Springboard Capstone-I
Medical Insurance Fraud Prediction

Deepak Sathyanarayanan

Contents

Introduction
Datasets
- Train Datasets
- Test Datasets

Approach
- Transformations
  - Merge 3 datasets together - consolidation (either by joins or unions)
  - Each record in the dataset resembles a visit made by the patient to the service provider (hospital/ physician). But we have to predict if the service provider is fraudulent or not. So we need to remove the hospital visit grain and group by the service providers
  - Categorical columns (flags) are either 1 or 2. Need to replace 2 by 0
  - Clean the dataset - check for missing values and outliers
- Exploratory Data Analysis
  - Univariate analysis
  - Bivariate analysis
  - Missing value treatment
  - Outlier treatment
- Model Building
  - Identification of useful columns
  - Classification Algorithms
  - PCA & Classification Algorithms
  - PCA & Hyper Parameter Tuning & Classification Algorithms

Introduction:

About Healthcare Frauds:

There are few organized health care groups who commit frauds. They take advantage of the loop holes present in the industry and do fraudulent medical claims which results in exponential increase of Medicare spending. These frauds may/ may not involve(s) a service providers, physicians, beneficiaries. Medical codes represent the treatment provided. These codes are not understandable by layman. Fraudsters take advantage of this and manipulate the medical codes and submit the claim

Most common frauds committed in health care are as follows:

Set up claims for procedures and services that did not take place. Billing for services using patient information that are obtained through identity theft. Submitting the same claim multiple times for the same service. Treatments which are not covered by insurance are falsely represented as procedures which are covered by insurance. This type of crime happens in cosmetic procedures. Do irrelevant medical procedures on patients in order to get payment from the insurance companies. Charge extra from the insurance providers and defer the patient's payment

Potential Clients:

Insurance providers/ industry.

Dataset:

Open source public data from Kaggle that has over a million data, each record containing hospital visits in the US region. The data set can be found here

The dataset is of .CSV format and has been divided into 3 csv files. They contain Inpatient claims details, Outpatient claims details and Beneficiary details. Another csv classifies whether the service providers (hospitals) are fraudulent or not. Using this we can train the model.

Dataset Information:

Train Datasets:

1) Beneficiary Details Data

Dataset Name: Train_Beneficiarydata-1542865627584.csv
(Rows, Columns): (138556, 25)
Header: BeneID; DOB; DOD; Gender; Race; RenalDiseaseIndicator; State; County; NoOfMonths_PartACov; NoOfMonths_PartBCov; ChronicCond_Alzheimer; ChronicCond_Heartfailure; ChronicCond_KidneyDisease; ChronicCond_Cancer; ChronicCond_ObstrPulmonary; ChronicCond_Depression; ChronicCond_Diabetes; ChronicCond_IschemicHeart; ChronicCond_Osteoporasis; ChronicCond_rheumatoidarthritis; ChronicCond_stroke; IPAnnualReimbursementAmt; IPAnnualDeductibleAmt; OPAnnualReimbursementAmt; OPAnnualDeductibleAmt
Description: This data set contains the details of each patient

```
bene.shape
```

```
(138556, 25)
```

```
bene.columns
```

```
Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
       'OPAnnualDeductibleAmt'],
      dtype='object')
```

```
bene.head(5).transpose()
```

	0	1	2	3	4
BeneID	BENE11001	BENE11002	BENE11003	BENE11004	BENE11005
DOB	1943-01-01	1936-09-01	1936-08-01	1922-07-01	1935-09-01
DOD	NaN	NaN	NaN	NaN	NaN
Gender	1	2	1	1	1
Race	1	1	1	1	1
RenalDiseaseIndicator	0	0	0	0	0
State	39	39	52	39	24
County	230	280	590	270	680
NoOfMonths_PartACov	12	12	12	12	12
NoOfMonths_PartBCov	12	12	12	12	12
ChronicCond_Alzheimer	1	2	1	1	2
ChronicCond_Heartfailure	2	2	2	1	2
ChronicCond_KidneyDisease	1	2	2	2	2
ChronicCond_Cancer	2	2	2	2	2
ChronicCond_ObstrPulmonary	2	2	2	2	1
ChronicCond_Depression	1	2	2	2	2
ChronicCond_Diabetes	1	2	2	1	1
ChronicCond_IschemicHeart	1	2	1	1	2
ChronicCond_Osteoporasis	2	2	2	1	2
ChronicCond_rheumatoidarthritis	1	2	2	1	2
ChronicCond_stroke	1	2	2	2	2
IPAnnualReimbursementAmt	36000	0	0	0	0
IPAnnualDeductibleAmt	3204	0	0	0	0
OPAnnualReimbursementAmt	60	30	90	1810	1790
OPAnnualDeductibleAmt	70	50	40	760	1200

2) Inpatient Data

Dataset Name: Train_Inpatientdata-1542865627584.csv
(Rows, Columns): (40474, 30)

Header:
BeneID;ClaimID;ClaimStartDt;ClaimEndDt;Provider;InscClaimAmtReimbursed;AttendingPhysician;OperatingPhysician;OtherPhysician;AdmissionDt;ClmAdmitDiagnosisCode;DeductibleAmtPaid;DischargeDt;DiagnosisGroupCode;ClmDiagnosisCode_1;ClmDiagnosisCode_2;ClmDiagnosisCode_3;ClmDiagnosisCode_4;ClmDiagnosisCode_5;ClmDiagnosisCode_6;ClmDiagnosisCode_7;ClmDiagnosisCode_8;ClmDiagnosisCode_9;ClmDiagnosisCode_10;ClmProcedureCode_1;ClmProcedureCode_2;ClmProcedureCode_3;ClmProcedureCode_4;ClmProcedureCode_5;ClmProcedureCode_
Description: This dataset contains the details of the claims filed. The patients are admitted in the hospital

```
In [7]: inpat.shape
Out[7]: (40474, 30)

In [8]: inpat.columns
Out[8]: Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
               'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
               'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
               'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
               'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
               'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
               'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
               'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
               'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
               'ClmProcedureCode_6'],
              dtype='object')
```

```
inpat.head(5).transpose()
```

