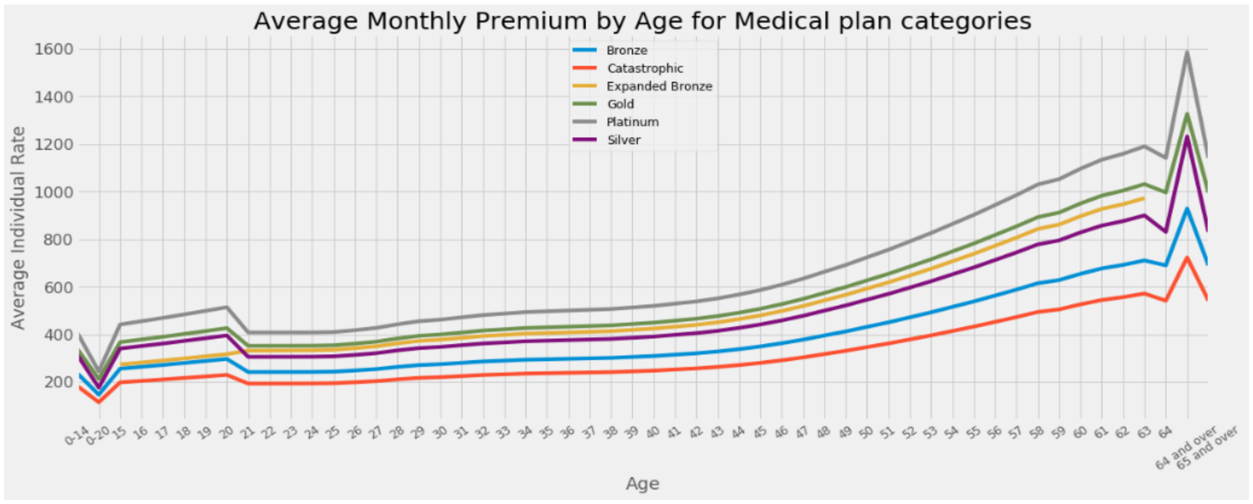

Mean premium Offered By Coverage Levels
for different states for medical plans
Mean premium
State Code
Bronze
Catastrophic
Expanded Bronze
Gold
Platinum
Silver
0
500
1000
1500
2000
2500
3000
3500
4000
AK WY WV NC DE NJ MO IL WI SC VA ME NE LA FL IA GA SD OK OH IN HI MT NV TX PA AZ MI KS ND NH AL OR UT TN NM KY MS AR ID

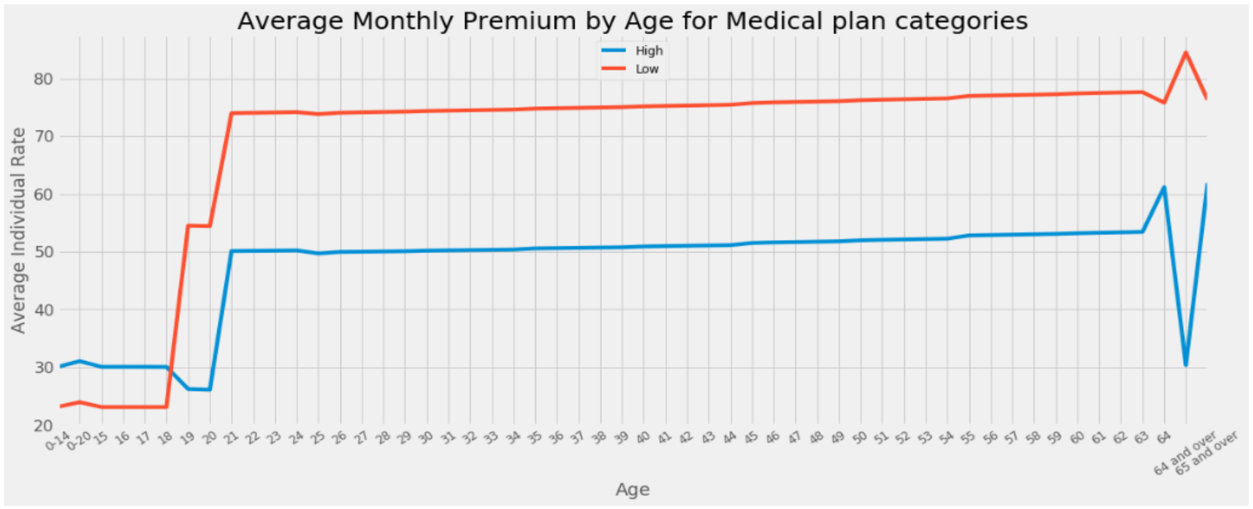

Mean premium Offered By Coverage Levels
for different states for dental plans
Mean premium
State Code
High
Low
0
50
100
150
200
250
300
NH NJ WI IL AZ ME NC LA SC WV GA KS VA DE PA IA OK AL IN NE OH TN MI AK HI OR NV ID WY MT MO ND SD MS AR TX FL UT NM KY

We can see that maximum premium is charged by Alaska in Medical plan and Cheapest plans are offered by Idaho
In Dental plan NH seems to be charging way too high than other states.

9. **Variation of Monthly premium by Age**
We have already checked how plan rates vary by age for different years. Now we will analyze how rates vary by age for different plan categories.

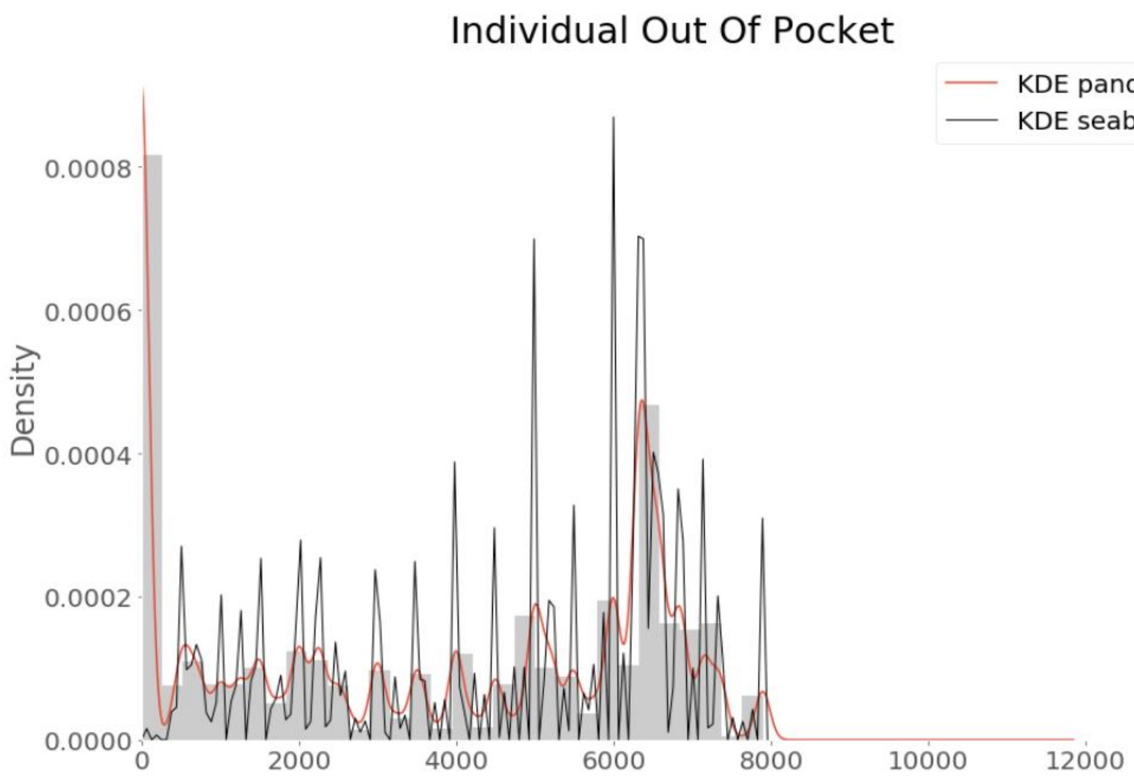
a) As expected, Platinum plan category is expensive
b) Bronze plan category is not the least expensive. Catastrophic plans are for specific needs only so they are cheaper but covers few benefits only
c) The variations at the ends of both graphs are due to the policy changes in the year 2017. Age limit for Old enrollees is changed from 64 to 65. Policies for children is reduced from 20 to 15.
d) Delta Plans has very little variations with Age after 20 years. Category 'Low' is quite double the category 'High'





10. Out of Pocket analysis

For any plan OutOfPocket is the amount which everyone should care about. Apart from Monthly Premium, population must fetch out extra amount for every Service he/she takes. Let's see how they are different for In Network and Out Network. I am going to investigate EHB benefits for all the plans. I have used KDE to analyze it.



As the KDE chart shows, lots of plan limits are 0, that means these plans are for the poor people who need help. And we can find that most limits are more than 12000, that means the U.S. insurance insurers like focusing on high-end plan products. Maybe that's why Americans usually spend loads of money on their health insurance.

11. Total Plans offered and how many states have participated

Unique plan offered to Americans:	**981**
Number of states participated in the Exchange program:	**40**

Over the period of 6 years, American were offered 981 unique benefits. Interestingly not all the 50 states have participated in the marketplace or Exchange. For example California, New york. States who are not participating in the federal run marketplace they have their own state-run exchange. There are only 39 states who are using federal run health insurance marketplace. Idaho moved to state run marketplace after the first year of enrollment. And state-run marketplace of Kentucky was dismantled in year 2016 and joined federal run exchange. The above number 40 is the number of states who were/are associated with federal run exchange since its inception.

