Machine Learning Engineer Nanodegree

Capstone Project Report

PREDICTING INSURANCE COST

MANOJ RAJESH

June 26th,2018

I. Definition

Project Overview

The domain of my proposal data is related medical insurance and I choose this because if an insurance company wants to make money, it should collect the more money in year premiums than that it spends on medical treatment for a beneficiary so, insurers should invest more time to predict the annual cost of an individual and charge a little more than that , as it is a very time taking process to predict each individual cost manually, and as time is the most precious factor in today's life, I am trying to develop a model that predicts estimated cost of an individual based on previous data. And this problem is to be solved because every day number people who are applying for health are increasing, insurance providing companies need a model that makes one of their task easier.

https://www.techemergence.com/machine-learning-at-insurance-companies/

https://cloud.google.com/blog/big-data/2017/03/using-machine-learning-for-insurance-pricing-optimization

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5134202/

Some other machine learning prediction problems similar to mine are:

1. https://archive.ics.uci.edu/ml/datasets/Wine+Quality
2. https://archive.ics.uci.edu/ml/datasets.html
3. https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice

Problem Statement

The problem statement is to predict the future medical expenses of individuals that help medical insurance to make decision on charging the premium and the medical expenses are difficult to estimate because many diseases are rare and random, but we can still predict the estimated that an individual needs in a annual year by observing some of the factors and help insurance company to make decision on charging the premium. And this can be done by applying past data as training data as past data is the data and it contains the exact cost of an Individual that an insurance company spend on them so, by looking and analyzing the features (smoker or not, body mass index, region) of individual of certain cost we can easily predict the future cost of an individual who is nearly with same features.

A brief outline of my solution :

My solution to the problem will look like giving output is insurance cost of an individual. As the problem is regression problem, I am going to use four regression algorithms namely Linear Regressor, Decision Tree Regressor, Adaptive Boosting Classifier and Random forest Regressor. I want to select the best algorithm based on r2 - score (My evaluation metric to measure performance) and then using Grid Search technique, I want to improve the performance of my model to a little bit.

Metrics

The evaluation metric that I am using to measure the performance of our model r2– score . In statistics, the **coefficient of determination**, denoted R^2 or r^2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variables.

It is a statistic used in the context of statistical models whose main purpose is either the prediction of future outcomes or the testing of hypotheses, on the basis of other related information. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

There are several definitions of R^2 that are only sometimes equivalent. One class of such cases includes that of simple linear regression where r^2 is used instead of R^2. When an intercept is included, then r^2 is simply the square of the sample correlation coefficient (i.e., r) between the observed outcomes and the observed predictor values. If additional regressors are included, R^2 is the square of the coefficient of multiple correlations. In both such cases, the coefficient of determination ranges from 0 to 1.

Therefore, I am choosing r2 _score as my metric. In addition to that, I will also check training and testing time.

R2_score =1 - residual sum of square / total sum of squares

II. Analysis

Data Exploration

The dataset that I am working is downloaded from Kaggle.

Dataset URL :
https://www.kaggle.com/mirichoi0218/insurance/data

There are 7 features and nearly 1380 instances of data. The feature that I am interesting is Cost (Target Feature) there are no missing values in the data. I am going to use all this data for my project.

The attributes with different values are as follows:

Age: age of primary beneficiary
Sex: insurance contractor gender, female, male

bmi: Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9

Children: Number of children covered by health insurance / Number of dependents

Smoker: Smoking or not.