	0	1	2	3	4
BeneID	BENE11001	BENE11001	BENE11001	BENE11011	BENE11014
ClaimID	CLM46614	CLM66048	CLM68358	CLM38412	CLM63689
ClaimStartDt	2009-04-12	2009-08-31	2009-09-17	2009-02-14	2009-08-13
ClaimEndDt	2009-04-18	2009-09-02	2009-09-20	2009-02-22	2009-08-30
Provider	PRV55912	PRV55907	PRV56046	PRV52405	PRV56614
InscClaimAmtReimbursed	26000	5000	5000	5000	10000
AttendingPhysician	PHY390922	PHY318495	PHY372395	PHY369659	PHY379376
OperatingPhysician	NaN	PHY318495	NaN	PHY392961	PHY398258
OtherPhysician	NaN	NaN	PHY324689	PHY349768	NaN
AdmissionDt	2009-04-12	2009-08-31	2009-09-17	2009-02-14	2009-08-13
ClmAdmitDiagnosisCode	7866	6186	29590	431	78321
DeductibleAmtPaid	1068	1068	1068	1068	1068
DischargeDt	2009-04-18	2009-09-02	2009-09-20	2009-02-22	2009-08-30
DiagnosisGroupCode	201	750	883	067	975
ClmDiagnosisCode_1	1970	6186	29623	43491	042
ClmDiagnosisCode_2	4019	2948	30390	2762	3051
ClmDiagnosisCode_3	5853	56400	71690	7843	34400
ClmDiagnosisCode_4	7843	NaN	34590	32723	5856
ClmDiagnosisCode_5	2768	NaN	V1581	V1041	42732
ClmDiagnosisCode_6	71590	NaN	32723	4254	486
ClmDiagnosisCode_7	2724	NaN	NaN	25062	5119
ClmDiagnosisCode_8	19889	NaN	NaN	40390	29620
ClmDiagnosisCode_9	5849	NaN	NaN	4019	20300
ClmDiagnosisCode_10	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_1	NaN	7092	NaN	331	3893
ClmProcedureCode_2	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_3	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_4	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_5	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_6	NaN	NaN	NaN	NaN	NaN

3) Outpatient Data

Dataset Name: Train_Outpatientdata-1542865627584.csv

(Rows, Columns): (517737, 27)

Header: BeneID;ClaimID;ClaimStartDt;ClaimEndDt;Provider;
InscClaimAmtReimbursed;AttendingPhysician;OperatingPhysician;
OtherPhysician;ClmDiagnosisCode_1;ClmDiagnosisCode_2;
ClmDiagnosisCode_3;ClmDiagnosisCode_4;ClmDiagnosisCode_5;
ClmDiagnosisCode_6;ClmDiagnosisCode_7;ClmDiagnosisCode_8;
ClmDiagnosisCode_9;ClmDiagnosisCode_10;ClmProcedureCode_1;
ClmProcedureCode_2;ClmProcedureCode_3;ClmProcedureCode_4;
ClmProcedureCode_5;ClmProcedureCode_6;DeductibleAmtPaid;
ClmAdmitDiagnosisCode

Description: This dataset contains the details of the claims filed. The patients are not admitted in the hospital

```
outpat.shape
```

```
(517737, 27)
```

```
outpat.columns
```

```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2',
       'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5',
       'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8',
       'ClmDiagnosisCode_9', 'ClmDiagnosisCode_10', 'ClmProcedureCode_1',
       'ClmProcedureCode_2', 'ClmProcedureCode_3', 'ClmProcedureCode_4',
       'ClmProcedureCode_5', 'ClmProcedureCode_6', 'DeductibleAmtPaid',
       'ClmAdmitDiagnosisCode'],
      dtype='object')
```

```
outpat.head(5).transpose()
```

	0	1	2	3	4
BeneID	BENE11002	BENE11003	BENE11003	BENE11004	BENE11004
ClaimID	CLM624349	CLM189947	CLM438021	CLM121801	CLM150998
ClaimStartDt	2009-10-11	2009-02-12	2009-06-27	2009-01-06	2009-01-22
ClaimEndDt	2009-10-11	2009-02-12	2009-06-27	2009-01-06	2009-01-22
Provider	PRV56011	PRV57610	PRV57595	PRV56011	PRV56011
InscClaimAmtReimbursed	30	80	10	40	200
AttendingPhysician	PHY326117	PHY362868	PHY328821	PHY334319	PHY403831
OperatingPhysician	NaN	NaN	NaN	NaN	NaN
OtherPhysician	NaN	NaN	NaN	NaN	NaN
ClmDiagnosisCode_1	78943	6115	2723	71988	82382
ClmDiagnosisCode_2	V5866	NaN	NaN	NaN	30000
ClmDiagnosisCode_3	V1272	NaN	NaN	NaN	72887
ClmDiagnosisCode_4	NaN	NaN	NaN	NaN	4280
ClmDiagnosisCode_5	NaN	NaN	NaN	NaN	7197
ClmDiagnosisCode_6	NaN	NaN	NaN	NaN	V4577
ClmDiagnosisCode_7	NaN	NaN	NaN	NaN	NaN
ClmDiagnosisCode_8	NaN	NaN	NaN	NaN	NaN
ClmDiagnosisCode_9	NaN	NaN	NaN	NaN	NaN
ClmDiagnosisCode_10	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_1	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_2	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_3	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_4	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_5	NaN	NaN	NaN	NaN	NaN
ClmProcedureCode_6	NaN	NaN	NaN	NaN	NaN
DeductibleAmtPaid	0	0	0	0	0
ClmAdmitDiagnosisCode	56409	79380	NaN	NaN	71947

Dataset Name: Train-1542865627584.csv

(Rows, Columns): (517737, 27)

Header: BeneID;ClaimID;ClaimStartDt;ClaimEndDt;Provider;
InscClaimAmtReimbursed;AttendingPhysician;OperatingPhysician;
OtherPhysician;ClmDiagnosisCode_1;ClmDiagnosisCode_2;
ClmDiagnosisCode_3;ClmDiagnosisCode_4;ClmDiagnosisCode_5;
ClmDiagnosisCode_6;ClmDiagnosisCode_7;ClmDiagnosisCode_8;
ClmDiagnosisCode_9;ClmDiagnosisCode_10;ClmProcedureCode_1;
ClmProcedureCode_2;ClmProcedureCode_3;ClmProcedureCode_4;
ClmProcedureCode_5;ClmProcedureCode_6;DeductibleAmtPaid;
ClmAdmitDiagnosisCode

Description: This dataset contains the classification of all the providers whether they are Fraud or not

```
train_classification.shape
```

```
(5410, 2)
```

```
train_classification.columns
```

```
Index(['Provider', 'PotentialFraud'], dtype='object')
```

```
train_classification.head(5).transpose()
```

	0	1	2	3	4
Provider	PRV51001	PRV51003	PRV51004	PRV51005	PRV51007
PotentialFraud	No	Yes	No	Yes	No

<u>To be Predicted:</u> Provider

Test Data

All the columns are same as the train dataset

Beneficiary Data

```
bene.shape
```

```
(63968, 25)
```

```
bene.columns
```

```
Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
       'OPAnnualDeductibleAmt'],
      dtype='object')
```