12. **Overall Summary of Plans**

List of benefit type topping the chart in a given business year is given below. There is no surprise that top of the benefits offered are of dental as these are mandatory for children and its also come with Medical benefits.

All Plans benefits overview

BusinessYear	Benefits			
	count	unique	top	freq
2014	1164869	496	Orthodontia -Adult	18719
2015	2079286	517	Orthodontia -Adult	31269
2016	1774255	421	Orthodontia -Adult	26997
2017	1324275	281	Orthodontia -Child	21371
2018	829652	252	Major Dental Care- Adult	13857
2019	967050	244	Basic Dental Care -Child	15695

Let's check how the statistics change after dividing them into dental and Medical data.

Medical Plans benefits overview

BusinessYear	Benefits			
	count	unique	top	freq
2014	1134699	339	Mental/Behavioral Health Outpatient Services	15247
2015	2038625	387	Home Health Care Services	26991
2016	1737894	317	Orthodontia - Adult	23138
2017	1298219	228	Accidental Dental	18568
2018	808437	191	Mental/Behavioral Health Outpatient Services	11565
2019	947765	193	Routine Dental Services (Adult)	13618

Dental Plans benefits overview

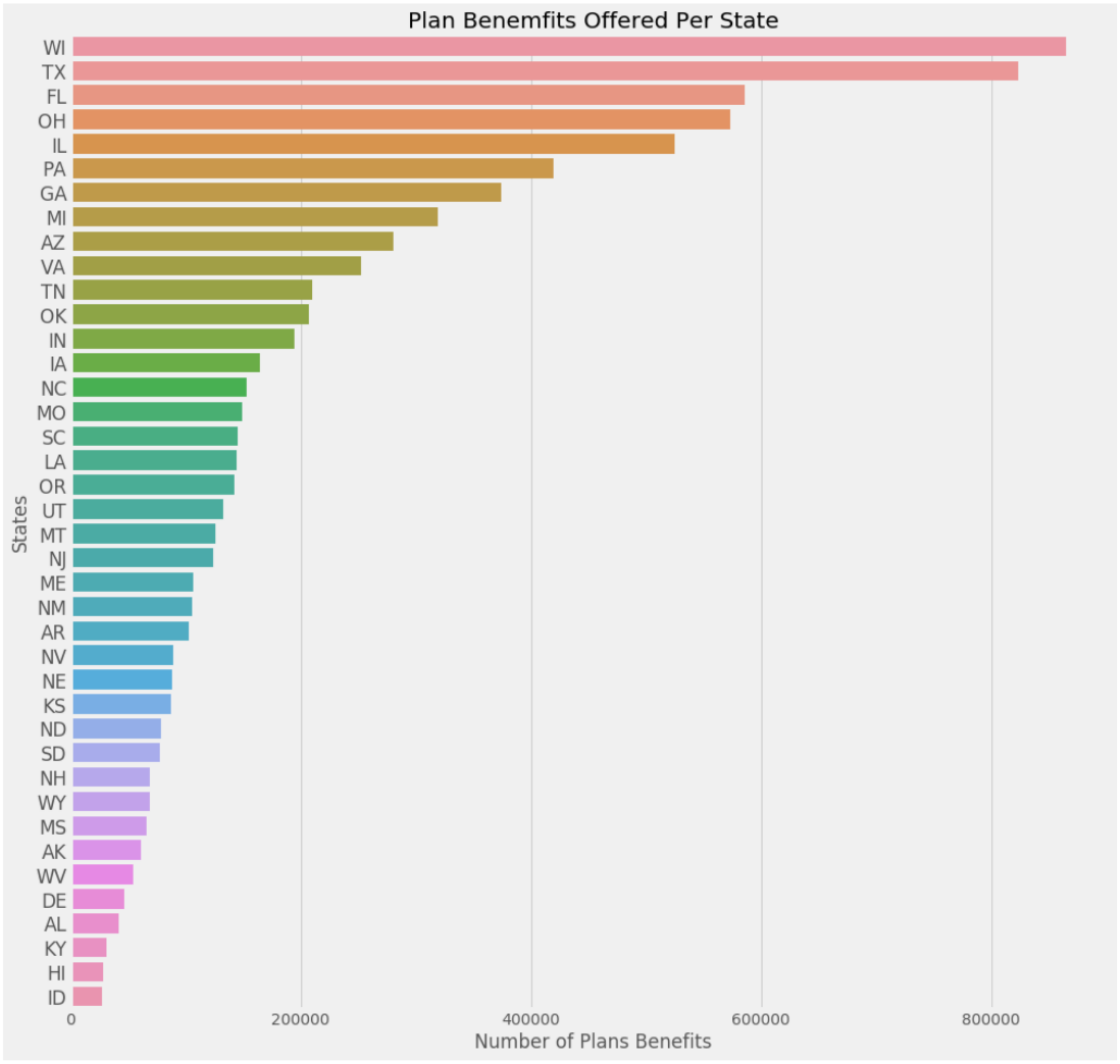
BusinessYear	Benefits			
	count	unique	top	freq
2014	34508	172	Basic Dental Care - Child	3532
2015	40661	141	Orthodontia - Adult	4278
2016	36361	164	Orthodontia - Adult	3859
2017	26056	62	Basic Dental Care - Child	2803
2018	21215	70	Basic Dental Care - Child	2292
2019	19285	60	Basic Dental Care - Child	2077

a) Few of the benefits are still in the Medical Benefit plan as they are the plans which give both Dental as well as Medical benefits
b) Number of Medical plans decreases from the year of inception.
c) Maximum number of plans were offered in 2015 but by 2019 it was reduced to 193 from 387, almost 50% reduction.
d) Same reduction is in Dental care as well.

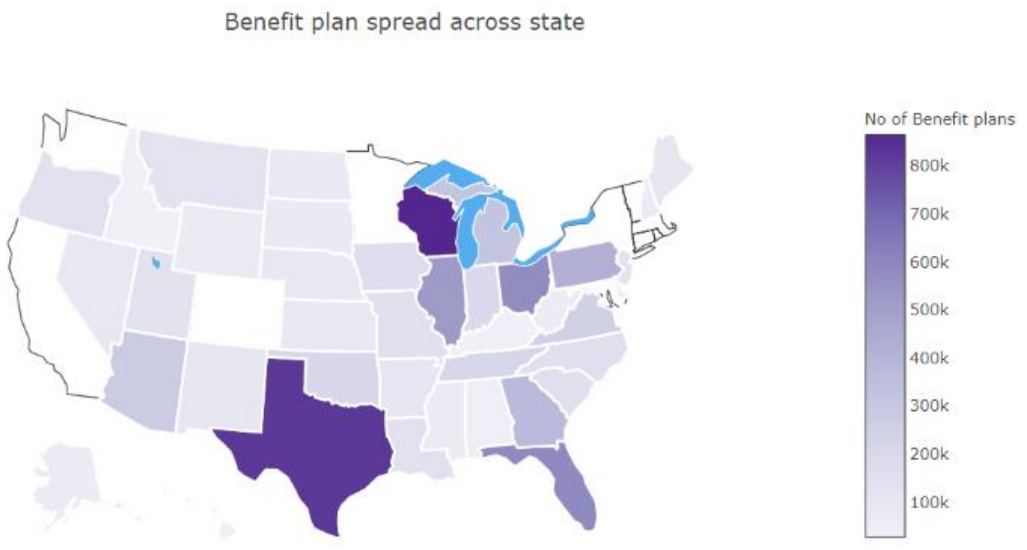
e) This is not a good sign of the marketplace!!

13. **Number of Plan Benefits offered in each state**

Benefits covered in the plan is the major factor which drives the cost of the plan. More the benefits covered, expensive the plan will be.



So, we can say that Wisconsin (WI) offers most number of benefit plans to the population and Texas is second. This can also be viewed on the America's map below.



Summary of Analysis

In the whole analysis I have tried to figure out how Plans, benefits and rates are related to each other and how they are spread among all the states.

There are 40 states who have participated in the Exchange program and California is the biggest state which has not participated in any of these years. In the coming year if California participated in the Exchange programs then it will be a huge opportunity for the companies, as California has maximum population to target

As the exchange program getting older, number of plans and benefits are decreasing. This is a huge concern for ACA initiative.

Wisconsin gives maximum number of benefits to choose from and has reasonable Monthly premium. It suggests that Wisconsin state is a huge success for ACA initiative.

Alaska offers minimum number of Benefits and premium is super expensive as compared to other States. It might be due to its harsh conditions and small population to deal with.

Huge premium will need to be paid if a user is Smoker. It is a good penalty imposed on people who are living Unhealthy life.

Machine Learning:

Introduction:

Now, we will build the models to predict the individual premium rates using regression models. We will compare the models results and check which gives the best optimum solution. There are few challenges that we have faced in running the machine learning code in the single machine. I will try to resolve them as well as we decipher the best model for prediction.

We have researched the data on all states of US. The total files volume is more than 10 GB which increases exponentially if we do one hot encoding on the dataset. So, to avoid getting into memory run out issues we will concentrate on Florida's data. As per our analysis Florida has the maximum number of Plans and it is a good representative sample of the whole US coverage.