Region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

Charges: Individual medical costs billed by health insurance

```
# No. of Attributes and Instances  in the data
print(' Shape of the data :',data.shape)
# Names of Features
print('\nThe features present in the data are :\n\n',data.columns)
# Displaying First five Observations
display(data.head())
# Displaying last five Observations
display(data.tail())
```

```
 Shape of the data : (1338, 7)
The features present in the data are :
 Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

In these data set inputs are sex, age, body mass index, number of children , smoker or not, region of the applying person.

Output is the individual medical cost.

Displaying head and tail of the data:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

	age	sex	bmi	children	smoker	region	charges
1333	50	male	30.97	3	no	northwest	10600.5483
1334	18	female	31.92	0	no	northeast	2205.9808
1335	18	female	36.85	0	no	southeast	1629.8335
1336	21	female	25.80	0	no	southwest	2007.9450
1337	61	female	29.07	0	yes	northwest	29141.3603

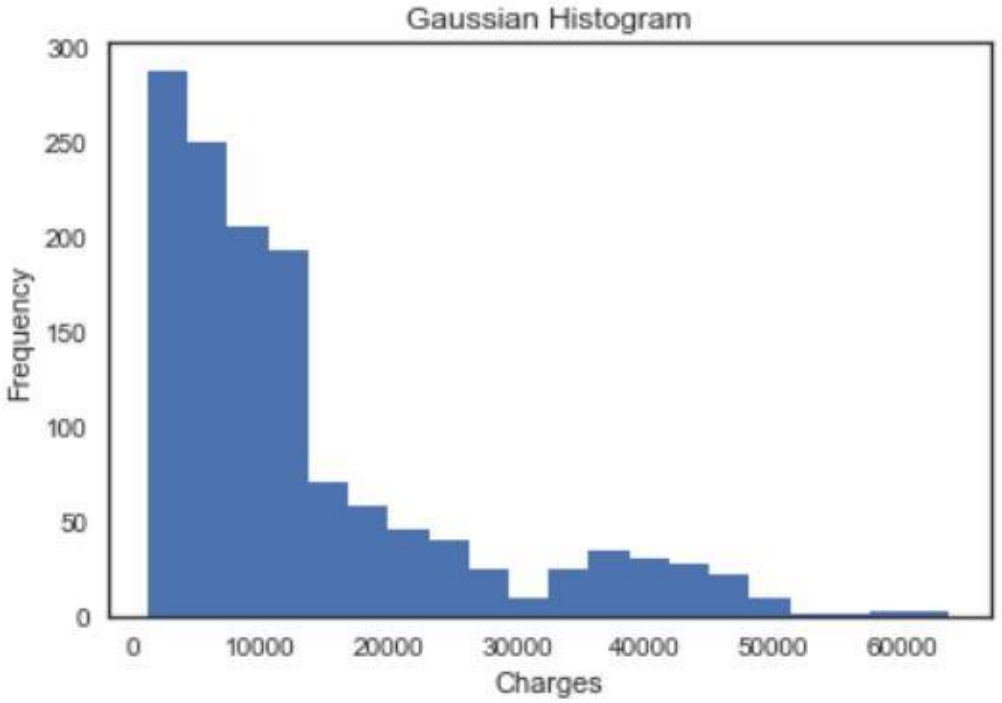
Describing the data:

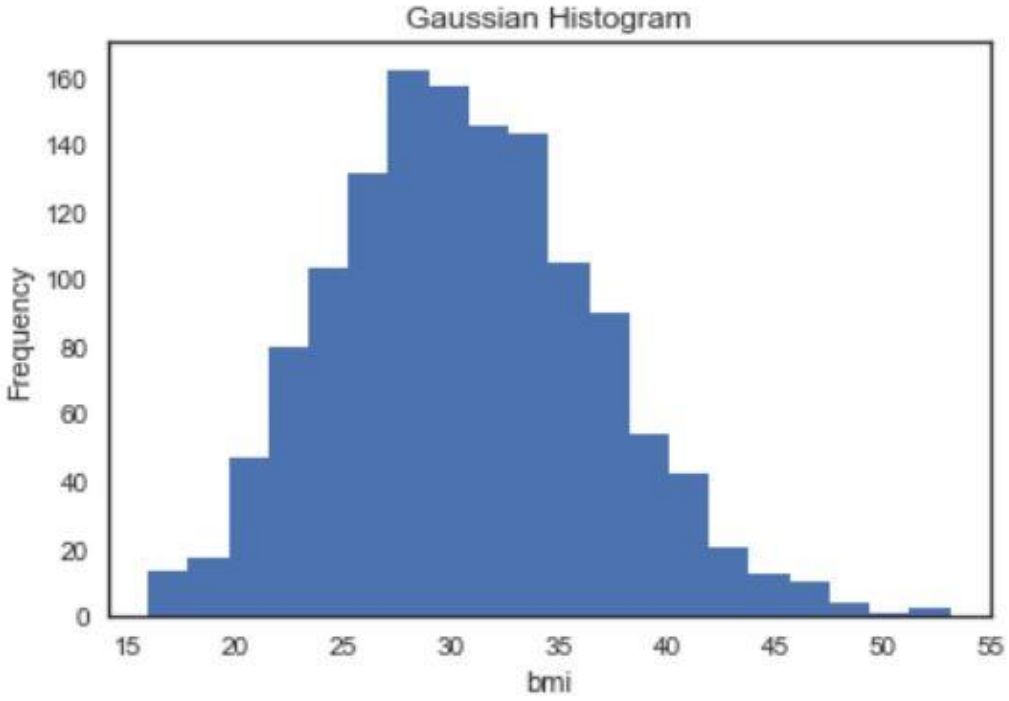
Describing the Data

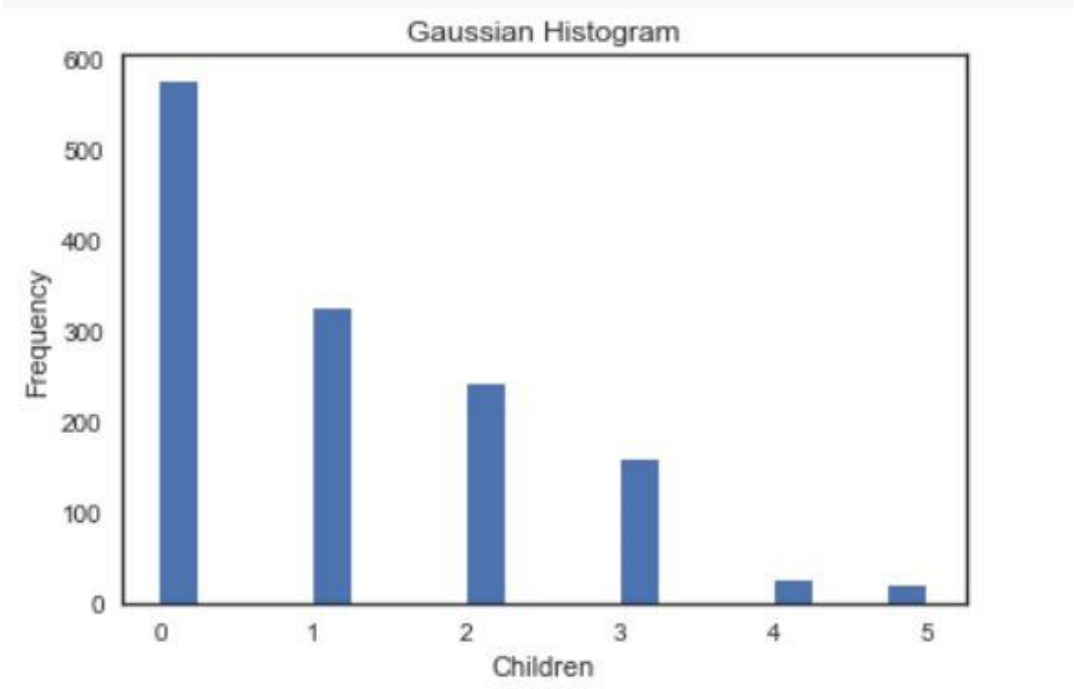
```
# Description of the data to get some insights