Inpatients Data

```
inpat.shape
```

```
(9551, 30)
```

```
inpat.columns
```

```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
       'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
       'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
       'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
       'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
       'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
       'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
       'ClmProcedureCode_6'],
      dtype='object')
```

Outpatients Data

```
outpat.shape
```

```
(125841, 27)
```

```
outpat.columns
```

```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2',
       'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5',
       'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8',
       'ClmDiagnosisCode_9', 'ClmDiagnosisCode_10', 'ClmProcedureCode_1',
       'ClmProcedureCode_2', 'ClmProcedureCode_3', 'ClmProcedureCode_4',
       'ClmProcedureCode_5', 'ClmProcedureCode_6', 'DeductibleAmtPaid',
       'ClmAdmitDiagnosisCode'],
      dtype='object')
```

Test Data

```
test.shape
```

```
(1353, 1)
```

```
test.columns
```

```
Index(['Provider'], dtype='object')
```

As we can see that there are 4 different data sets. Our final goal is to predict if the hospital provider was making fraud claims or not. So we need to make some transformations to the datasets

We want the pre-processed columns like the below…

Provider	ChronicCond_Cancer_median
Age_median	ChronicCond_Cancer_mode
Age_mode	ChronicCond_Depression_median
Deceased_sum	ChronicCond_Depression_mode
Gender_mode	ChronicCond_Diabetes_median
Race_mode	ChronicCond_Diabetes_mode
State_code_mode	ChronicCond_Heartfailure_median
County_code_mode	ChronicCond_Heartfailure_mode
No_days_admitted_median	ChronicCond_IschemicHeart_median
No_days_admitted_mode	ChronicCond_IschemicHeart_mode
No_days_ClaimProcessing_median	ChronicCond_KidneyDisease_median
No_days_ClaimProcessing_mean	ChronicCond_KidneyDisease_mode
Attending_Physician_mode	ChronicCond_ObstrPulmonary_median
OperatingPhysician_mode	ChronicCond_ObstrPulmonary_mode
Other_Physician_mode	ChronicCond_Osteoporasis_median
RenalDiseaseIndicator_mode	ChronicCond_Osteoporasis_mode
ChronicCond_Alzheimer_median	ChronicCond_rheumatoidarthritis_median
ChronicCond_Alzheimer_mode	ChronicCond_rheumatoidarthritis_mode
ChronicCond_stroke_median	OPAnnualDeductibleAmt_median
ChronicCond_stroke_mode	OPAnnualReimbursementAmt_sum
NoOfMonths_PartACov_median	OPAnnualReimbursementAmt_median
NoOfMonths_PartACov_mode	IPAnnualDeductibleAmt_sum
NoOfMonths_PartBCov_median	IPAnnualDeductibleAmt_median
NoOfMonths_PartBCov_mode	IPAnnualReimbursementAmt_sum
no_of_service_provided	IPAnnualReimbursementAmt_median
InscClaimAmtReimbursed_sum	DeductibleAmtPaid_sum
InscClaimAmtReimbursed_median	DeductibleAmtPaid_median
OPAnnualDeductibleAmt_sum	PotentialFraud(Target)

Approach:

- Transformations
  - Merge 3 datasets together - consolidation (either by joins or unions)
  - Each record in the dataset resembles a visit made by the patient to the service provider (hospital/ physician). But we have to predict if the service provider is fraudulent or not. So we need to remove the hospital visit grain and group by the service providers
  - Categorical columns (flags) are either 1 or 2. Need to replace 2 by 0
  - Clean the dataset - check for missing values and outliers
- Exploratory Data Analysis
  - Univariate analysis
  - Bivariate analysis
  - Missing value treatment
  - Outlier treatment
- Model Building
  - Identification of useful columns
  - Classification Algorithms
  - PCA & Classification Algorithms
  - PCA & Hyper Parameter Tuning & Classification Algorithms

Transformation:

```
inpat_bene = pd.merge(inpat, bene, left_on = 'BeneID',right_on = 'BeneID', how = 'left')
```

```
outpat_bene = pd.merge(outpat, bene, left_on = 'BeneID',right_on = 'BeneID', how = 'left')
```

```
train = pd.concat([inpat_bene, outpat_bene])
```

Steps:

1. Left join on inpatients dataset and beneficiary dataset – inpat_bene
2. Left Join on outpatients dataset and beneficiary data set – outpat_bene
3. Union of inpat_bene & outpat_bene - train
4. Perform Group by with respect to Provider column
5. Inner Join on train classification & train – train_df

The train_df has 5,58,211 rows and 55 columns

```
train_df.shape
```

```
(558211, 55)
```

Sample data of train_df:

```
train_df.head().transpose()
```

	0	1	2	3	4
Provider	PRV55912	PRV55907	PRV56046	PRV52405	PRV56614
age_mode	67	81	69	71	72
age_median	75	75	70	71	72
deceased_sum	105	242	20	88	26
Gender_mode	0	0	0	0	0
Race_mode	1	1	1	1	1
State_mode	39	39	39	1	45
County_mode	510	770	230	340	780
No_days_admitted_median	2	1	3	1	1.5
No_days_admitted_mode	1	1	1	1	1
No_days_ClaimProcessing_median	3	0	3	0	2
No_days_ClaimProcessing_mode	0	0	0	0	0
AttendingPhysician_mode	PHY414999	PHY331860	PHY324689	PHY369659	PHY338104
RenalDiseaseIndicator_mode	0	0	0	0	0
ChronicCond_Alzheimer_median	2	2	1	2	2
ChronicCond_Alzheimer_mode	2	2	1	2	2
ChronicCond_Cancer_median	2	2	2	2	2
ChronicCond_Cancer_mode	2	2	2	2	2
ChronicCond_Depression_median	2	2	1	1	1
ChronicCond_Depression_mode	2	2	1	1	1
ChronicCond_Diabetes_median	1	1	1	1	1
ChronicCond_Diabetes_mode	1	1	1	1	1
ChronicCond_Heartfailure_median	1	1	1	1	1
ChronicCond_Heartfailure_mode	1	1	1	1	1
ChronicCond_IschemicHeart_median	1	1	1	1	1
ChronicCond_IschemicHeart_mode	1	1	1	1	1
ChronicCond_KidneyDisease_median	1	2	1	2	1
ChronicCond_KidneyDisease_mode	1	2	1	2	1
ChronicCond_ObstrPulmonary_median	2	2	2	2	1
ChronicCond_ObstrPulmonary_mode	2	2	2	2	1
ChronicCond_Osteoporasis_median	2	2	1.5	2	2
ChronicCond_Osteoporasis_mode	2	2	1	2	2
ChronicCond_rheumatoidarthritis_median	2	2	2	2	2
ChronicCond_rheumatoidarthritis_mode	2	2	2	2	2
ChronicCond_stroke_median	2	2	2	2	2
ChronicCond_stroke_mode	2	2	2	2	2
NoOfMonths_PartACov_median	12	12	12	12	12
NoOfMonths_PartACov_mode	12	12	12	12	12
NoOfMonths_PartBCov_median	12	12	12	12	12
NoOfMonths_PartBCov_mode	12	12	12	12	12
InscClaimAmtReimbursed_sum	750080	598180	121680	189070	126100
InscClaimAmtReimbursed_median	4000	100	5000	100	2500
OPAnnualDeductibleAmt_sum	59690	171700	4730	88310	28700
OPAnnualDeductibleAmt_median	300	420	185	330	400
OPAnnualReimbursementAmt_sum	249270	608230	24050	251730	103960
OPAnnualReimbursementAmt_median	900	1210	700	970	970
IPAnnualDeductibleAmt_sum	148908	204920	33632	65148	33108
IPAnnualDeductibleAmt_median	1068	1068	1068	0	1068
IPAnnualReimbursementAmt_sum	1420920	1946470	366840	656110	311840