We did decrease the volume a lot by above solution, even then we can land up in memory issue as the personal system is generally not RAM/Cores heavy. So, to run algorithms I have rented the cloud cluster (Microsoft Azure) with 8 cores and 64GB RAM which helped me to run the extensive Random and Grid Search.

Data Preparation

Unfortunately, we aren't quite at the point where you can just feed raw data into a model and have it return an answer (although people are working on this)! We will need to do some minor modification to put our data into machine-understandable terms.

The exact steps for preparation of the data will depend on the model used and the data gathered, but some amount of data manipulation will be required for any machine learning application.

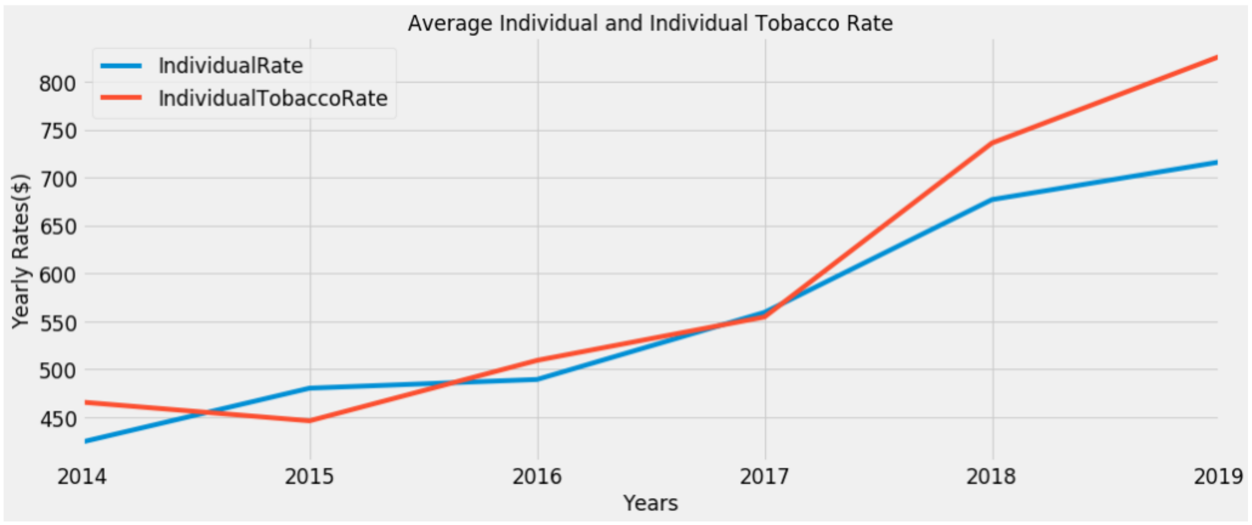
At first, we must merge the Rates and Plan Attributes datasets as we Plan Attribute has the variables which can define the type of plan. It helps to distinguish the Dental Plans and Medical plans. It also has the attribute which aids us to get the plan category which could be the major attribute for modelling. Then we must merge the modified rates dataset to the benefit dataset as the number of benefits offered in the plan is major factor which drives the premium rate. Benefits dataset is huge and cannot be merged with rates dataset in the single machine with limited resources, so to mitigate it I have used the count of benefits offered for the Plan.

I have also changed the type of the attributes to save the space issue. I have changed '*int64'* to '*int32'*, '*float64'* to '*float32'* and changed all object/categorical data type to 'category' data. This methodology saves lot of space and helps quicken the process.
The next process I have checked the distribution of the premium rates. Below is the snapshot of the summary of premium rates. The premium rates are different for Tobacco user and Non-Tobacco users. The total premium to be paid will be the sum of both the values. So, for prediction we have added them together.

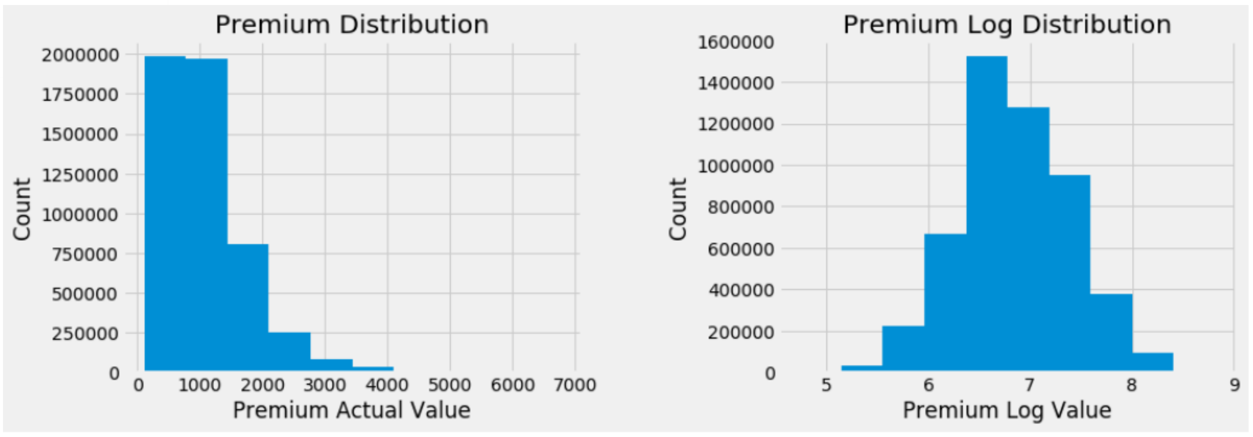
Premium Rate Summary		
Summary Parameters	**IndividualRate**	**IndividualTobaccoRate**
count	5155187	5155187
mean	537.4643	563.5067
std	289.3101	395.2735
min	51.99	0
25%	332.69	329.71
50%	453.61	480.57
75%	671.65	753.94
max	3072.06	3686.49

Average Premium Rate Per Year		
BusinessYear	**Average IndividualRate**	**Average IndividualTobaccoRate**
2014	424.254497	465.777226
2015	480.328384	446.30746
2016	489.392785	509.28896
2017	559.321436	554.570253
2018	676.881823	735.876146
2019	716.075636	826.059768

We can see below that the individual rates have been increasing since its inception. The increase in rate is also getting bigger every year. But the same is not true for the Individual Tobacco Rate. The second year of exchange saw a fall in individual tobacco rate but again started increasing every year and last year the increase in rate is very large.



From the below graph we can see that total Individual rate is right skewed. Most of the algorithms doesn’t work the best way if they are not normally distributed. So, to make it normal we must take a log of it.



You can see that log of premium seems normal as compared to premium actual values.

Then the next step was to do the One Hot Encoding for on my categorical featured like '*MetalLevel*'. This process takes categorical variables and converts it to a numerical representation without an arbitrary ordering. I have used '*pd.getdummies*' to get the One Hot Encoding.

Below is the snapshot of data after one-hot encoding. Here we can see that '*MetalLevel_Silver'* is 1 and rest of them are 0.

MetalLevel_Catastrophic	MetalLevel_Expanded Bronze	MetalLevel_Gold	MetalLevel_Platinum	MetalLevel_Silver
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1

As we can see that we have split the data into 2 parts, one '*Test*' dataset and the other '*Train*' dataset. During training, we let the model 'see' the answers, in our case the monthly premium, so it can learn how to predict the premium from the features. We expect there to be some relationship between all the features and the target value, and the model's job is to learn this relationship during training. Then, when it comes time to evaluate the model, we ask it to make predictions on a testing set where it only has access to the features (not the answers)! Because we do have the actual answers for the test set, we can compare these predictions to the true value to judge how accurate the model is.
We have split the dataset according to the years. In the training set I have the data from the years '2014' to '2018' and in the 'Hold-Out' Test dataset I have data from the year '2019'. The split is

logical as we must predict the upcoming year premiums so we will always have single year data only for prediction.

Baseline Prediction

Now we must establish a baseline a sensible measure that we hope to beat with our model. If our model cannot improve upon the baseline, then it will be a failure and we should try a different model or admit that machine learning is not right for our problem. The baseline prediction for our case can be the Monthly Premium averages. In other words, our baseline is the error we would get if we simply predicted the average Monthly Premium for all plans.

Predicting the average of Monthly Premium has given a very bad performance on 2019 data. Negative R-Square value means that this is the worst guess that we can have for the prediction. It cannot explain any variance with this assumption or guess.

Model Training Strategy

After all the work of data preparation, we must fit our training data in the different models starting from simple linear models to complex ensemble model. During this exploration we will test these models on the testing data and compare them on different parameters.

The strategy to compare the model is to check the R-square, Mean Square Error, Mean Absolute Error and Accuracy. After that we will also see how the optimization can be done with Hyperparameters so that we don't lose much on prediction values and training can be done with minimum computational size.

