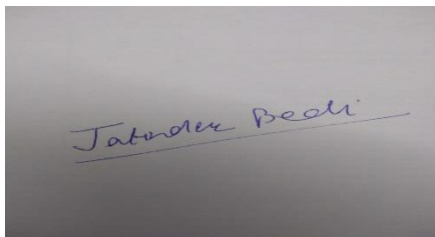

Project Summary

Batch details	PGPDSE-FT Online June21-A
Team members	Beryl David, Kathyayani Sayana, Priyadharsini, Tharun Prabhakar, Visharad
Domain of Project	Finance and Risk Analytics
Proposed project title	Medical Insurance Premium Prediction
Group Number	7
Team Leader	Priyadharsini
Mentor Name	Mr. Jatinder Bedi

Date: 17/03/2022

A photograph of a handwritten signature in blue ink on a white piece of paper. The signature reads "Jatinder Bedi" and is underlined.

Signature of the Mentor

Priyadharsini.N

Signature of the Team Leader

Table of Contents

Sl NO	Topic	Page No
1	Overview	3
2	Problem Statement (Business problem goals)	3
3	Abstract	4
4	Data Description and pre processing	5
5	Data Preperation	8
6	Exploratory Data Analysis & Business Insights	8
7	Basic Model	10
8	Feature engineering & feature extraction	10
9	Hyper parameters tuning	11
10	Comparison and selection of model	
11	Results & Discussion	
12	Description of criterion	

1.OVERVIEW

The healthcare market can increase three-fold to Rs. 8.6 trillion (US\$ 133.44 billion) by 2022[1], and everyone needs some level of health care. Health insurance is one of the most significant investment an individual makes every year.

The healthcare premiums keep changing every year because of various factors [2] such as medical trends, pharmaceutical trends, and political factors etc. over which the customer has no control. The only option an individual can have is to plan carefully for future expenses.

one of the important tasks for health insurance companies is to determine the policy premiums. By using predictive modelling, the insurers can determine the policy premium for the insured based on their behaviors which are indicated by attributes. [3].

Machine Learning models are helpful here and make it more efficient for the insurance companies to predict the insurance premium amount for a customer based on the personal health data of every individual. We will analyze the data using Exploratory Data Analytics and try to predict the medical costs of the individuals using several Machine Learning Models such as Random Forest, Linear Regression, etc., and compare the results and come up with the best approach.

2.PROBLEM STATEMENT (BUSINESS PROBLEM GOALS)

1. Business Problem Understanding

- Predicting the Medical Insurance Premium has a significant importance in the insurance sector. predict the insurance charges of a person and identify those patients with their medical details whether they have any health issues or not. Machine Learning models help to predict the premium amount for an individual based on their health data.

2. Business Objective

- Predict the insurance premiums based on the Health-Related Parameters of the Customers collected from the individuals so that insurance companies can make useful and accurate predictions.

3. Approach

- Approach starts from understanding the dataset
- Perform the basic preprocessing like missing value treatment and outlier treatment
- Building Model by applying Machine Learning algorithms (like Linear Regression, Random Forest) to predict the premium cost.
- Evaluating and fine tuning the model to get optimum accuracy.

4. Conclusions

- Building different model for Predicting the premium based on medical conditions and compare the results for the correctly anticipate insurance policy costs

3.ABSTRACT

The main foundational block of health insurance industry is to estimate the future events and measure the associated risk/value of these events, hence it is needless to say that predictive analytics is used widely to determine the risk, insurance premium and enrich overall customer experience.

The health insurance industry has always been a slow-moving industry when it comes to adopting the data analytics practices into its business models. With the advent of advanced data analytics technologies, it has become important more than ever to take advantage of such sophisticated analytics to accurately assess and predict the insurance premiums for the insured.

Thus, one of the important tasks for health insurance companies is to determine the policy premiums. By using predictive modelling, the insurers can determine the policy premium for the insured based on their behaviors which are indicated by attributes like age.