IPAnnualReimbursementAmt_median	8600	3000	7520	0	5500
DeductibleAmtPaid_sum	64110	64490	14952	23606	13884
DeductibleAmtPaid_median	1068	0	1068	0	1068
OperatingPhysician_mode	[PHY370861]	[PHY375412]	[PHY372395]	[PHY325656]	[PHY398258]
OtherPhysician_mode	[PHY387346]	[PHY347835, PHY348866]	[PHY324689]	[PHY369659]	[PHY338104]
no_of_service_provided	107	243	20	89	26
PotentialFraud	Yes	No	No	No	No

Conversion of objects to datetime format

```
train['DOB'] = pd.to_datetime(train['DOB'])
train['DOD'] = pd.to_datetime(train['DOD'])
train['ClaimEndDt'] = pd.to_datetime(train['ClaimEndDt'])
train['ClaimStartDt'] = pd.to_datetime(train['ClaimStartDt'])
train['AdmissionDt'] = pd.to_datetime(train['AdmissionDt'])
train['DischargeDt'] = pd.to_datetime(train['DischargeDt'])
```

We want our final dataset to have unique Provider columns so that we can predict if the corresponding provider is fraud or not. So we have grouped by all the columns with respect to Provider…

Also, we are converting the Target Variable into numerical column by transforming Yes to 1 & No to 0

```
#Changing the "PotentialFraud" column from Yes/No to a number as 1/0
PotentialFraud_map = {'Yes' : 1, 'No' : 0}
data_train['PotentialFraud'] = data_train['PotentialFraud'].map(PotentialFraud_map)
```

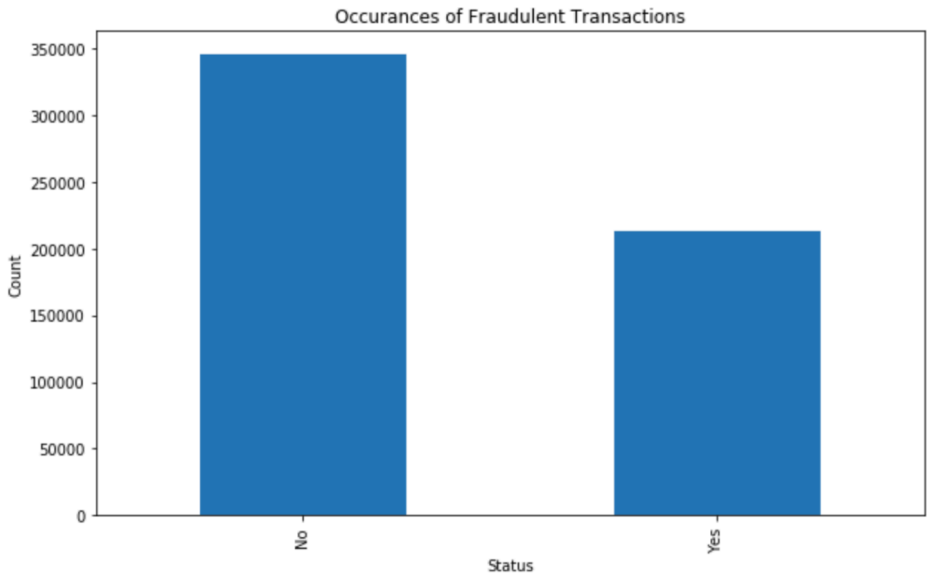
Missing Values:

```
# since we know which column has missing data, we are going to fill the column with mean value of that attribute
train_df['DeductibleAmtPaid_median'] = train_df['DeductibleAmtPaid_median'].fillna(train_df['DeductibleAmtPaid_median']
```

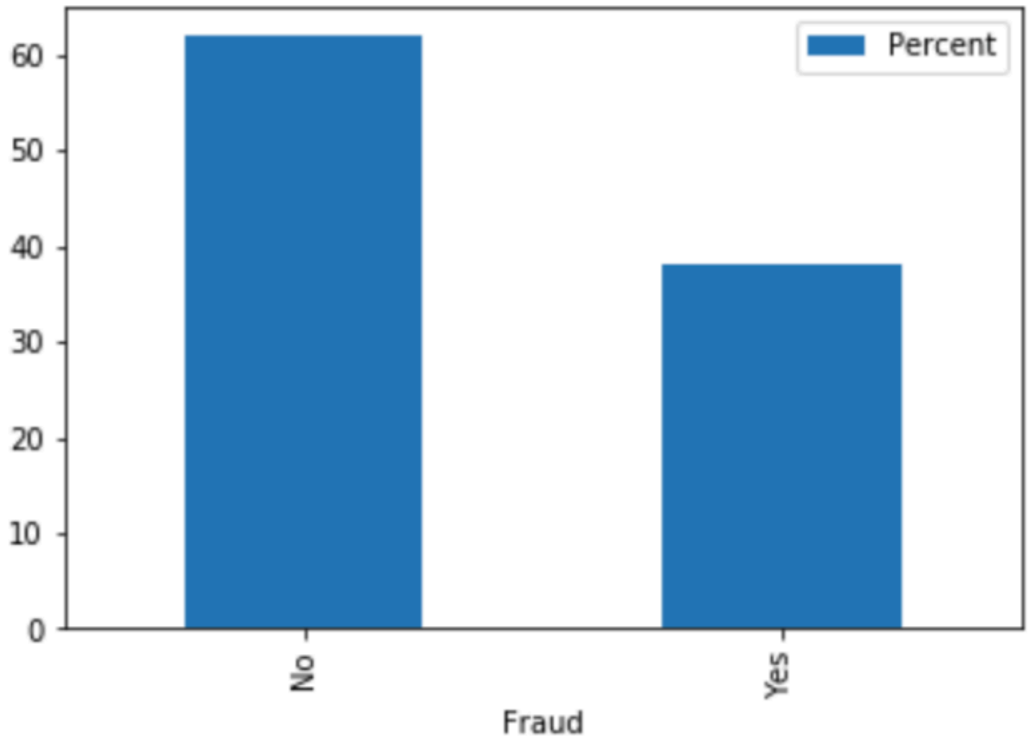
Exploratory Data Analysis & Visualization