data.describe()
```

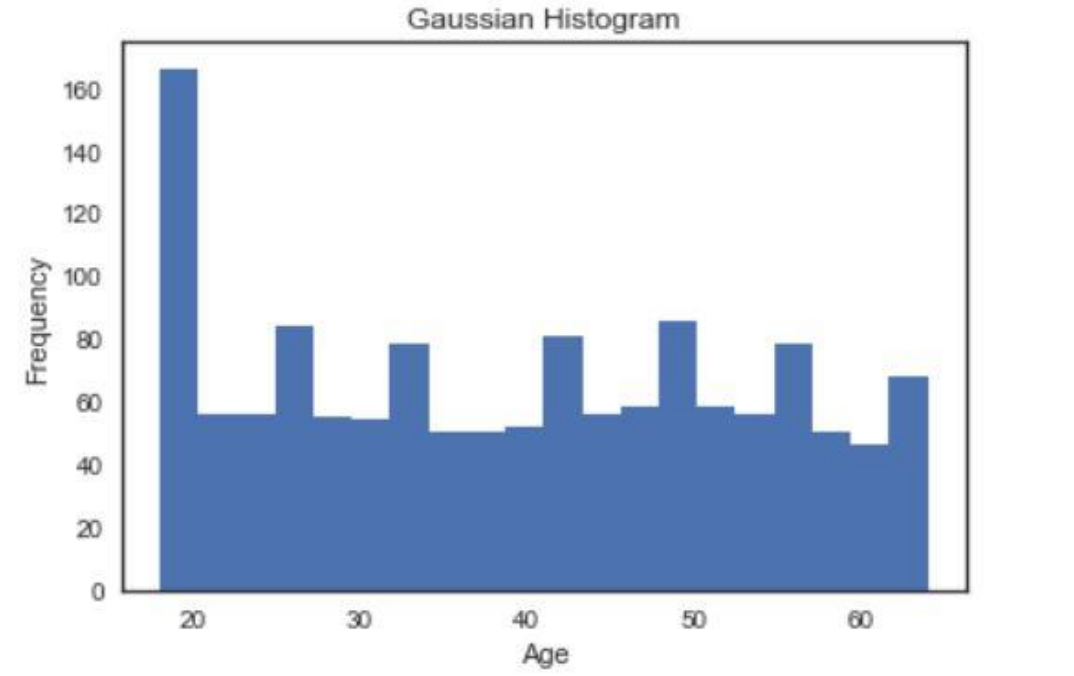
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Plotting Histograms of some Parameters :









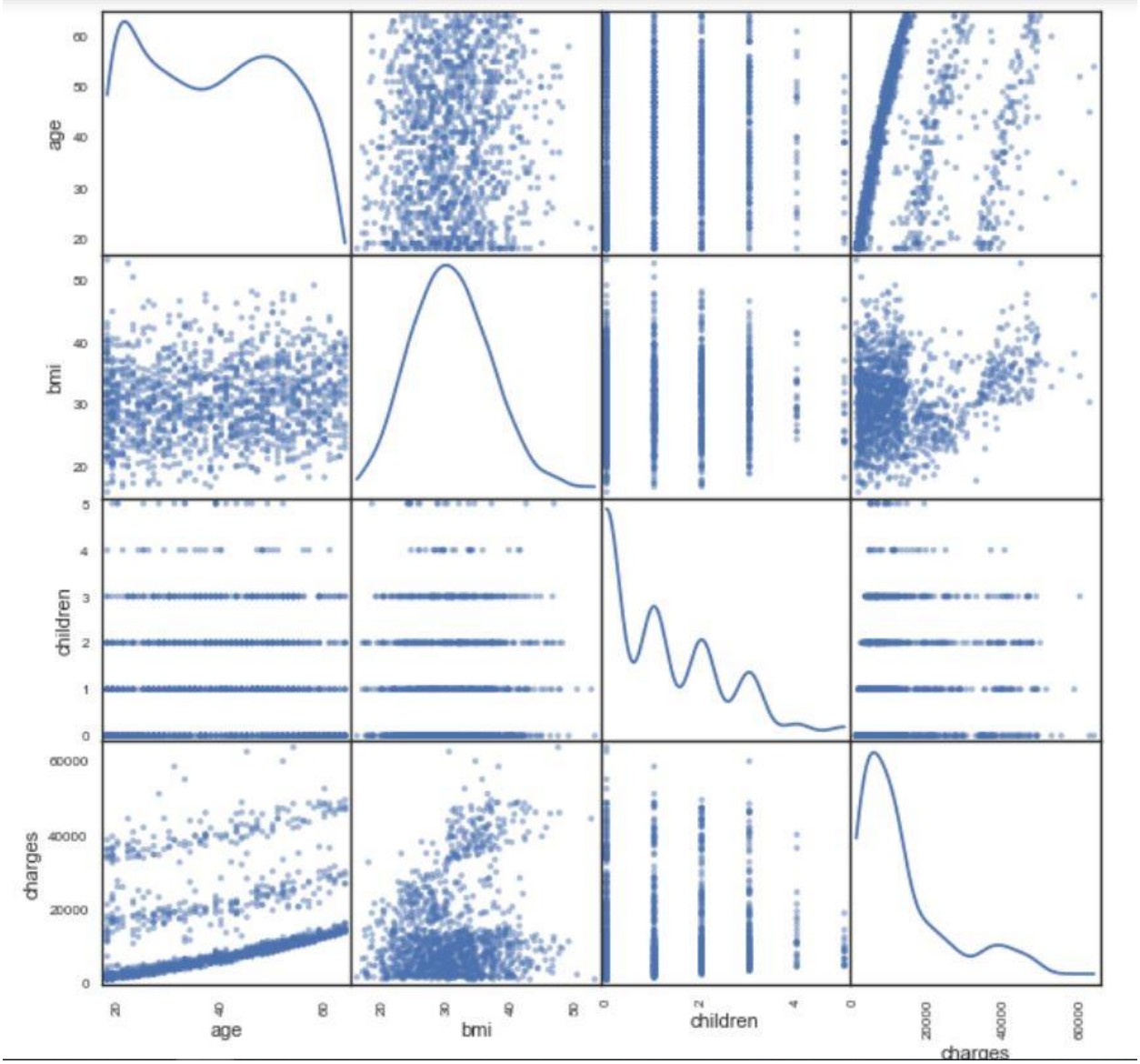
By seeing the above histograms we can say that charges feature is right skewed.

Exploratory Visualization

