Link to the dataset

https://www.kaggle.com/karthickveerakumar/claims-data/data

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

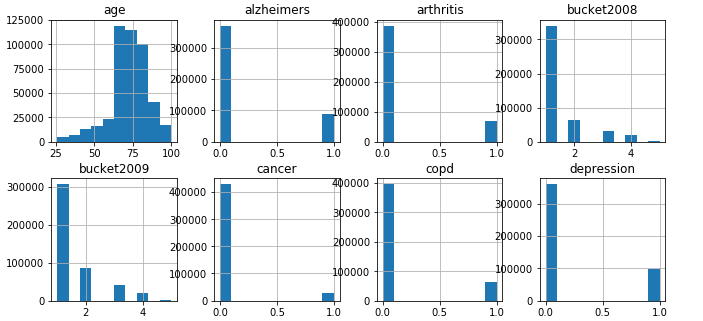
dataset = pd.read\_csv('ClaimsData.csv')

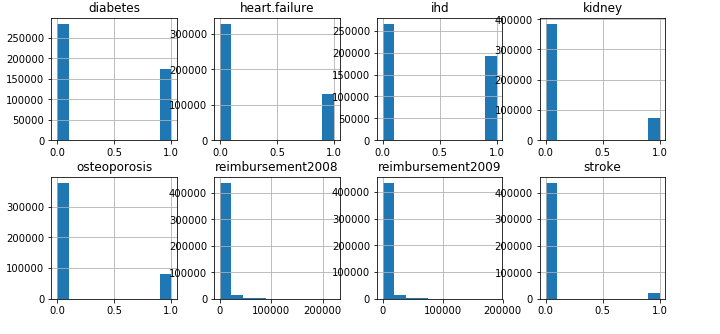
**Exploratory Analysis:**

Lets see the distribution of variables

dataset[dataset.dtypes[(dataset.dtypes=="float64")|(dataset.dtypes=="int64")]

.index.values].hist(figsize=[15,15])





The distribution of the age variable is not normal and all other variables of diseases/conditions are binary variables. Also, the distribution of reimbursement variables are not normal and bucket variable is ordinal categorical variable having category 1 as reimbursement in the range of 0-3000, i.e,.

|  |  |
| --- | --- |
| Reimbursment | Bucket |
| 0-2999 | 1 |
| 3000-7999 | 2 |
| 8000-19999 | 3 |
| 19000-54999 | 4 |
| >55000 | 5 |

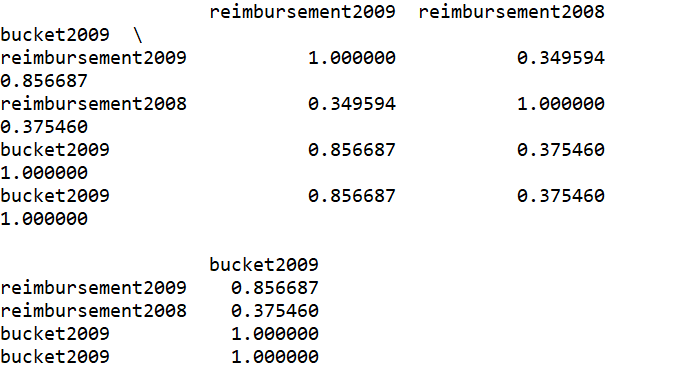
It is evident that all the variables are to be scaled to better predictive the model(as we are using K-Nearest neighbors, Ordinal Logistic regression)

**#Finding if there is any correlation between the variables**

corr= dataset.corr(method='spearman')

dataset[['reimbursement2009', 'reimbursement2008', 'bucket2009', 'bucket2009']].corr(method='pearson')

It seems only the reimbursement2008, reimbursement2009, bucket2008 and bucket2009 have highest correlations as expected.



As there is highest correlation between the pairs (reimbursement2008, bucket2008)

(reimbursement2009, bucket2009) we shall take one among these into the model, but not both. Lets focus on getting the buckets for predicting the cost of patients.

**Variable selection:**

We have the actual buckets for the 2008 and 2009. Lets see if the bukets of 2008 are useful in predicting the buckets of 2009, so that we can use these data to predict the reimbursement buckets of 2010.

So, the y variable is bucket2009 with all others as x variables except reimbursements in 2008 and 2009. Since we are trying to find the bucket we know the range of reimbursement in prediction

#Data Preprocessing

dataset = dataset.drop(['reimbursement2008', 'reimbursement2009'], axis=1)

X = dataset.iloc[:, 0:12].values

y = dataset.iloc[:, 13].values

# Splitting the dataset into the Training set and Test set

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

# Feature Scaling- we should not scale the y-variables here as we want them to discrete

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

X\_train = sc\_X.fit\_transform(X\_train)

X\_test = sc\_X.transform(X\_test)

#from sklearn.preprocessing import scale

#y\_train = scale( y\_train, axis=0, with\_mean=True, with\_std=True, copy=True ) #not required in this case

As the y variables are ordinal and discrete we can use ordinal logistic regression

**#Fitting OrderedLogistic Regression**

from scipy import stats

import mord as m

c = m.OrdinalRidge()

c.fit(X\_train, y\_train)

y\_pred = c.predict(X\_test)

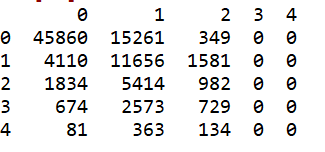
# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

cm = pd.DataFrame(cm)



ac = accuracy\_score(y\_test, y\_pred)

0.6386

**Using K-Nearest Neighbors classification:**

# Fitting K-NN to the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

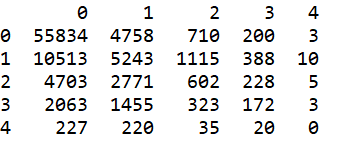
# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

cm = pd.DataFrame(cm)



ac = accuracy\_score(y\_test, y\_pred)

Accuracy = 0.6752