Totally we have 212,796 valid claims and 345,415 Fraud claims

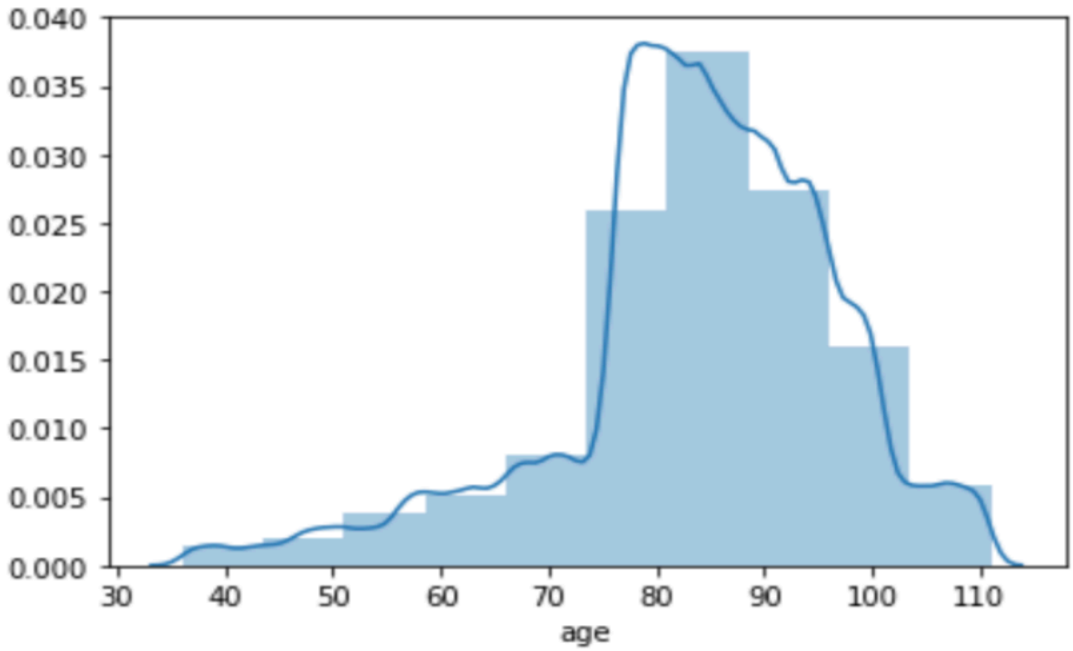
```
No      345415
Yes     212796
Name: PotentialFraud, dtype: int64
```



Representation in Percentage:

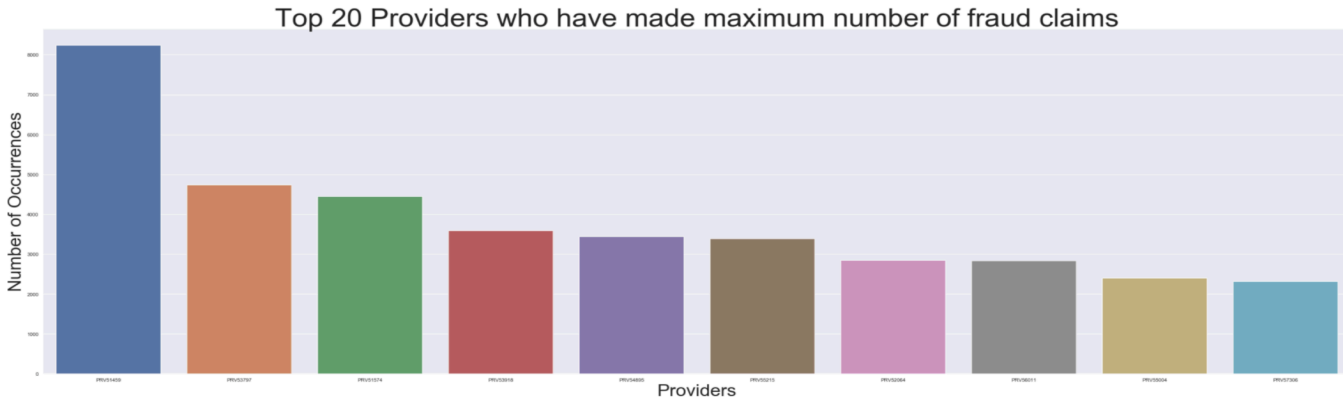


Age Distribution of Fraudulent claims:



Top 20 providers who are frequently committing fraud

```
PRV51459    8240
PRV53797    4739
PRV51574    4444
PRV53918    3588
PRV54895    3436
PRV55215    3393
PRV52064    2844
PRV56011    2833
PRV55004    2399
PRV57306    2315
PRV56560    2313
PRV52030    2275
PRV52649    2156
PRV54772    2115
PRV52628    2098
PRV51369    2083
PRV51347    2067
PRV55039    2058
PRV57103    2049
PRV52019    1961
Name: Provider, dtype: int64
```



Top 10 InscClaimAmtReimbursed which are fraudulent

	Provider	max_InscClaimAmtReimbursed
0	PRV55172	125000
1	PRV54742	125000
2	PRV53461	125000
3	PRV52815	125000
4	PRV54942	124000
5	PRV52467	123000
6	PRV57173	120000
7	PRV53706	119000
8	PRV51119	118000
9	PRV53762	112000

Top 10 providers & their count of fraudulent claims

	Provider	PotentialFraud	top_10_scam_count
0	PRV51459	Yes	8240
1	PRV53797	Yes	4739
2	PRV51574	Yes	4444
3	PRV53918	Yes	3588
4	PRV54895	Yes	3436
5	PRV55215	Yes	3393
6	PRV52064	Yes	2844
7	PRV56011	Yes	2833
8	PRV55004	Yes	2399
9	PRV57306	Yes	2315