This determination of premiums based on the data collected for an individual medical condition helps insurance companies in enhanced pricing, underwriting and risk selection. Additionally, it helps in making better decisions, understanding customer needs and be fair to the customers. Acquiring a comprehensive understanding of customer behaviors and habits from historical data helps insurers to anticipate future behaviors and provide the right insurance product and policy premium

4.Data Descriprion and pre processing

- Contains 986 rows and 11 columns
- Numerical features Age, Height, Weight, Premium Price
- Categorical features – Diabetes, BP, Any Transplants, Any Chronic Diseases, Known Allergies, History of Cancer in Family, Number of Major Surgeries
- New feature BMI is added for a better model

```
1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 986 entries, 0 to 985
Data columns (total 11 columns):
#   Column                                Non-Null Count  Dtype  
---  -
0   Age                                    986 non-null    int64  
1   Diabetes                              986 non-null    int64  
2   BloodPressureProblems                 986 non-null    int64  
3   AnyTransplants                        986 non-null    int64  
4   AnyChronicDiseases                    986 non-null    int64  
5   Height                                986 non-null    int64  
6   Weight                                986 non-null    int64  
7   KnownAllergies                        986 non-null    int64  
8   HistoryOfCancerInFamily               986 non-null    int64  
9   NumberOfMajorSurgeries                986 non-null    int64  
10  PremiumPrice                          986 non-null    int64  
dtypes: int64(11)
memory usage: 84.9 KB
```

We can observe that there are 986 records with 11 columns

Numeric Variables

```
In [16]: 1 df[num].describe()

Out[16]:
```

	Age	Height	Weight	PremiumPrice	BMI
count	986.000000	986.000000	986.000000	986.000000	986.000000
mean	41.745436	168.182556	76.950304	24336.713996	27.460709
std	13.963371	10.098155	14.265096	6248.184382	5.878671
min	18.000000	145.000000	51.000000	15000.000000	15.156281
25%	30.000000	161.000000	67.000000	21000.000000	23.393392
50%	42.000000	168.000000	75.000000	23000.000000	27.156602
75%	53.000000	176.000000	87.000000	28000.000000	30.759870
max	66.000000	188.000000	132.000000	40000.000000	50.000000

The maximum age of the customer is 66 and the minimum age is 18. Mean age is 41.75 years.
The maximum height of the customer is 188cm and the minimum height is 145 cm. Average height is 168 cm.
The maximum weight of the customer is 132 kgs and the minimum weight is 51 kgs. Average weight is 76.95kgs.
The maximum premium price is 40000 and the minimum premium price is 15000. Average is 24336.
The maximum BMI is 50 and the minimum BMI is 15.15. Average is 27.46.

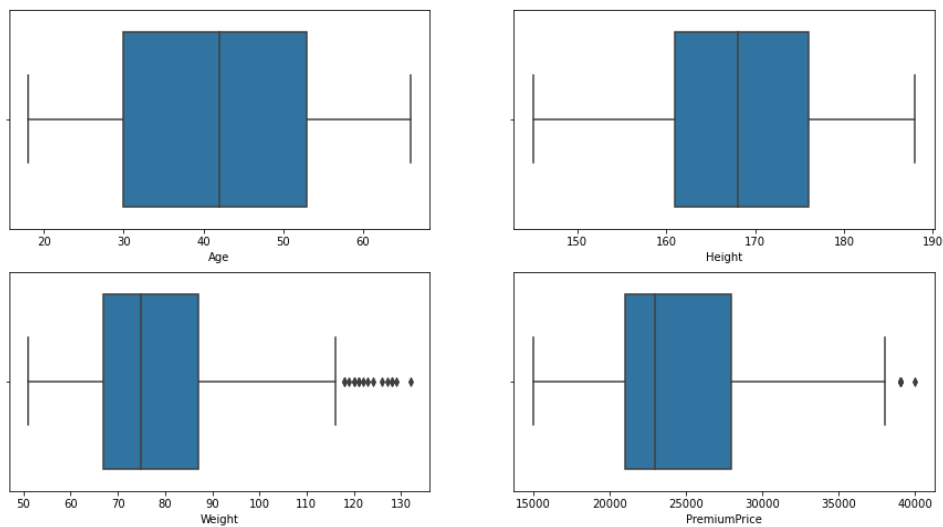
Data Preparation:

1. Check for outliers:

```
In [620]: cols=df[['Age','Height','Weight','PremiumPrice']].columns
          cols
```

```
Out[620]: Index(['Age', 'Height', 'Weight', 'PremiumPrice'], dtype='object')
```

```
In [621]: fig,ax=plt.subplots(nrows=2,ncols=2)
          for i,j in zip(cols,ax.flatten()):
              sns.boxplot(df[i],ax=j)
          plt.show()
```



2. Scaling the numerical data

```
In [622]: from sklearn.preprocessing import StandardScaler
          sc=StandardScaler()
```

```
In [623]: df2=pd.DataFrame(sc.fit_transform(df[cols]),columns=df[cols].columns)
          df2.head()
```

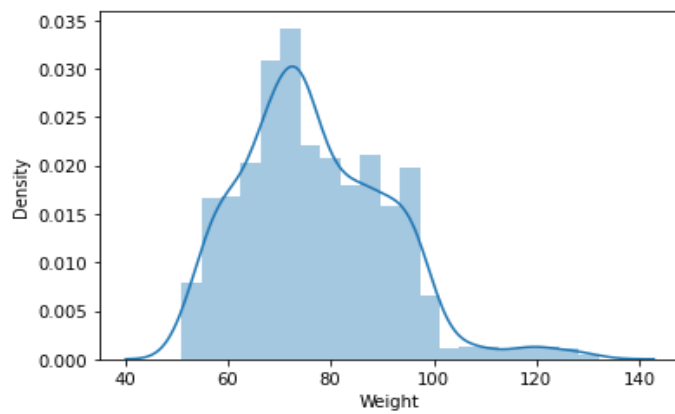
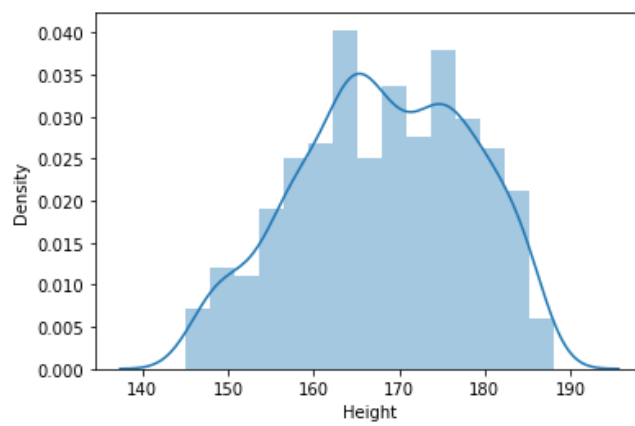
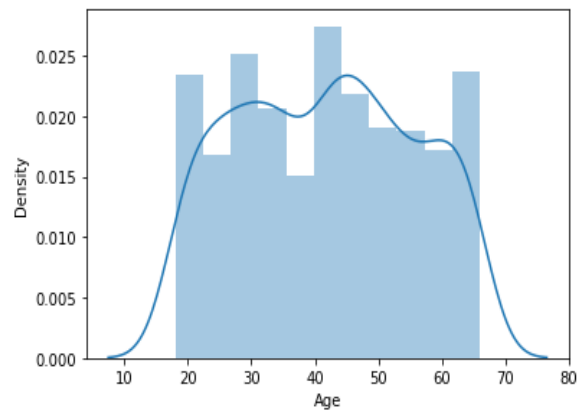
```
Out[623]:
```