The below graph shows how my features distributed in the data:

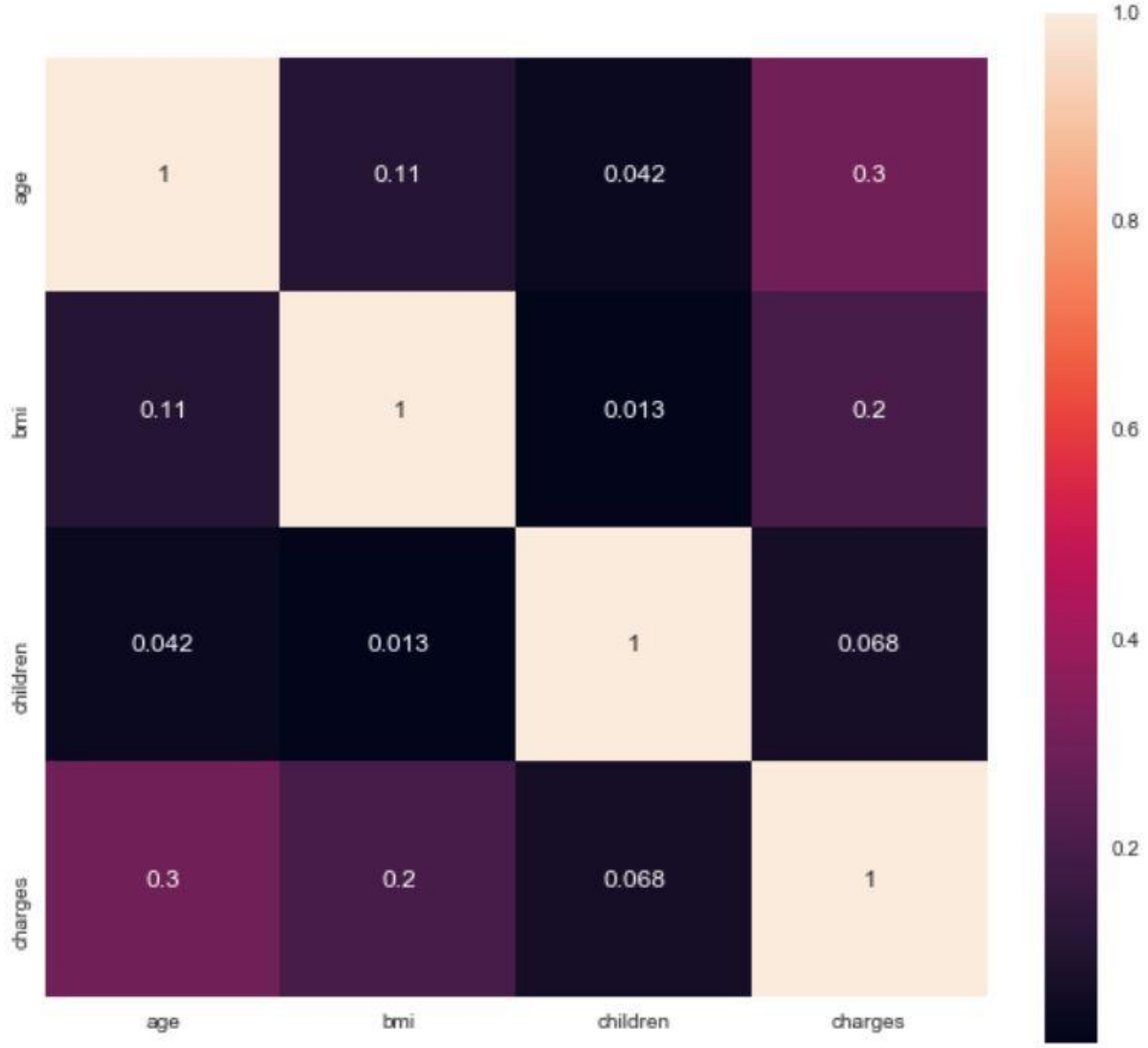
```
# plotting a scatter matrix to know the relationship ( correlation ) between the features
print('Scatter Matrix ')
pd.plotting.scatter_matrix( data, alpha = 0.5 , figsize = ( 10, 10 ) , diagonal = 'kde' );
plt.show()
```