Train Model: OLS

In statistics, ordinary least squares (OLS) or linear least squares is a method for estimating the unknown parameters in a linear regression model. OLS chooses the parameters of a linear function of a set of explanatory variables by minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being predicted) in the given dataset and those predicted by the linear function.
Geometrically this is seen as the sum of the squared distances, parallel to the axis of the dependent variable, between each data point in the set and the corresponding point on the

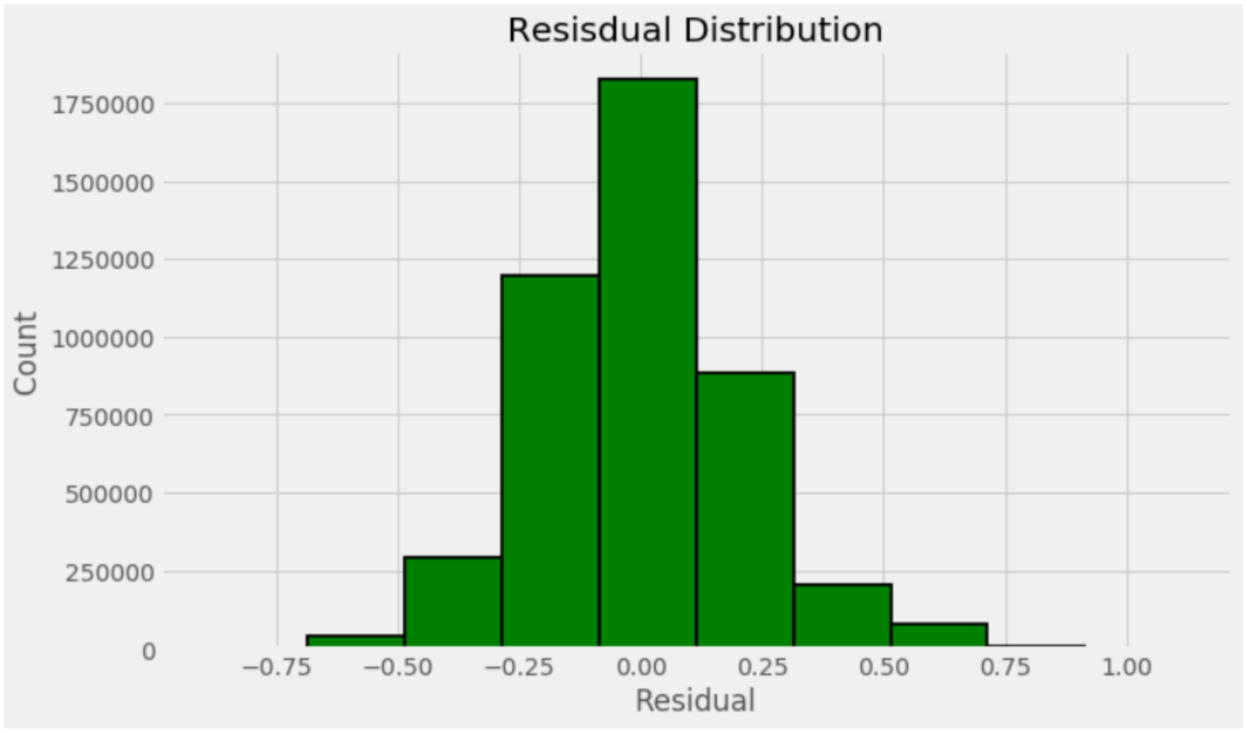
regression line – the smaller the differences, the better the model fits the data. The resulting estimator can be expressed by a simple formula, especially in the case of a single regressor on the right-hand side. The OLS estimator is consistent when the regressors are exogenous, and optimal in the class of linear unbiased estimators when the errors are homoscedastic and serially uncorrelated. Under these conditions, the method of OLS provides minimum-variance mean-unbiased estimation when the errors have finite variances. Under the additional assumption that the errors are normally distributed, OLS is the maximum likelihood estimator.

We have implemented the OLS method to predict the individual monthly rate. We got below summary on the OLS regression model.

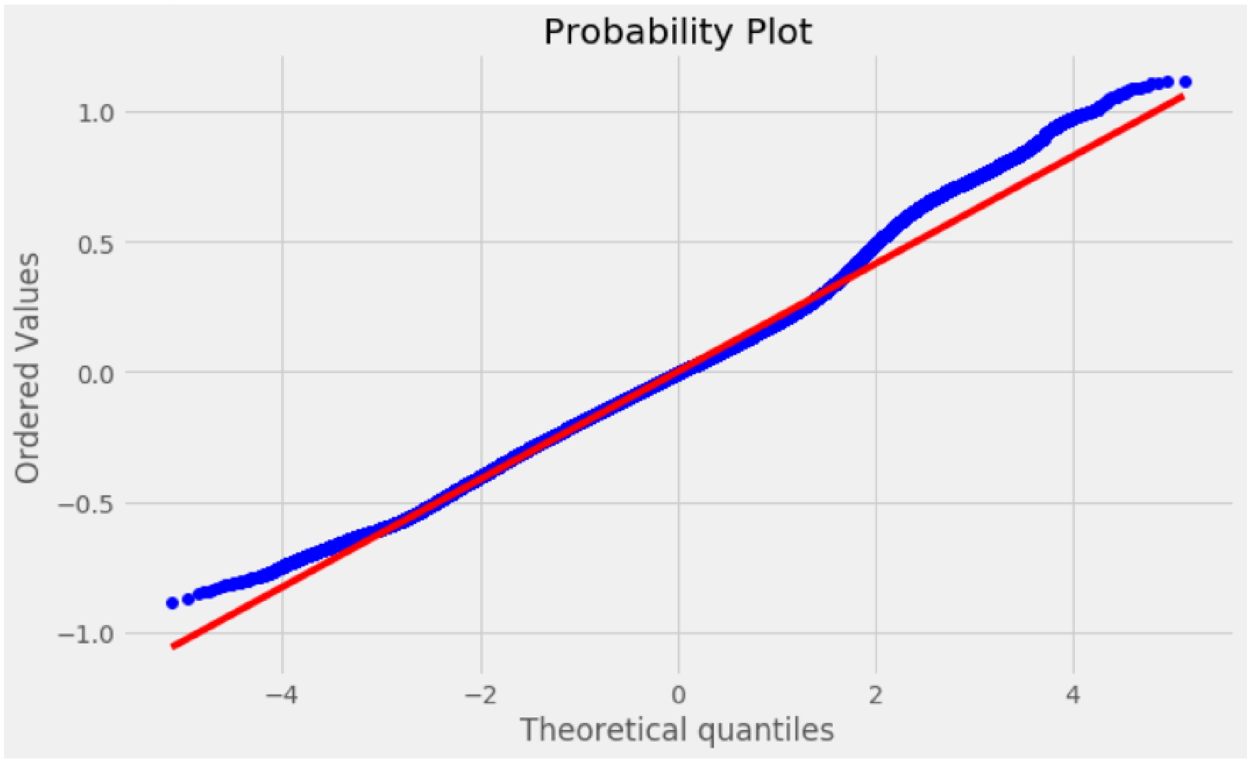
OLS Summary Result 1							
Dep. Variable:	**Model:**	**Method:**	**Date:**	**Time:**	**No. Observations:**	**Df Residuals:**	**Df Model:**
IndividualRateTotallog	OLS	Least Squares	Mon, 25 Feb 2019	22:08:05	4531151	4531136	14
R-squared:	**Adj. R-squared:**	**F-statistic:**	**Prob (F-statistic):**	**Log-Likelihood:**	**AIC:**	**BIC:**	**Covariance Type:**
0.848	0.848	1812000	0	682010	-1364000	-1364000	nonrobust

OLS Summary Results 2						
Predictors\Statistics	**coef**	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Age	0.0286	7.26E-06	3935.914	0	0.029	0.029
BusinessYear	0.124	7.37E-05	1682.769	0	0.124	0.124
RatingAreaId	0.0004	4.78E-06	74.449	0	0	0
Duration	0.0005	1.74E-06	306.623	0	0.001	0.001
TEHBInnTier1IndividualMOOP	-1.13E-05	4.51E-08	-249.599	0	-1.13E-05	-1.12E-05
TEHBOutOfNetIndividualMOOP	8.69E-06	1.42E-08	612.855	0	8.66E-06	8.72E-06
BenefitName	-1.34E-05	1.97E-07	-67.828	0	-1.38E-05	-1.30E-05
Tobacco_No Preference	-105.142	0.064	-1652.37	0	-105.267	-105.017
Tobacco_Tobacco User/Non-Tobacco User	-104.582	0.064	-1643.81	0	-104.707	-104.457
MetalLevel_Bronze	-35.1198	0.021	-1657.7	0	-35.161	-35.078
MetalLevel_Catastrophic	-35.3238	0.021	-1668.99	0	-35.365	-35.282
MetalLevel_Expanded Bronze	-35.0937	0.021	-1642.78	0	-35.136	-35.052
MetalLevel_Gold	-34.7251	0.021	-1639.35	0	-34.767	-34.684
MetalLevel_Platinum	-34.5966	0.021	-1631.74	0	-34.638	-34.555
MetalLevel_Silver	-34.865	0.021	-1644.64	0	-34.907	-34.823
IsEHB_No	-104.818	0.064	-1648.45	0	-104.943	-104.694
IsEHB_Yes	-104.906	0.064	-1647.65	0	-105.03	-104.781