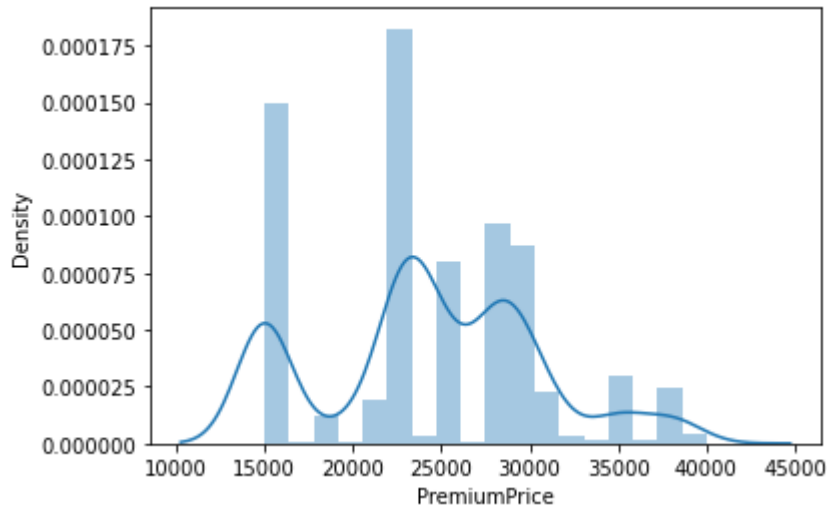
	Age	Height	Weight	PremiumPrice
0	0.233197	-1.306105	-1.399250	0.106210
1	1.307981	1.170852	-0.277062	0.746721
2	-0.411674	-1.008870	-1.258976	-0.214045
3	0.734763	1.468086	1.125674	0.586594
4	-0.268369	-0.216244	0.774990	-0.214045

Exploratory Data Analysis & Business Insights

Univariate Analysis:

Numerical Data:

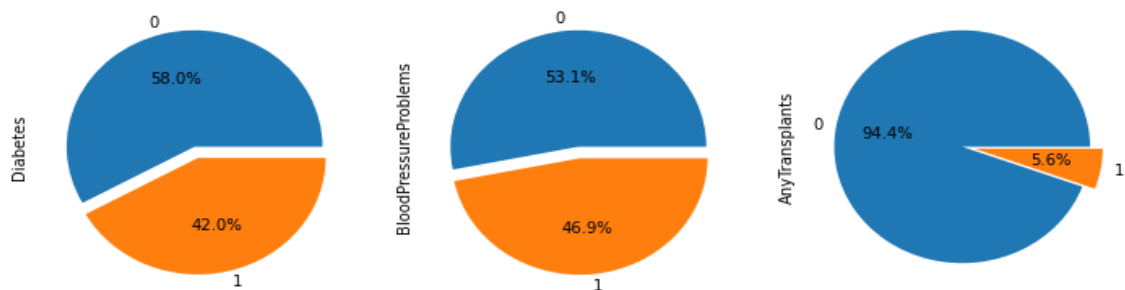


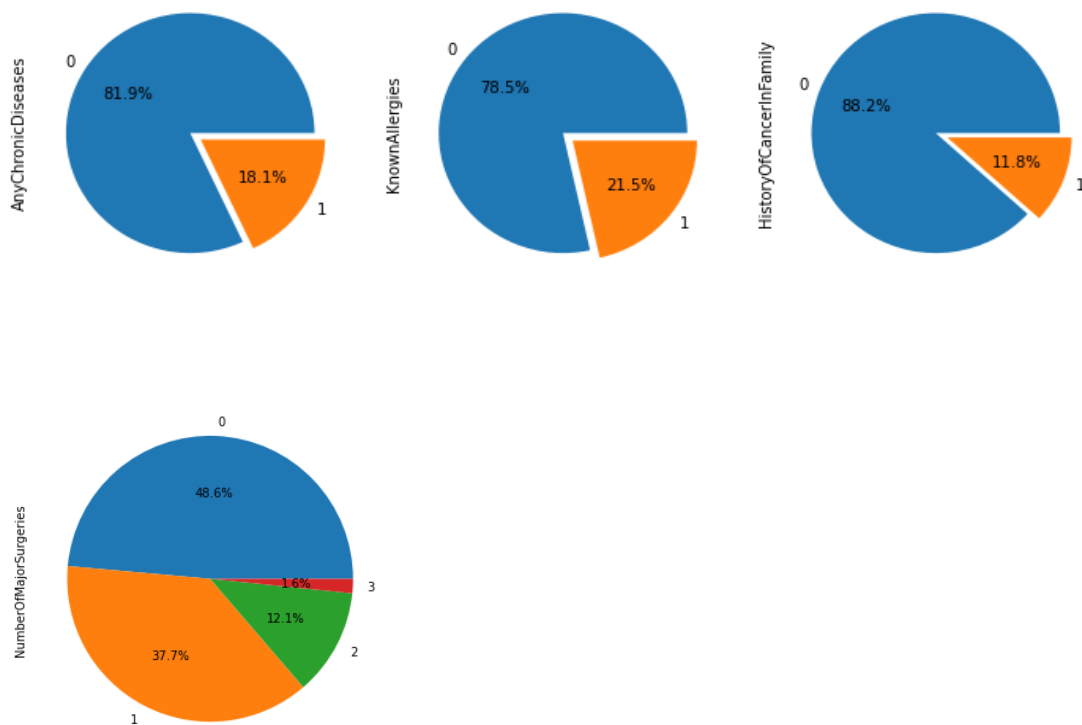


INFERENCES:

- The maximum age of the customer is 66 and the minimum age is 18. Mean age is 42 years.
- The maximum height of the customer is 188cm and the minimum height is 145 cm. Average height is 168 cm.
- The maximum weight of the customer is 132 kgs and the minimum weight is 51 kgs. Average weight is 77 kgs.
- The maximum premium price is 40000 and the minimum premium price is 15000. Average is 24336.
- The maximum BMI is 50 and the minimum BMI is 15.15. Average is 27.46.

Categorical Data:

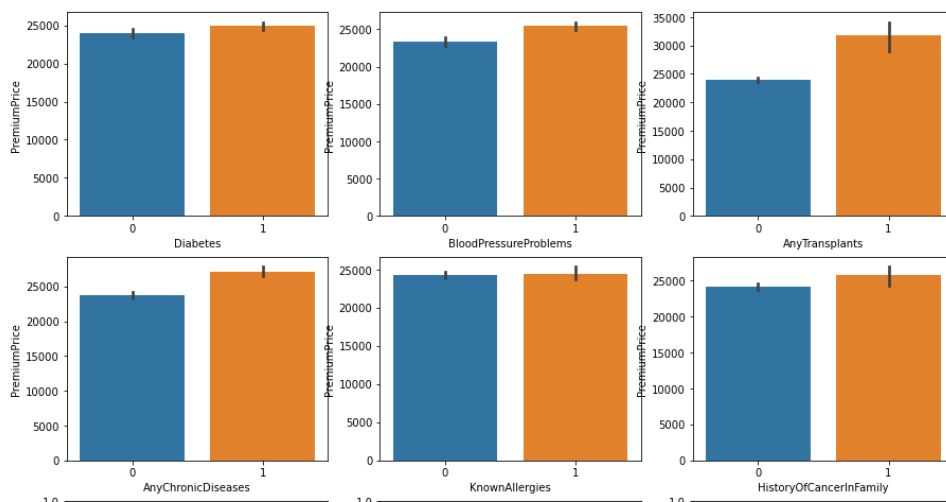




INFERENCES:

- There is a small difference in number of Diabetic (42%) and non-Diabetic people (58%).
- There is a very little difference in number of people with BP (47%) and without BP (53%).
- There are only 6% people had gone through any transplants and 18% of people who had any chronic diseases.
- There are 22% people with known allergies and 12% of people with History of Cancer in family.