```
Scatter Matrix
```



From above figure we can say that no features are correlated to each other so, every feature is important in predicting the Output.

The below heat map describes how my features are correlated with each other :



By above heatmap the features are not at all correlating with each other. I choose heatmap because it is the easiest way to analyze correlation between attributes.

Algorithms and Techniques

Algorithms:
The problem discussed here is a regression task . So , I choose four Algorithms. Then I compare the performance of each model and select the best one. The four algorithms that I chosen are:

1. Linear Regression :

I choose linear regression model That finds a linear function (a non-vertical straight line) that, as accurately as possible, predicts the dependent variable values as a function of the independent variables. The adjective simple refers to the fact that the outcome variable is related to a single predictor. So ,i assumed it will find the linear relationship between output cost and remaing input features.

2. Decision Tree Regressor :

The Second algorithm that I choose is Decision Tree regressor. The reason behind this is it is very powerful classification algorithm that converts our regression task into tree like structure and it is easy to interpret and understand.This algorithm is relatively easy and fast to predict value of our Target variable.The no. of hypertune parameters are almost null. The disadvantage of this algorithm is also overfitting which can be solved by Ensemble learning algorithms.

3. Adaptive Boost Regressor :

The final algorithm that I choose is Adaptive Boosting regressor. It belongs to the family of Ensemble learners. It is one of the most powerful and efficient algorithm which boosts the performance of a model by combining a set of weak learners into a single strong learner. This algorithm is less prone to Over-Fitting because it will allocate weights to set of weak learners which then contribute to strong learner.

4. Random forest Regressor:

Random forest Regressor is an ensemble learning method regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.Random decision forests correct for decision trees' habit of overfitting to their training set. So, i choosed this as my fourth algorithm.

Techniques:

The technique that I am going to use to improve the performance of model is Grid Search.

Grid Search:

I am using Grid Search technique is to tune hyper parameters because tuning hyper parameters is very important part in improving performance of the model. Finding best hyper parameters is a difficult task. So, we give a set of values to the parameters which are to be tuned and give this space to Grid Search algorithm. It will allocate each value in the space at a time and returns the best parameters.

Benchmark:

First I will use linear regression as my benchmark model and calculate it’s r2-score to test the performance of the model. I got around 71% r2-score which is less hence I implement another model and compare its r2-score to test its performance compared to Benchmark model. So by implementing other models we will get a better r2-score than Benchmark model.

III. Methodology

Data Preprocessing

The preprocessing steps I took are:

1.Converting the categorical data into numerical data

Converting categorical data into numbers