Some in depth analysis as follows…

Checking if any particular physician of the hospital provider is responsible for the fraudulent claim

	Provider	AttendingPhysician	OperatingPhysician	OtherPhysician	count
0	PRV53918	PHY330576	None	None	1332
1	PRV51567	PHY350277	None	None	813
2	PRV53797	PHY412132	None	None	661
3	PRV51459	PHY338032	None	None	631
4	PRV52652	PHY357307	None	None	610
5	PRV51459	PHY423534	None	None	608
6	PRV51459	PHY337425	None	None	603
7	PRV51459	PHY314027	None	None	600
8	PRV51459	PHY357120	None	None	590
9	PRV52628	PHY432650	None	None	587
10	PRV51459	PHY327046	None	None	586
11	PRV51459	PHY341578	None	None	576
12	PRV57191	PHY424712	None	None	570
13	PRV51369	PHY347064	None	None	558
14	PRV52628	PHY344389	None	None	537
15	PRV51567	PHY350277	None	PHY350277	522
16	PRV51369	PHY383481	None	None	521
17	PRV57191	PHY381249	None	None	507
18	PRV51574	PHY415321	None	None	502
19	PRV55034	PHY418434	None	None	485

We can see that there were 1332 occurances of fraud claims that happened when AttendingPhysician - PHY330576 was the a part of the claim

---

Now let us drill down and find the fraud claims made whenever 'PHY330576' was the AttendingPhysician

	Provider	AttendingPhysician	PotentialFraud	count
0	PRV53918	PHY330576	Yes	2534
1	PRV53918	PHY359424	Yes	101
2	PRV53918	PHY367783	Yes	92
3	PRV53918	PHY325821	Yes	91
4	PRV53918	PHY365507	Yes	90
5	PRV53918	PHY412631	Yes	87
6	PRV53918	PHY340158	Yes	85
7	PRV53918	PHY375607	Yes	52
8	PRV53918	PHY407804	Yes	51
9	PRV53918	PHY408463	Yes	48
10	PRV53918	PHY388395	Yes	47
11	PRV53918	PHY390615	Yes	47
12	PRV53918	PHY428134	Yes	47
13	PRV53918	PHY318455	Yes	45
14	PRV53918	PHY351111	Yes	44
15	PRV53918	PHY410990	Yes	43
16	PRV53918	PHY311394	Yes	41
17	PRV53918	PHY432213	Yes	37
18	PRV53918	None	Yes	6

We can see that there were 2534 fraud claims made whenever PHY330576 was the AttendingPhysician

---

Let us check the total fraudulent and real claims done by the corresponding provider

	Provider	PotentialFraud	count
0	PRV53918	Yes	3588

It looks like all the claims by the provider were fraudulent

In the dataset, there are certain codes (ClmDiagnosisCode_1, ClmDiagnosisCode_1,..etc) are like flags.

Aim: To find which code is frequent in all the fraudlent claims.

Steps:

- We are grouping by providers (P.S: Providers are not unique here)
- Again sub grouping all of them by the codes
- Excluding Nan & None from all the codes
- Setting the filter condition (i.e) PotentialFraud='Yes'
- Aggregating all the groups by count
- Creating a outer for ranks
- Rank is - partitioned by Provider & ordered by count in descending order
- We are using the filter condition rank=1

#1 rank of maximum(codes) processed corresponding to claims grouped by providers

Provider	ClmDiagnosisCode_1	ClmDiagnosisCode_2	ClmDiagnosisCode_3	...	ClmProcedureCode_5	ClmProcedureCode_6	count	code_ranks
PRV51459	4019	None	None	...	NaN	None	49	1
PRV53797	4019	None	None	...	NaN	None	34	1
PRV51574	4011	None	None	...	NaN	None	28	1
PRV55215	4011	None	None	...	NaN	None	27	1
PRV54895	4019	None	None	...	NaN	None	24	1
PRV53918	4019	None	None	...	NaN	None	22	1
PRV55004	4019	None	None	...	NaN	None	21	1
PRV56011	4011	None	None	...	NaN	None	19	1
PRV54070	4011	None	None	...	NaN	None	18	1
PRV54778	4019	None	None	...	NaN	None	18	1
PRV57306	4019	None	None	...	NaN	None	18	1

We can conclude that probability of fraudulent claims is high whenever there is a ClmDiagnosisCode_1

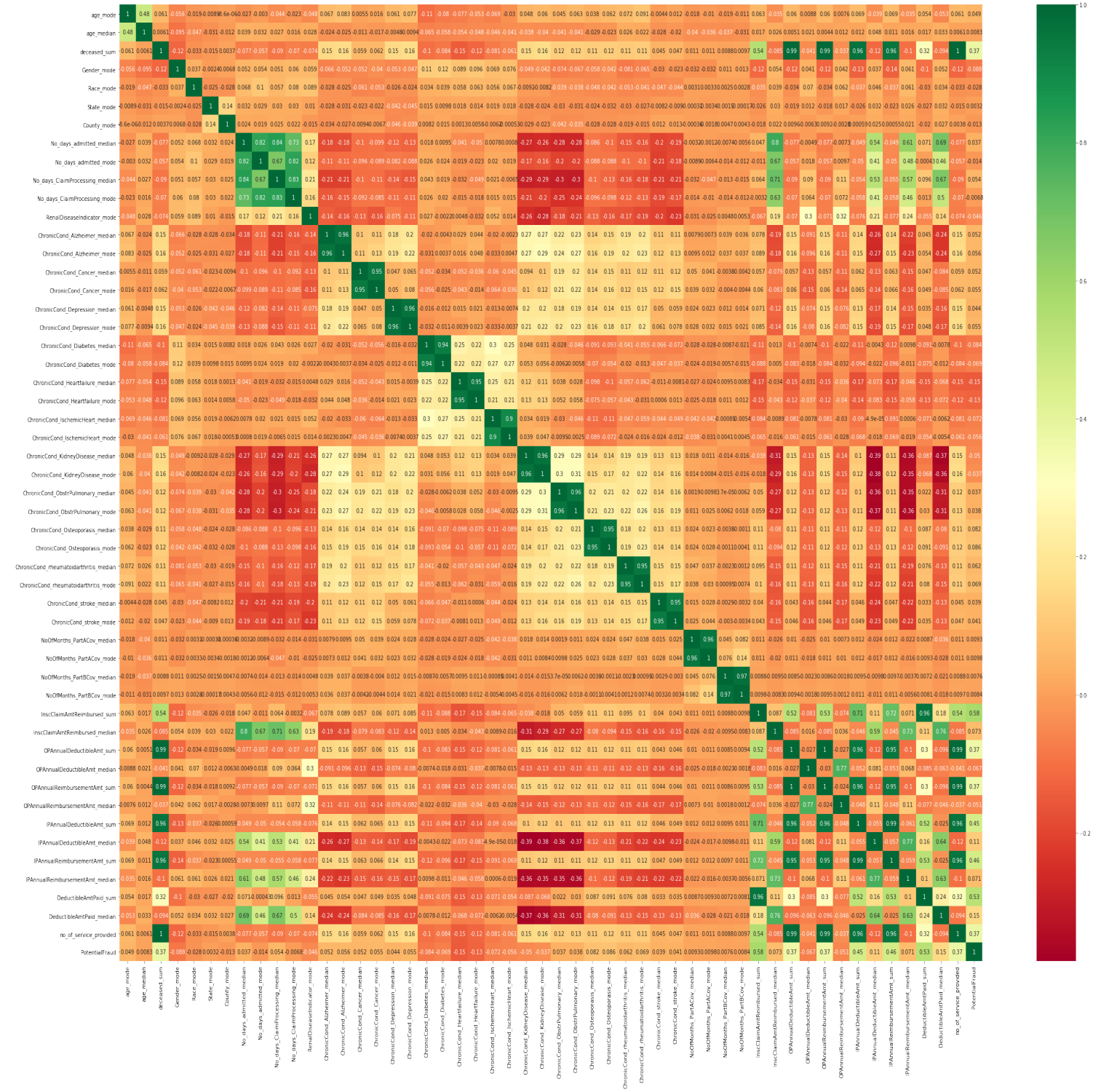
Model Building:

a. Identification of useful columns
b. Classification Algorithms
c. Hyperparameter Tuning & Classification Algorithms
d. PCA & Hyper Parameter Tuning & Classification Algorithms

Identification of useful features:

By correlation, there were few columns identified which was multi correlated. So ignored those columns.

Correlation Plot:



Columns that are multi collinear are

OPAnnualDeductibleAmt_sum	ChronicCond_IschemicHeart_mode	ChronicCond_ObstrPulmonary_mode
OPAnnualReimbursementAmt_sum	ChronicCond_Alzheimer_mode	ChronicCond_Osteoporasis_mode
IPAnnualDeductibleAmt_sum	ChronicCond_Cancer_mode	ChronicCond_rheumatoidarthritis_mode
IPAnnualReimbursementAmt_sum	ChronicCond_Depression_mode	ChronicCond_stroke_mode
no_of_service_provided	ChronicCond_Diabetes_mode	NoOfMonths_PartACov_mode
No_days_admitted_mode	ChronicCond_Heartfailure_mode	NoOfMonths_PartBCov_mode
No_days_ClaimProcessing_median	ChronicCond_KidneyDisease_mode	InscClaimAmtReimbursed_sum
No_days_ClaimProcessing_mode		

Hence ignoring the above columns.

Train Test Split: We have used 70-30 train test split.

```
## Train Test Split

from sklearn.model_selection import train_test_split

X1 = data_train[feature_columns1].values
y1 = data_train[predicted_class1].values

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size = 0.30, random_state=10)
```

<u>Model Building:</u>

Used ensemble model – Random Forest to build model on the training data

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=10, verbose=0, warm_start=False)
```

```
Accuracy = 0.933
```

HyperParameter Tuning:

Grid-search is used to find the optimal *hyperparameters* of a model which results in the most 'accurate' predictions.

```
{'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 100, 'n_estimators': 500}
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=15, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=100,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=1, verbose=0, warm_start=False)
```

```
Accuracy = 0.937
```

In *Randomized Search CV*, we can define a grid of hyperparameter ranges, and randomly sample from the grid, performing K-Fold CV with each combination of values.

```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
          estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=1, verbose=0, warm_start=False),
          fit_params=None, iid='warn', n_iter=500, n_jobs=-1,
          param_distributions={'n_estimators': [100, 300, 500, 800, 1200], 'max_depth': [5, 8, 15, 25, 30], 'min_samp
les_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]},
          pre_dispatch='2*n_jobs', random_state=42, refit=True,
          return_train_score='warn', scoring=None, verbose=2)
```

```
RandomF.best_params_
```

```
{'n_estimators': 100,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_depth': 25}
```

```
y_predF = RandomF.predict(X1_test)
print("Accuracy = {0:.3f}".format(metrics.accuracy_score(y1_test, y_predF)))
```