Bivariate Analysis:



INFERENCES:

- People with Health Problems such as Diabetes, Blood Pressure, any other Chronic Diseases and any known allergies pay a little more Premium than the people who do not have.
- People who had any transplants in the past pay more premium than those who did not.
- People with history of cancer in family also pay more premium price than those who do not.
- Known Allergies column has no much impact on the Premium Price paid by the customers.

Correlation:



INFERENCES:

- People with Health problems and having some medical history pay higher premium price
- Age is the more significant factor in determining the Premium Price
- Also, Chronic Diseases, transplants, Number of major surgeries influence the target variable

Basic Model:

The target variable is Continuous variable.

Hence we built Linear Regression as our base model

```
In [631]: SLR_model = sm.OLS(y_train_slr, X_train_slr).fit()  
SLR_model.summary()
```

Out[631]: OLS Regression Results

Dep. Variable:	PremiumPrice	R-squared:	0.660			
Model:	OLS	Adj. R-squared:	0.655			
Method:	Least Squares	F-statistic:	131.6			
Date:	Thu, 17 Mar 2022	Prob (F-statistic):	1.24e-151			
Time:	16:30:30	Log-Likelihood:	-608.36			
No. Observations:	690	AIC:	1239.			
Df Residuals:	679	BIC:	1289.			
Df Model:	10					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	-0.1351	0.044	-3.060	0.002	-0.222	-0.048
Age	0.7418	0.025	29.158	0.000	0.692	0.792
Diabetes	-0.0660	0.048	-1.387	0.166	-0.159	0.027
BloodPressureProblems	0.0270	0.047	0.569	0.570	-0.066	0.120
AnyTransplants	1.2429	0.097	12.750	0.000	1.051	1.434
AnyChronicDiseases	0.4547	0.057	7.910	0.000	0.342	0.568
Height	0.0027	0.023	0.118	0.906	-0.042	0.047
Weight	0.1559	0.023	6.813	0.000	0.111	0.201
KnownAllergies	0.0669	0.056	1.203	0.229	-0.042	0.176
HistoryOfCancerInFamily	0.3830	0.070	5.435	0.000	0.245	0.521
NumberOfMajorSurgeries	-0.1035	0.035	-2.941	0.003	-0.173	-0.034
Omnibus:	154.664	Durbin-Watson:	2.047			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	833.383			
Skew:	0.889	Prob(JB):	1.08e-181			
Kurtosis:	8.082	Cond. No.	6.55			

INFERENCES (Base Model):

- 1.From model we can see that age has the most positive impact
- 2.Height has the least impact

3. The Durbin Watson number is close to 2 indicating no autocorrelation
4. Condition Number (CN) is less than 1000 which means there is no multicollinearity
5. No particular pattern has been seen implies there is linearity present in the data
6. From Breusch-Pagan test we can see that $p\text{-value} > 0.05$. There is no heteroscedasticity present in the data i.e. the residuals have equal variance
7. From qqplot and Shapiro we can see that the residuals do not follow normal
8. The R^2 value is 0.614. Thus, we conclude that the 61.4% variation in the PremiumPrice is explained by the model.

Error Terms:

- MSE – (0.38, 0.28)
- RMSE – (0.62, 0.53)
- MAE – (0.44, 0.39)

Feature engineering & feature extraction:

1. We created BMI column as a feature with the column of Height and weight

Calculate BMI from the Height and Weight columns and add new column BMI to the dataset

```
2]: df['BMI'] = df['Weight']/((df['Height']/100)**2)
df.head()
```

sureProblems	AnyTransplants	AnyChronicDiseases	Height	Weight	KnownAllergies	HistoryOfCancerInFamily	NumberOfMajorSurgeries	PremiumPrice	BMI
0	0	0	155	57	0	0	0	25000	23.725288
0	0	0	180	73	0	0	0	29000	22.530864
1	0	0	158	59	0	0	1	23000	23.634033
1	0	1	183	93	0	0	2	28000	27.770313
0	0	1	166	88	0	0	1	23000	31.934969