```
In [199]: data['sex'] = data['sex'].apply(lambda x: 1 if x == 'male' else 0)

          data['smoker'] = data['smoker'].apply(lambda x: 1 if x == 'yes' else 0)
          data['region'].value_counts()

Out[199]: southeast    364
          northwest    325
          southwest    325
          northeast    324
          Name: region, dtype: int64
```

```
In [200]: data['region'].dtype
          data['region'] = pd.factorize(data['region'])[0]
          data['region'].value_counts()

Out[200]: 1    364
          2    325
          0    325
          3    324
          Name: region, dtype: int64
```

2. Describing the converted data.

In [201]:
```
data.head()
data.tail()
```

Out[201]:

	age	sex	bmi	children	smoker	region	charges
1333	50	1	30.97	3	0	2	10600.5483
1334	18	0	31.92	0	0	3	2205.9808
1335	18	0	36.85	0	0	1	1629.8335
1336	21	0	25.80	0	0	0	2007.9450
1337	61	0	29.07	0	1	2	29141.3603

Describing Converted Data

In [202]:
```
data.describe()
```

Out[202]:

	age	sex	bmi	children	smoker	region	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.204783	1.484305	13270.422265
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.104885	12110.011237
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900
25%	27.000000	0.000000	26.296250	0.000000	0.000000	1.000000	4740.287150
50%	39.000000	1.000000	30.400000	1.000000	0.000000	1.000000	9382.033000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	2.000000	16639.912515
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010

3. Split the data into training and testing data using train_test_split function in sklearn.

Splitting the total data into training and Testing data

```
# Dividing the taken data into training and testing set


X_train , X_test , y_train , y_test = train_test_split( X , y , test_size = 0.2 , random_state = 29 )

print(X_train.shape )
print(X_test.shape )
```

```
(1070, 6)
(268, 6)
```

There 1070 records are used for taraining the model and 268 are used for testing the model.

4. Finding the outliers.

Outlier Treatment

```
# Identifying outliers in each attribute

for feature in data.keys():

    Q1 = np.percentile(data[feature], q=25)

    Q3 = np.percentile(data[feature], q=75)

    IQR = Q3 - Q1
    step = 1.5 * IQR

    print("Outliers for the feature '{}':".format(feature))
    display( data[~((data[feature] >= Q1 - step) & (data[feature] <= Q3 + step))] )
```

I did not remove the outliers even though I found them because they contain the some important data.

Implementation:

I choose implement the following four algorithms:

1. Linear Regression

2. Decision Tree Regressor

3. Ada Boost Regressor

4. RandomForestClassifier

After I calculated the r2-scores and training and testing time of these four models to check their performance. By fitting the data into the classifiers and calculated the r2-score of the four models using r2-score function from sklearn.metrics.

Benchmark Model (Linear Regression)

I am using linear regression as my benchmark model. After I calculated the r2-scores and training and testing time of this model to check performance.

Performance of Benchmark model

```
model = linear_model.LinearRegression()

performance(model,X_train,y_train,X_test,y_test)
```

```
Training time : 0.8324148654937744
Testing time :  0.1319603919982910 2
r2_score is0.71
```

For my benchmark model I r2_score of 0.72.

For remaining models using different algorithms we got below r2_score and training and testing time.