```
Accuracy = 0.935
```

PCA:

Now let us see, instead of building a identifying and dropping the columns manually, if we use Principle Component Analysis, we will get low dimensional set of features by taking a projection of irrelevant dimensions from a high dimensional data set. It captures maximum information with minimum loss of data

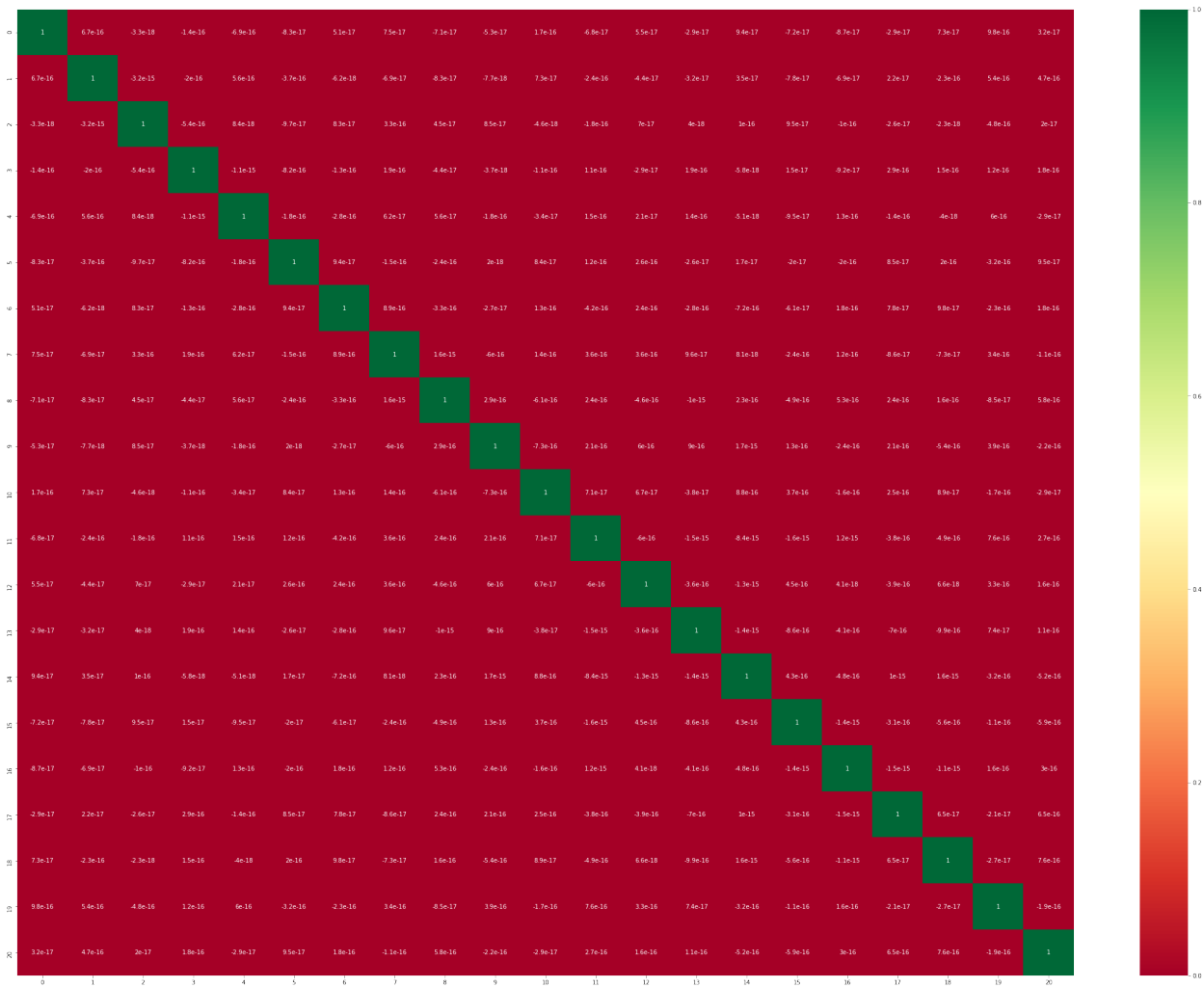
```
PCA(copy=True, iterated_power='auto', n_components=21, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

```
(21, 2)
    Cumulative Variance Ratio  Explained Variance Ratio
0                    0.214828                  0.214828
1                    0.169105                  0.383933
2                    0.065332                  0.449266
3                    0.054023                  0.503288
4                    0.047567                  0.550856
5                    0.043495                  0.594350
6                    0.038866                  0.633217
7                    0.032728                  0.665945
8                    0.031163                  0.697107
9                    0.029222                  0.726329
10                   0.029027                  0.755356
11                   0.026731                  0.782087
12                   0.025125                  0.807211
13                   0.024729                  0.831941
14                   0.023790                  0.855730
15                   0.023050                  0.878780
16                   0.021937                  0.900718
17                   0.020581                  0.921299
18                   0.019865                  0.941163
19                   0.015353                  0.956517
20                   0.012932                  0.969449
```

A look at the Head Transpose of the PCA components

	0	1	2	3	4	5	6	7	8	9
0	-0.193893	-0.767107	-0.912325	-0.345870	-0.340303	-0.242838	0.285701	1.418347	0.256473	1.993524
1	-1.400968	-1.058003	-1.170559	0.197976	-0.833925	-1.485912	0.356631	-0.711023	-1.162837	-0.343018
2	-0.556511	-0.281633	0.844526	-0.702328	0.345709	-1.419155	-0.183566	-0.067524	-0.058130	-0.122043
3	-0.140357	0.202787	-0.024554	-1.099081	-1.190982	0.189405	-1.204410	-0.159399	-0.185122	-0.337604
4	-0.108809	0.215886	2.466577	-0.315198	0.360241	-0.226186	-0.184574	0.580890	0.076697	0.351843
5	1.033727	-1.034713	-2.174578	-0.047187	0.387378	0.682039	-0.979771	0.781483	1.078694	0.468384
6	0.086832	-1.637847	-0.172514	0.095310	0.137685	-0.559736	-0.737344	-0.020235	0.223777	0.308921
7	0.109015	-0.701466	-0.864135	-0.203452	-0.113404	0.039391	-0.004673	0.001298	0.154697	0.023751
8	0.433106	-0.341868	-0.471856	0.284919	0.131680	0.404383	-1.144286	0.262226	0.347033	0.219887
9	0.398755	0.914859	-0.743371	0.326021	-0.070437	0.502982	-0.624059	0.225659	0.254909	0.230916

PCA Component's Correlation:



We can see that none of the components are multi correlated with each other

Now building Random Forest on the components, we get..

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
#Now lets predict and find the accuracy of our model
y_predict = rfc.predict(x_test_pca)

print("Accuracy = {0:.3f}".format(metrics.accuracy_score(y_test, y_predict)))
```

```
Accuracy = 0.934
```

Let us use PCA + Hyper Parameter Tuning (Grid Search) + Model building

We get a accuracy of…

```
{'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 15, 'n_estimators': 800}
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=15,
            min_weight_fraction_leaf=0.0, n_estimators=800, n_jobs=None,
            oob_score=False, random_state=1, verbose=0, warm_start=False)
```

```
y_predict_grid = gridF.predict(x_test_pca)
print("Accuracy = {0:.3f}".format(metrics.accuracy_score(y_test, y_predict_grid)))
```

```
Accuracy = 0.938
```

If we use PCA + Hyper Parameter Tuning (Randomized Search CV) + Model Building

We get a accuracy of…

```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
          estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
            oob_score=False, random_state=1, verbose=0, warm_start=False),
          fit_params=None, iid='warn', n_iter=500, n_jobs=-1,
          param_distributions={'n_estimators': [100, 300, 500, 800, 1200], 'max_depth': [5, 8, 15, 25, 30], 'min_samp
les_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]},
          pre_dispatch='2*n_jobs', random_state=42, refit=True,
          return_train_score='warn', scoring=None, verbose=2)
```

```
RandomF.best_params_
```

```
{'n_estimators': 100,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_depth': 15}
```

```
y_predict_randomSearhCV = RandomF.predict(x_test_pca)
print("Accuracy = {0:.3f}".format(metrics.accuracy_score(y_test, y_predict_randomSearhCV)))
```

```
Accuracy = 0.935
```

In a similar fashion, if we try to build different models by using the PCA components and Hyper Parameter Tuning, we get the following accuracy…

Models	Generic	Grid Search	Randomized Search CV
#RandomForestClassifier	0.934	0.938	0.935
#LogisticRegression	0.939	0.94	0.94
#KNeighborsClassifier	0.931	0.937	0.938
#SVM.SVC(kernel)	0.94	0.94	0.94
#GaussianProcessClassifier	0.935	NA	NA
#MLPClassifier	0.932	0.939	0.937
#AdaBoostClassifier	0.936	0.935	0.935