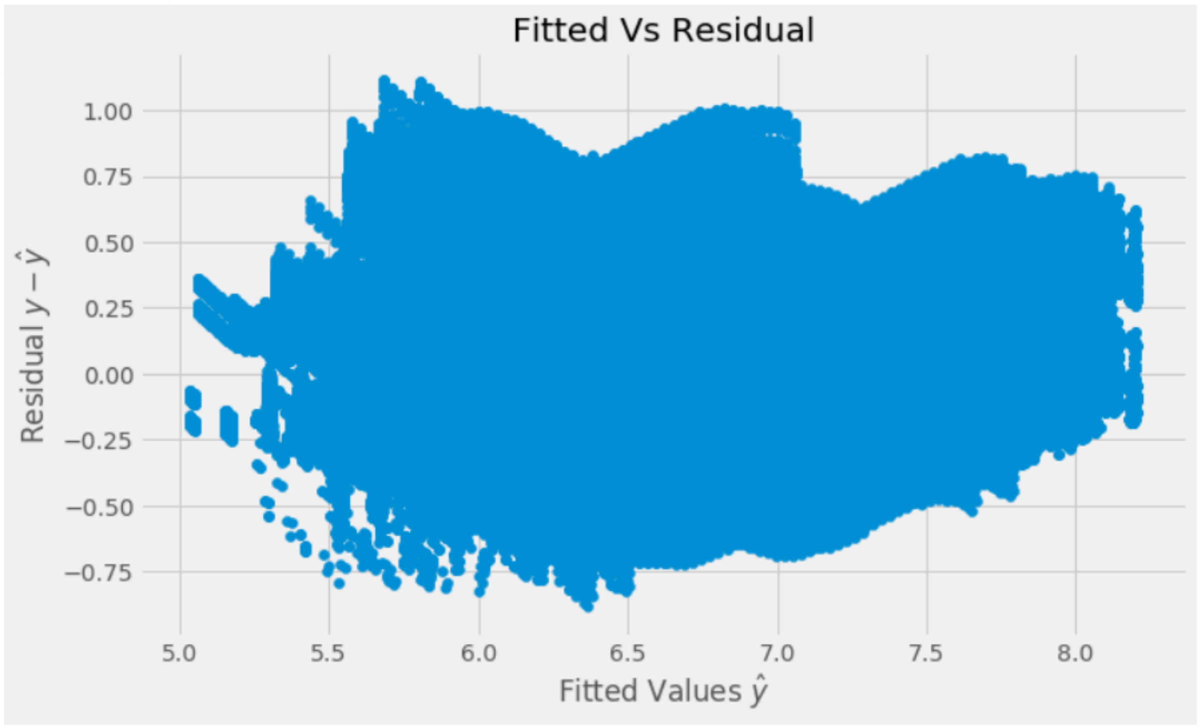
The R-Square value on the training dataset is .883 which is of course way better than our base line model. To see how well the data has been predicted via any model we can see the Residual distribution of it. If the distribution is normal, it means that model is a good prediction. Below graph is the distribution of Residuals and we can see that it is a bit right skewed. This suggests that OLS predictions are not the best for the current data.



I have also plotted the Quantile plot. In statistics, a **Q–Q (quantile-quantile) plot** is a probability plot, which is a graphical method for comparing two probability distributions by plotting their quantiles against each other. The above two plots suggest that the residual distribution is normal but with some outliers. OLS is not good in handling the outliers. This is the reason the Sci-Kit learn algorithms are very famous, as they are inherently capable of handling the outliers.



I have also plotted the Fitted vs Residual plot. It is used to detect non-linearity, unequal error variances, and outliers. Since most of the data are near 0 on y axis the residual looks linear. There is no pattern which means that errors are normally distributed. Looks like there are outliers. We do not detect any violations of the model assumption but there are outliers. It suggests the same as what we got from the last Q-Q plot.



I ran the model on our hold out test data. The result is impressive, and we got R-Square as .82 which suggests that it can explain 82% of the variance in test data. We really get a good result, but we will check the same with the linear Regression of the SciKit Learn algorithm as well.

OLS Test Run Metrics	
Mean Absolute Error	**0.166688**
Mean Square Error	**0.041974**
R square	**0.823046**

Train Model: SciKit Learn- Linrear Regressor

This is the linear model with the “SciKit Learn” package in python for data science. The principle algorithm is same as the OLS method, but it is better equipped to handle outliers and influencers. This we can see from the below results as well from the Linear Regression Model.

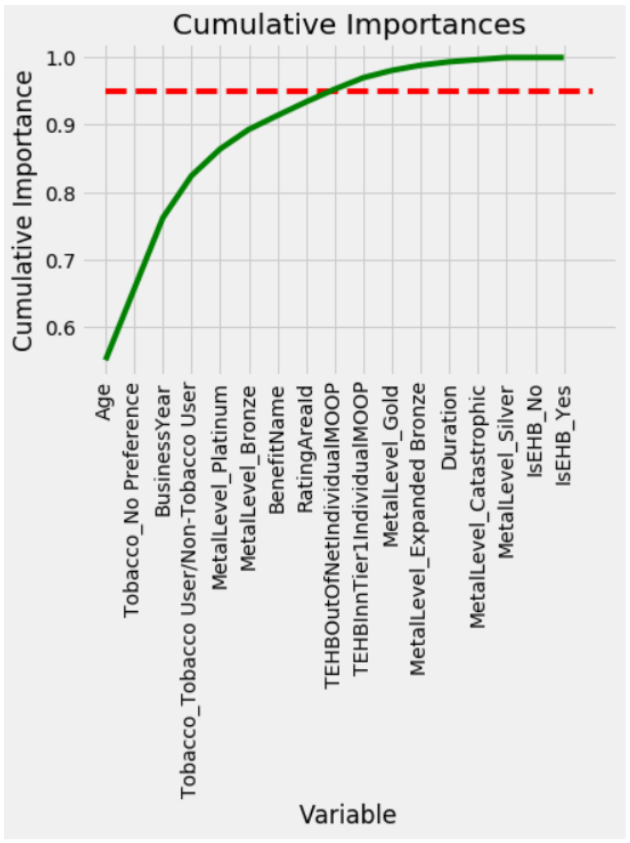
OLS Test Run Metrics	
Mean Absolute Error	**0.159717**
Mean Square Error	**0.04333**
R square	**0.848488**

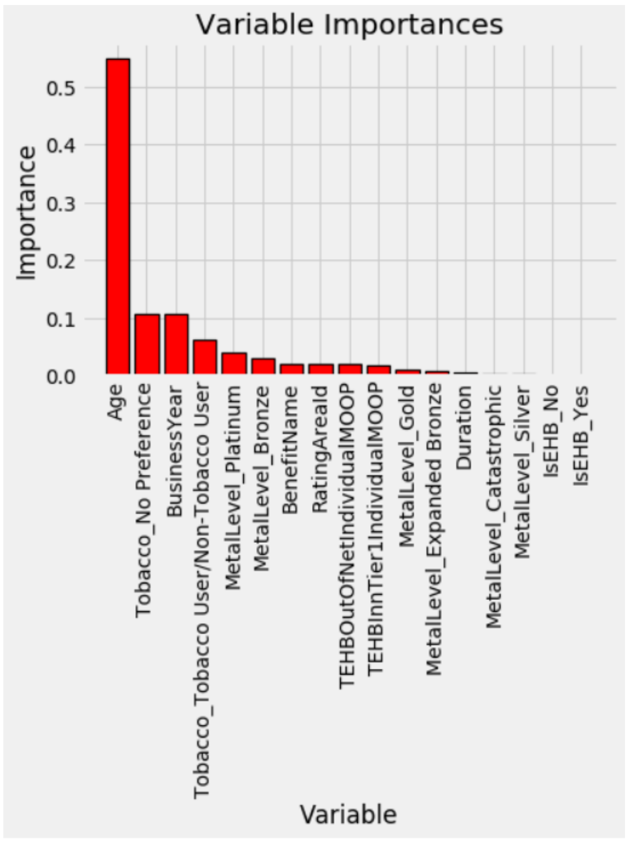
Now it can explain 85.6% variations in the test dataset which is almost 3% better that what OLS has predicted.

Train Model: Decision Tree regression

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. We have used this regression tree in our prediction model

It helped us to understand the important features available in the dataset. Below graph tells us which feature can explain most of the variance in the train dataset. Using this we can reduce the number of features for our predictions. This methodology is good in mitigating the Curse of Dimensionality. In our case we do not have many features to work on. But it is a good way to eliminate the useless features when we have 100s or 1000s of dimensions to work with as in the case of Image recognition. Cumulative graph shows that how many features can explain 95% of the variations in the train dataset. Now we can get exactly all the important features. Here only half of the features can explain the 95% of the variations. In this way we can halved out our dataset and thus saving the space as well as speeds up the process.





Then I have checked the prediction level of the Tree based model, and it came out to be 0.90 on the Test Hold-Out dataset which is again an impressive gain from OLS and Regression.