```
Performance of DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           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,
           presort=False, random_state=None, splitter='best')


Training time : 0.020961761474609375
Testing time :  0.000997781753540039
r2_score is0.74


Performance of AdaBoostRegressor(base_estimator=None, learning_rate=1.0, loss='linear',
         n_estimators=50, random_state=None)


Training time : 0.03699898719787598
Testing time :  0.04393315315246582
r2_score is0.79


Performance of RandomForestRegressor(bootstrap=True, criterion='mse', 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=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)


Training time : 0.04459095001220703
Testing time :  0.0009996891021728516
r2_score is0.81
```

I choose Random Forest Regressor as my solution model as it has high r2-score compared to remaining three models. The initial model has a r2_ score value of 71% whereas DTR has a r2_score of 74% which is quite well compared to Linear Regression then I got r2_score of 79% for Ada boost Regressor. Finally I got an r2_ score of around 81% by using RFR.

Refinement:

I used Grid search technique to refine my solution model. But the initial(before tuning) and final solutions(after tuning) are changed i.e., I got an r2_score of 81% before tuning and after tuning also I got the 84%.

Refinement

Hyper Parameter Tuning using Grid Search

```
    regressor = RFR()
    scoring_fnc = make_scorer(per_metric)
    param_grid = {'n_estimators': [90,130],
                  'max_depth': [4,5],
                  'min_samples_leaf':[1,5],
                  'min_samples_split' :[3,5],
                }
    grid = GridSearchCV(estimator=regressor,param_grid=param_grid,scoring = scoring_fnc)
    grids = grid.fit(X_train, y_train)

    # Return the optimal model after fitting the data
    best=grids.best_estimator_
    print(grids.best_estimator_)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=4,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=5, min_samples_split=5,
           min_weight_fraction_leaf=0.0, n_estimators=130, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

IV. Results

Model Evaluation and Validation:

Finally our solution got an r2_score of 84% which is pretty good compared to our benchmark model.

Performance of the improved model.

Checking performance of the Optimized model

```
best.fit(X_train , y_train)     # Calculating the training time
y_preds = best.predict(X_test)
grid = GridSearchCV(estimator=regressor,param_grid=param_grid,scoring = scoring_fnc)
best=grids.best_estimator_
print('r2_score is :{:.2f}'.format(r2_score(y_test, y_preds)))
```

```
r2_score is :0.84
```

Justification:

Performance of Benchmark model

```
model = linear_model.LinearRegression()

performance(model,X_train,y_train,X_test,y_test)
```

```
Training time : 0.8324148654937744
Testing time :  0.13196039199829102
r2_score is0.71
```