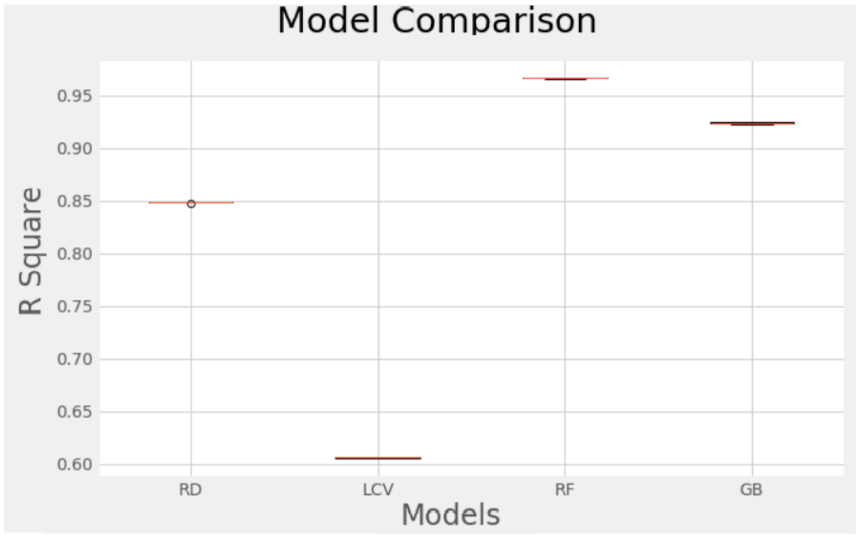
Cross validation & Model Evaluation

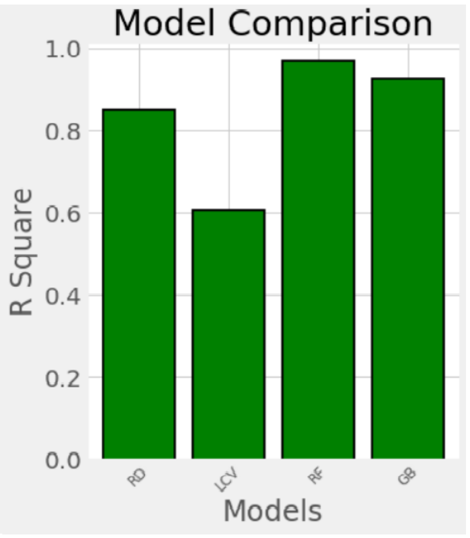
Now the strategy for getting the best model for our prediction we will use our training dataset and run it on 5-fold cross validation. Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. So, for k-fold we will have k scores and then we take average of those scores to measure the accuracy

We have implemented the 5-fold cross-validation on different regression estimators and see how well they fit with our current training data. It is good idea to understand which regressor works best for us among many regressor algorithms available in the market. For my case I have taken 4 estimators which I believed to give me the best results. We can add as many numbers of estimators as we want. The Estimators I have used are *'Ridge','LassoCV', 'RandomForestRegressor'* and *'GradientBoostingRegressor'*. Below are the result from the Cross-Validation run.

Model Evaluation result by K-fold					
	CV1	**CV2**	**CV3**	**CV4**	**CV5**
RD	**0.847774**	**0.848656**	**0.848949**	**0.848799**	**0.848521**
LCV	**0.604713**	**0.60523**	**0.606447**	**0.60607**	**0.606141**
RF	**0.966181**	**0.966406**	**0.966495**	**0.966441**	**0.966278**
GB	**0.924355**	**0.923462**	**0.924485**	**0.923539**	**0.923008**





The graph shows that Random Forest Regressor is the clear winner among the regressors used here. We can do further analysis on Ridge and Gradient Booster as well but for this study I will concentrate on Random Forest Regressor solely based on the above results.

Hyperparameter tuning for Random Forest

While model parameters are learned during training—such as the slope and intercept in a linear regression—hyperparameters must be set before training. Usually, we only have a vague idea of the best hyperparameters and thus the best approach to narrow our search is to evaluate a wide range of values for each hyperparameter.

Scikit-Learn implements a set of sensible default hyperparameters for all models, but these are not guaranteed to be optimal for a problem. The best hyperparameters are usually impossible to determine ahead of time, and tuning a model is where machine learning turns from a science into trial-and-error based engineering.

Here I will try to get the best parameters for Random Forest estimator before running the test validations on the estimators**.**

Below are the Hyperparameters I am going to optimize for my current work. There are other parameters as well which can be tuned, but here I am concentrating only on the below parameters.

Definition of Hyperparameters:

n_estimators = number of trees in the forest

max_features = max number of features considered for splitting a node

max_depth = max number of levels in each decision tree

min_samples_split = min number of data points placed in a node before the node is split

min_samples_leaf = min number of data points allowed in a leaf node

bootstrap = method for sampling data points (with or without replacement)

To begin our search for the best hyper parameter, we have to setup the baseline first so that we can compare our future results with them. The baseline hyperparameter values are given below. This is the simplest Random Forest Regressor with n_estimators = 1. Computationally it is the least expensive. If we cannot get better with this then it we don't need to train on more computationally expensive Random Forest regressor.

<u>Base Model Regression Hyperparameters::</u>

Base Model HyperParameters				
Hyperparameter	**Value**		**Hyperparameter**	**Value**
bootstrap	**TRUE**		**min_samples_split**	**2**
criterion	**mse**		**min_weight_fraction_leaf**	**0**
max_depth	**None**		**n_estimators**	**warn**
max_features	**auto**		**n_jobs**	**None**
max_leaf_nodes	**None**		**oob_score**	**FALSE**
min_impurity_decrease	**0**		**random_state**	**7**
min_impurity_split	**None**		**verbose**	**0**
min_samples_leaf	**1**		**warm_start**	**FALSE**

Then we ran the model on the validation set only. We will keep our Hold-Out Test data safe till we get all the results from the hyperparameter analysis results. In the end we will check which all of them works best for us.

Base Model Validation Run Metrics	
Mean Absolute Error	0.0634619
Mean Square Error	0.0117556
R square	0.9588077
Accuracy	0.9908

The performance of the least expensive regressor is very good. It is able to explain the 95.88% of variation on our validation dataset. Now this will be our base model and will compare the other expensive model's results from it.

Randomize Search CV:

We need to search the parameters and computationally it is very expensive to search on all the parameters. In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter.
To use RandomizedSearchCV, we first need to create a parameter grid to sample from during fitting:

Random Search CV Grid	
HyperParameters	Values
bootstrap	[True, False],
max_depth	[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
max_features	['auto', 'sqrt', 'log2'],
min_samples_leaf	[1, 2, 4],
min_samples_split	[2, 5, 10],
n_estimators	[5, 16, 28, 39, 51, 62, 74, 85, 97, 108, 120]

On each iteration, the algorithm will choose a difference combination of the features. Altogether, there are 2 * 12 * 3 * 3 * 3 * 11 = 7128 settings! However, the benefit of a random search is that we are not trying every combination but selecting at random to sample a wide range of values.

After instantiating the RandomSearchCV and fitting with our training dataset, we got the best parameters searched by the model. The result we got were:

Random Search CV Best parameters	
HyperParameters	Values
bootstrap	TRUE
max_depth	50
max_features	auto
min_samples_leaf	1
min_samples_split	50
n_estimators	120

Then I fit the model with the best parameters and checked the result. There was a 0.06% increment in the accuracy of the model comparing to base model which considers to be a good increment.

Random Search Validation Test Run Metrics	
Mean Absolute Error	0.059109038
Mean Square Error	0.00838429
R square	0.970620925
Accuracy	99.14%

Grid Search with Cross Validation

Random search allowed us to narrow down the range for each hyperparameter. Now that we know where to concentrate our search, we can explicitly specify every combination of settings to try. We do this with GridSearchCV, a method that, instead of sampling randomly from a distribution, evaluates all combinations we define. To use Grid Search, we make another grid based on the best values provided by random search:

Below is the run of 36 combinations of data with which Grid Search ran. It took 252 mins to complete with parallel jobs with 8 core CPU. You can see that how expensive it can be computationally.

Grid Search CV Grid	
HyperParameters	Values
bootstrap	[True, False],
max_depth	[30, 50]
max_features	['auto'],
min_samples_leaf	[1]
min_samples_split	[8,10],
n_estimators	[100, 120, 150]

There was a small change in the improvement with respect to R Square in grid search compare to Random Search. The change is not that considerable but n_estimators is changed from 120 to 150, which is again time taking process.

Grid Search CV Best parameters	
HyperParameters	**Values**
bootstrap	**TRUE**
max_depth	**30**
max_features	**auto**
min_samples_leaf	**1**
min_samples_split	**10**
n_estimators	**150**

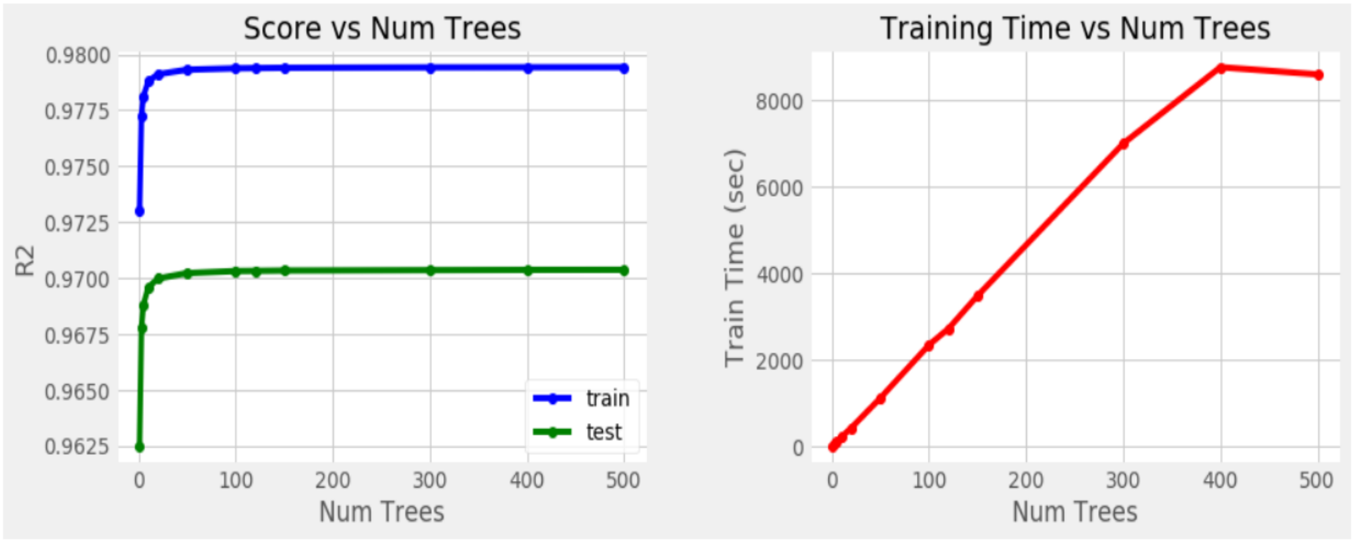
Grid Search CV Validation Test Run Metrics	
Mean Absolute Error	**0.059177773**
Mean Square Error	**0.008342143**
R square	**0.970768613**
Accuracy	**99.14%**

Other hyperparameter are almost constant now but the n_estimators is still taking the maximum value of it. This will tempt us to take one more round of Grid Search CV by changing only the n_estimators in it.

The next round of Grid Search has the below Grid parameters. This will give us the best n_estimator choice for the model. Keeping everything constant I have changed only the n_estimator as **[1,3,5,10,20,50,100,120,150,300,400,500]**. This search will give us idea to trade off between Accuracy of the model versus time to train the model.

The below graph is plotted with change in number of trees versus the Rsquare value. We can observe that there is no much to the change in the R-square value after '120'. But if you see the graph with Training Time vs number of trees, it shows us that it will exponentially expensive if the number of tress grow while the R square value change is negligible.

Plot of the metrics obtained from running Grid search Model



Grid Search CV2 Best Parameters Validation Test Run Metrics	
Mean Absolute Error	**0.059145726**
Mean Square Error	**0.008332583**
R square	**0.970802111**
Accuracy	**99.14%**

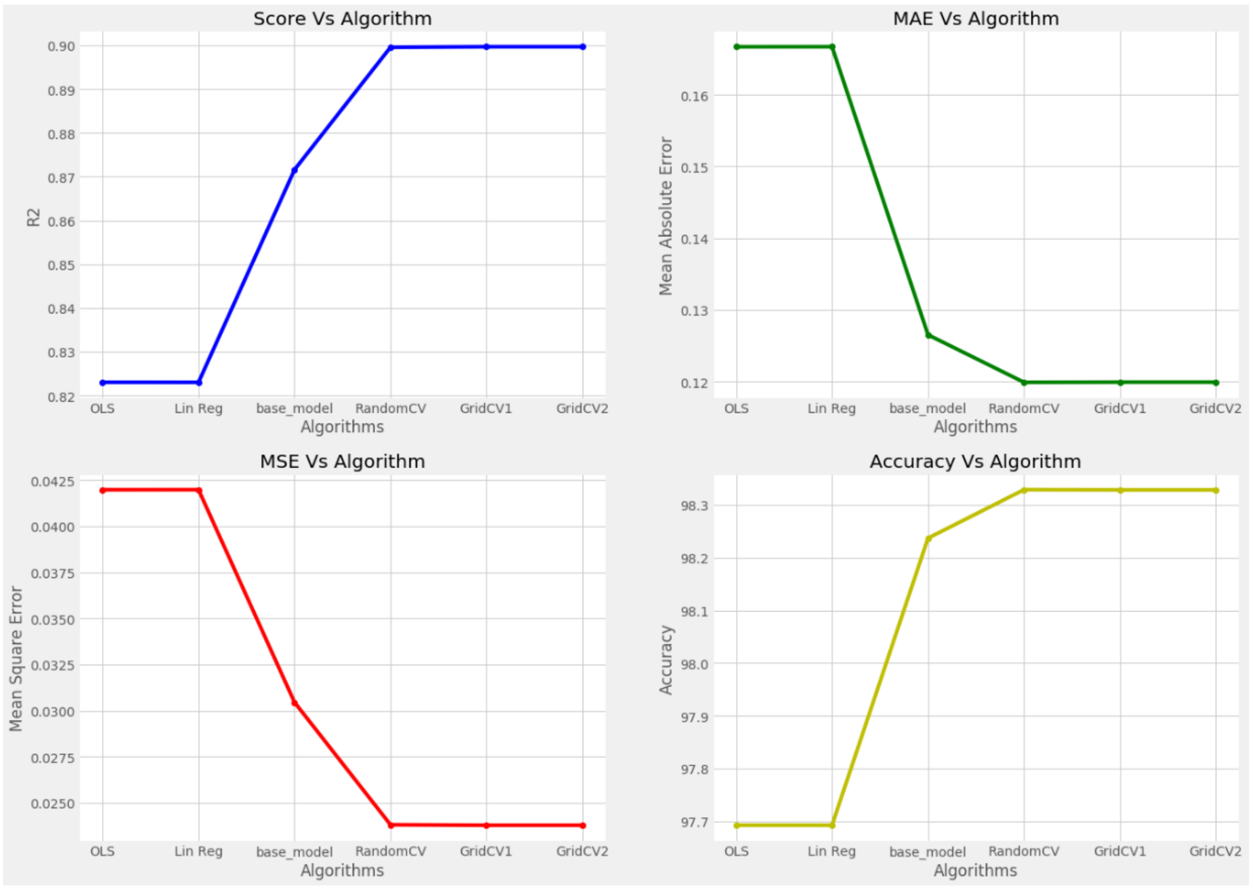
There is a negligible change in the R Square value but the n_estimator is changed from 120 to 500.

All Models Performances on Hold out Dataset:

After researching on all the models in contention, lets analyze them all together on the Hold Out dataset of year 2019. We can see that the best model that we got is the one with Grid Search CV2 having n_estimators as 500. But the overall change is almost negligible here as well.

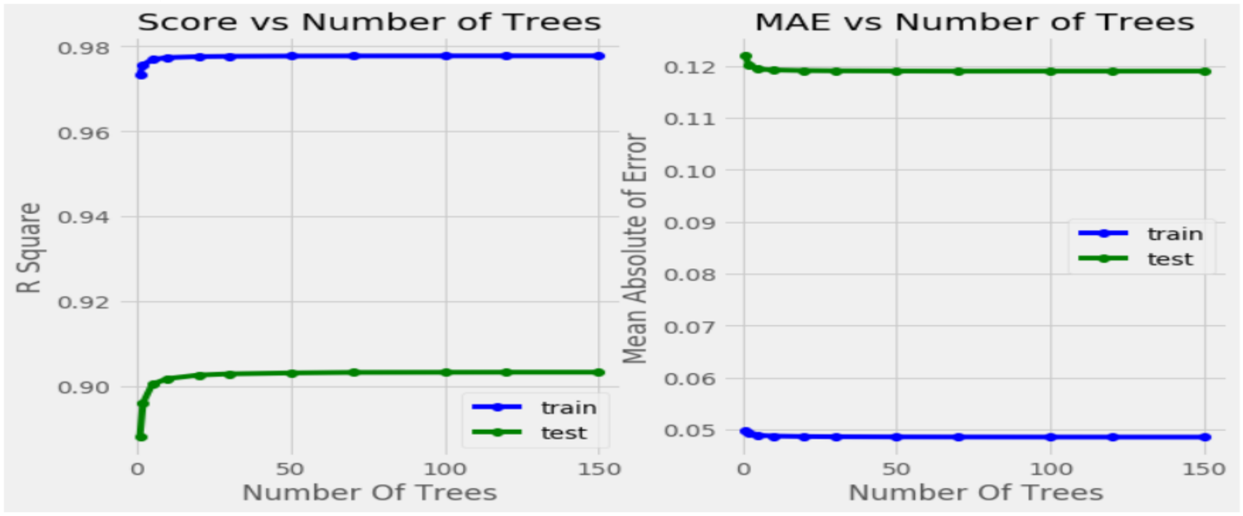
All Models Performace on Hold Out Data				
	R2	**MAE**	**MSE**	**Accuracy**
OLS	**0.823046**	**0.166688**	**0.041974**	**97.69262**
Lin Reg	**0.823035**	**0.166697**	**0.041976**	**97.69254**
base_model	**0.871604**	**0.126529**	**0.030456**	**98.23696**
RandomCV	**0.899587**	**0.119905**	**0.023818**	**98.32891**
GridCV1	**0.899701**	**0.119929**	**0.023791**	**98.32855**
GridCV2	**0.899701**	**0.119929**	**0.023791**	**98.32854**