Checking performance of the Optimized model

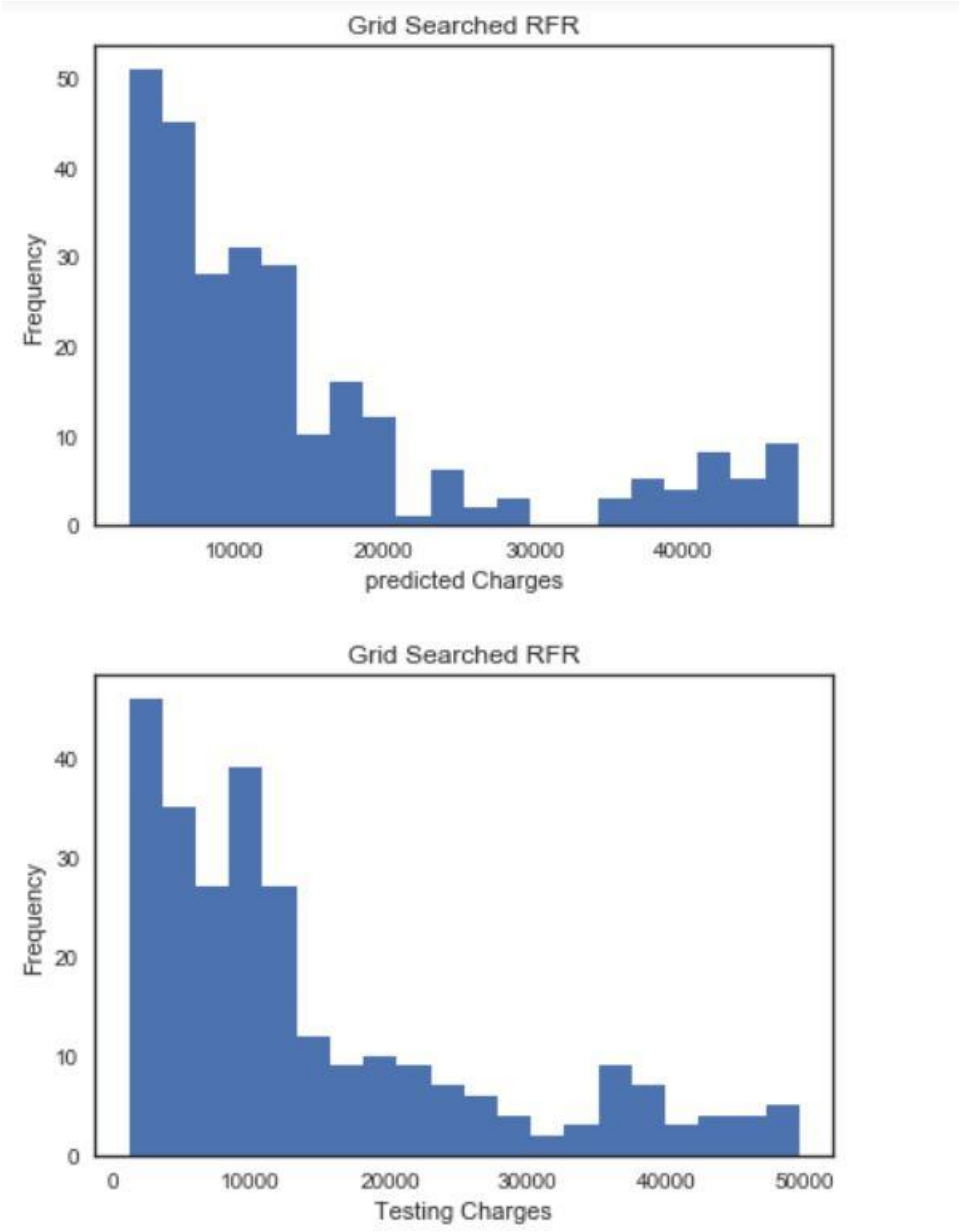
```
best.fit(X_train , y_train)     # Calculating the training time
y_preds = best.predict(X_test)
grid = GridSearchCV(estimator=regressor,param_grid=param_grid,scoring = scoring_fnc)
best=grids.best_estimator_
print('r2_score is :{:.2f}'.format(r2_score(y_test, y_preds)))
```

```
r2_score is :0.84
```

By seeing the r2_score values of our four models we can say that our solution model works pretty well than the benchmark model and the other model. It's r2_score value is pretty high compared to remaining three models and it also takes less hyper parameters to solve the problem. So, Random Forest Regressor is my final Model.

Conclusion

Visualization of Charges predicted and Testing Charges.



We can see in the above graphs nearly similar and have some errors. Hence if we have more data we can get better results compared to the current results.

Predicting Charges of some persons with Random Data.

Now i would like to predict some peoples insurane cost based on some input factors

- Age
- Sex{ male-> 1, Female ->0}
- Bmi
- Children
- Smoker { yes->1 , no ->0)
- Region {southeast -> 1 , northwest -> 2 ,southwest -> 0 ,northeast -> 3 }

```
def opfun(X_new):
    best.fit(X , y)
    y_new = best.predict(X_new)
    return y_new
```

```
X_new=[[30,1,28.9,2,0,2],[29,0,30.0,1,1,3],[30,0,28.9,2,1,2],[22,0,28.9,1,0,2]] # some random data
for i, price in enumerate(opfun(X_new)):
    print("Predicted Insurance cost of Client {} : ${:,.2f}".format(i+1, price))
```

```
Predicted Insurance cost of Client 1 : $6,537.21
Predicted Insurance cost of Client 2 : $24,592.61
Predicted Insurance cost of Client 3 : $19,739.13
Predicted Insurance cost of Client 4 : $4,158.00
```

Reflection

- First , I downloaded the data from Kaggle and read the data using Pandas Data Frame.

- Then I defined my problem statement as to predict the insurance cost of an individual.

- Then I found the no. of instances and attributes present in the data at the Data exploration step.

- Then I performed Bi-variant analysis to see whether the features are correlated or not. Almost no features are correlated.

- At the Data Pre-processing step , I tried to find missing values but there were no missing values. Then I tried to find out Outliers in the data and I not removed outliers in the data.

- Then I defined a Benchmark model which will predict the insurance cost but it gave me bad performance as expected.

- So, I choose three algorithms Decision Tree and Ada Boost Classifier and Random Forest Regressor out of which I choose Random Forest Regressor with proper reasoning.

- I tried to improve the performance of Random Forest Regressor with Grid Search technique (Hyper parameter optimization) and r2_score was improved.

- Overall, I had a great experience with the project and I enjoyed doing the project very much.

Improvement:

- I think the results can be improved if I have more balanced data, then automatically the performance of models will increase after parameter tuning.