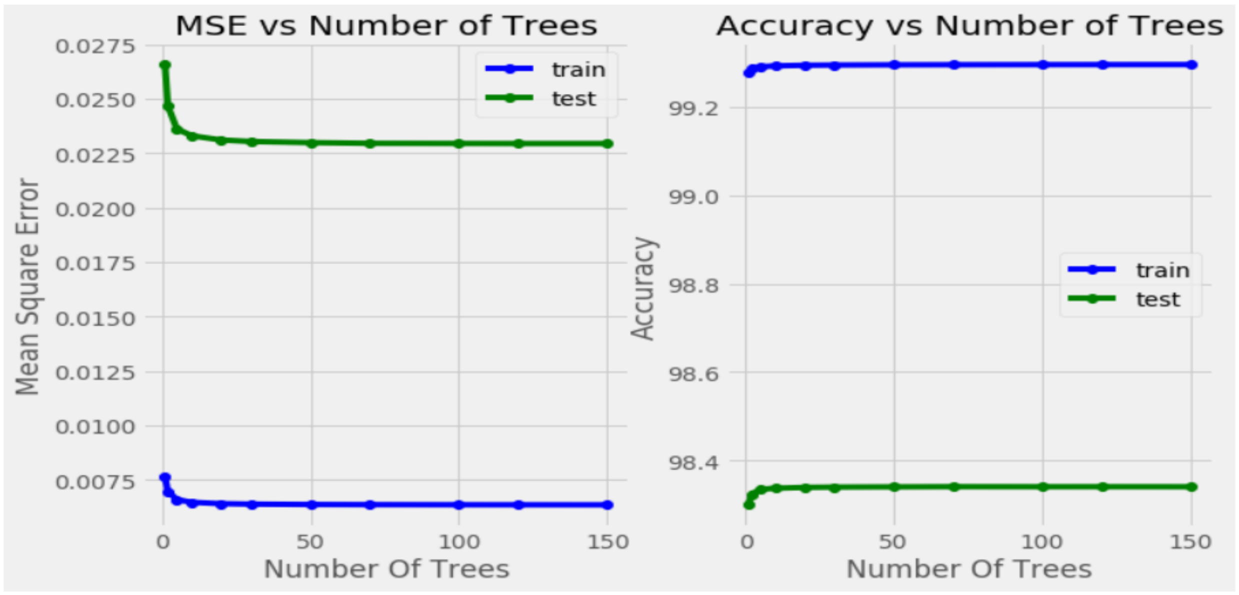
Plot of the metrics obtained from running the Hold Out Test data



Search of n_estimator from Hold-Out dataset

We got all the best parameters but n_estimator. So keeping all constant and running the random forest regressor with increment of n_estimators, I tried to see how much we gain if we increase the number of trees in it. More the n_estimator, more the time it takes to train the dataset. This exercise will also help us to drill down the optimum number of trees we need to work on. It is a trade off with time to Accuracy.





These graphs show us that there is no significant change in the Accuracy or R square after the n_estimator = 50. We may get some change in the accuracy after increasing the number of trees, but it will be an expensive trade-off with computational time. The accuracy for the n_estimator=120 is below:

RandomForestRegressor with n_estimator=120 Hold Out Test Run Metrics	
Mean Absolute Error	0.119427162
Mean Square Error	0.023630097
R square	0.900379764
Accuracy	98.34%

Limitations:

We have huge data from CMS and have all the details related to Plans and their rates, but we are getting only one side of the story. It would have been an interesting problem if we could get the data of the consumers as well and how they reacted to these plans. If we can get the Sales Data of the Plans from Marketplace or how many inquiries got for plan.

There are few data points which cannot be shared in public platform because of PHI and HIPPA rules. These data points can help us in getting the better predictions for the premium.

Handling the dataset of HealthCare Industry is never been an easy task. We have lots of data and every field of it is helpful in prediction. The data that we have taken is huge and sometimes extend 25GBs when merged together. Dealing with such kind of industrial data is difficult to run on individual desktop. Memory and Computational limitation will always be there. So, to complete the analysis and predictions I have to compromise with below points:

1. I have done my exploratory data analysis on all the data available but couldn't use my whole data for Machine Learning. For my Machine learning I used only the Florida Dataset.
2. I have rented the virtual machine from Microsoft Azure to do the analysis. These virtual machines are not that expensive if you take the basic machine, but it gets very expensive if we buy a heavy machine. So even the cloud wasn't that helpful to tackle the huge dataset that we have.
3. I have used parallel processing to process the Machine Learning Algorithms which really helps me in running the GridSearchCV. But I had the limitation with number of CPU cores in it. In my all run I did the parallelism of 8 cores only.

Future Work:

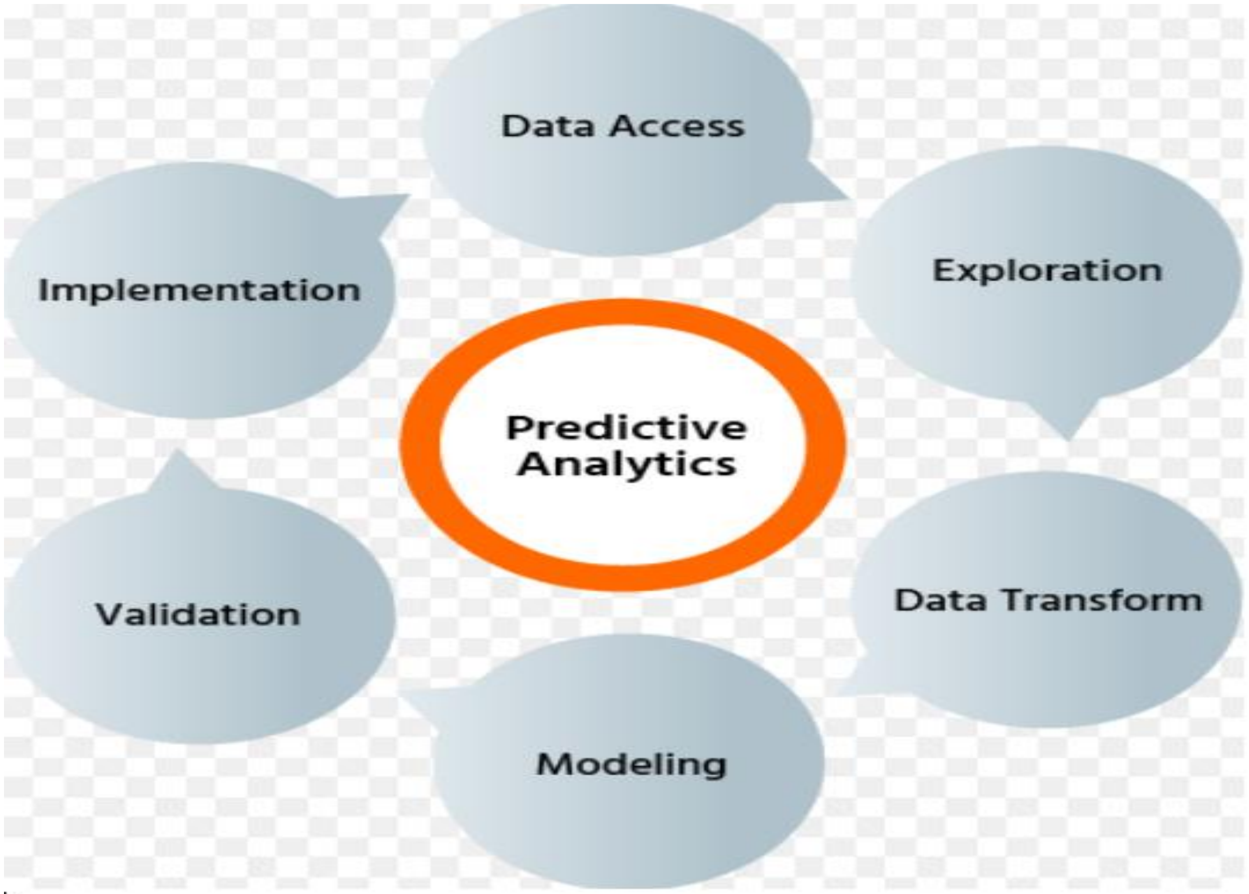
We have the data for all the states but due to few limitations we were able to perform Machine Learning on Florida Dataset only. It would be great if we can extend the learning to all the states.

The entire concept of exchange was to make sure that all the Americans are covered under the health insurance. It would be great to find out if these insurances are really helping people in getting quality medical services. Once we start digging about all this information, this analysis part of this project would be more helpful to the consumers and for the issuers.

In machine learning part would like to scale the model to predict couples, and couples with dependents. There are almost 5 to 7 kinds of rates which can be predicted.

Conclusions:

We acquired the data from CMS and tried to decipher the unknown trends in the data and in the process, we were able prove the established facts as well. The data was huge and its difficult to process everything together. So, funneling the data to get the best predictors is the most critical steps for us. Below diagram shows what steps we followed in completing this project.



Below I have briefed out what we did in our project

1. **Exploratory Data Analysis (EDA)**:
   During the EDA, we found that not all the States in the U.S participated in the Federal run health insurance marketplace or exchange. There are states which were part of this exchange in the beginning but later moved to state-based exchange. And there were states which were having State based exchange but later moved on to the federal run exchange.
   We have also seen the states who are large consumers of medical services like Wisconsin, Texas and Florida to name few.
   We have analyzed how the rates are spread across all over the U.S. What are the cheapest and costliest states to live in, in terms of healthcare.
   We had tried to find the relation among the number of issuers, number of benefit plans offered and the monthly premium rates across the U.S.

2. Machine Learning Models:

Our purpose of the project was to predict the monthly premium of the plan having variety of benefits. I took the baby steps to solve the problem. Once I have the clean data to perform the analysis, I started with the very basic model i.e. Linear Regression. It gave me an initial insight about the data behavior and trend. It helped me out to check the outliers in the data. Then I analyzed the data on Decision Tree which helps me in finding the important variables and in turn feature reduction. With this analysis we can reduce the dataset to almost half.

Once I get that the data is free of noise, I ran the data on most of the ensemble algorithm to check which model is giving us the best results. We got the clear winner i.e. Random Forest Regressor. With this algorithm we moved forward and did the Hyper Parameter Optimization.

We were able to get the accuracy of 98.34% with R-Square as .9037 using the best Hyper Parameters on the 2019 Dataset. Nonetheless this is very impressive results. We can get even better result then this but on the expense of time. As we analyzed, using the optimized Hyperparameters then the best one saves us lots of time with marginal accuracy loss. We can take that minimal loss if minimal Accuracy can be trade off.