We created BMI column as a feature with the column of Height and weight

Formula:

$$\text{bmi} = \text{df}[\text{'Weight'}]/((\text{df}[\text{'Height'}]/100)**2)$$

2. We use Forward Selection for selecting best features

```
In [645]: linreg=LinearRegression()
fwd=sfs(estimator=linreg,k_features='best',forward=False,cv=5,verbose=2,scoring='r2')
fwd_model=fwd.fit(X_train,y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   0.0s remaining:   0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:   0.1s finished

[2022-03-17 16:30:30] Features: 10/1 -- score: 0.6502128492097415[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   0.0s remaining:   0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:   0.0s finished

[2022-03-17 16:30:31] Features: 9/1 -- score: 0.6513488154771864[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   0.0s remaining:   0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:   0.0s finished

[2022-03-17 16:30:31] Features: 8/1 -- score: 0.6513578397747756[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   0.0s remaining:   0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:   0.0s finished

[2022-03-17 16:30:31] Features: 7/1 -- score: 0.6513578397747756[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   0.0s remaining:   0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:   0.0s finished
```

```
In [646]: fwd_model.k_feature_names_
```

```
Out[646]: ('Age',
'AnyTransplants',
'AnyChronicDiseases',
'Weight',
'KnownAllergies',
'HistoryOfCancerInFamily',
'NumberOfMajorSurgeries')
```

Selected Features:

```
'Age',
'AnyTransplants',
'AnyChronicDiseases',
'Weight',
'KnownAllergies',
'HistoryOfCancerInFamily',
'NumberOfMajorSurgeries'
```

Model Fitting after feature selection:

```
In [652]: result_table = result_table.append(MLR_full_model_metrics, ignore_index = True)
result_table
```

```
Out[652]:
```

	Model_Name	R-squared:train	R-squared:test	Adj.R-squared	MSE train	MSE test	RMSE train	RMSE test	MAE	MAPE
0	MLR Full Model	0.659725	0.600709	0.594231	0.341500	0.395200	0.584400	0.628600	0.439066	1.136038
1	MLR Model FWD	0.658665	0.599367	0.594660	0.342500	0.396600	0.585200	0.629800	0.440398	1.132491

Hyper parameters tuning

1.We build model with parameters:

```
In [672]: result_table = result_table.append(MLR_full_model_metrics, ignore_index = True)
result_table
```

Out[672]:

	Model_Name	R-squared:train	R-squared:test	Adj. R-squared	MSE train	MSE test	RMSE train	RMSE test	MAE	MAPE
0	MLR Full Model	0.659725	0.600709	0.594231	0.341500	0.395200	0.584400	0.628600	0.439066	1.136038
1	MLR Model FWD	0.658665	0.599367	0.594660	0.342500	0.396600	0.585200	0.629800	0.440398	1.132491
2	GD model	0.582790	0.582790	0.578508	0.360100	0.413000	0.600100	0.642700	0.452824	1.125142
3	Ridge (alpha=1)	0.659660	0.600624	0.594144	0.341500	0.395300	0.584400	0.628700	0.439555	1.133520
4	Ridge (alpha=2)	0.659477	0.600435	0.593952	0.341700	0.395500	0.584600	0.628900	0.440080	1.131099

2.After Hyper parameter Tuning with different models:

```
In [748]: result_table = result_table.append(m, ignore_index = True)
result_table
```

Out[748]:

	Model_Name	R-squared:train	R-squared:test	Adj. R-squared	MSE train	MSE test	RMSE train	RMSE test	MAE	MAPE
0	MLR Full Model	0.659725	0.600709	0.594231	0.341500	0.395200	0.584400	0.628600	0.439066	1.136038
1	MLR Model FWD	0.658665	0.599367	0.594660	0.342500	0.396600	0.585200	0.629800	0.440398	1.132491
2	GD model	0.582790	0.582790	0.578508	0.360100	0.413000	0.600100	0.642700	0.452824	1.125142
3	Ridge (alpha=1)	0.659660	0.600624	0.594144	0.341500	0.395300	0.584400	0.628700	0.439555	1.133520
4	Ridge (alpha=2)	0.659477	0.600435	0.593952	0.341700	0.395500	0.584600	0.628900	0.440080	1.131099
5	Ridge gridsearch	0.659660	0.600624	0.594144	0.341500	0.395300	0.584400	0.628700	0.439555	1.133520
6	lasso gridsearch	0.659667	0.600986	0.594512	0.341500	0.395000	0.584400	0.628500	0.439299	1.130515
7	elasticnet gridsearch	0.659692	0.600731	0.594253	0.341500	0.395200	0.584400	0.628600	0.439373	1.133524
8	decisiontreeregressor grid search	0.762111	0.769505	0.765765	0.238700	0.228100	0.488600	0.477600	0.263866	0.651508
9	Randomforestregressor grid search(n=100)	0.789939	0.776467	0.772840	0.238700	0.228100	0.488600	0.477600	0.263866	0.651508
10	Randomforestregressor grid search(n=50)	0.789727	0.772566	0.768876	0.211000	0.225100	0.459300	0.474400	0.279685	0.767883
11	Randomforestregressor grid search(n=50)2	0.890398	0.836069	0.833409	0.110000	0.162300	0.331700	0.402900	0.143266	0.285410
12	Randomforestregressor grid search(n=70)	0.889271	0.834884	0.832205	0.111100	0.163400	0.333300	0.404200	0.145027	0.289804
13	Adaboost base model	0.620138	0.573788	0.566874	0.381200	0.421900	0.617400	0.649500	0.500329	1.246109
14	Adaboost base_est=rf	0.620138	0.573788	0.769289	0.039600	0.224700	0.199000	0.474000	0.302104	0.764084
15	Adaboost grid	0.620138	0.573788	0.830681	0.100500	0.164900	0.317000	0.406100	0.145693	0.290656
16	XGB base	0.999810	0.796730	0.793432	0.000200	0.201200	0.014100	0.448600	0.186249	0.426148
17	XGB_grid	0.947441	0.813176	0.810145	0.052700	0.184900	0.229600	0.430000	0.199698	0.440030

Comparison and selection of model

```
In [756]: result_table=result_table.sort_values(by=['R-squared:test'],ascending=False).reset_index(drop=True)
result_table
```

Out[756]:

	Model_Name	R-squared:train	R-squared:test	Adj. R-squared	MSE train	MSE test	RMSE train	RMSE test	MAE	MAPE
0	Stacking	0.869657	0.838324	0.835701	0.130800	0.160000	0.361700	0.400000	0.145225	0.346632
1	Randomforestregressor grid search(n=50)2	0.890398	0.836069	0.833409	0.110000	0.162300	0.331700	0.402900	0.143266	0.285410
2	Randomforestregressor grid search(n=70)	0.889271	0.834884	0.832205	0.111100	0.163400	0.333300	0.404200	0.145027	0.289804
3	XGB_grid	0.947441	0.813176	0.810145	0.052700	0.184900	0.229600	0.430000	0.199698	0.440030
4	XGB base	0.999810	0.796730	0.793432	0.000200	0.201200	0.014100	0.448600	0.186249	0.426148
5	Randomforestregressor grid search(n=100)	0.789939	0.776467	0.772840	0.238700	0.228100	0.488600	0.477600	0.263866	0.651508
6	Randomforestregressor grid search(n=50)	0.789727	0.772566	0.768876	0.211000	0.225100	0.459300	0.474400	0.279685	0.767883
7	decisiontreeregressor grid search	0.762111	0.769505	0.765765	0.238700	0.228100	0.488600	0.477600	0.263866	0.651508
8	lasso gridsearch	0.659667	0.600986	0.594512	0.341500	0.395000	0.584400	0.628500	0.439299	1.130515
9	elasticnet gridsearch	0.659692	0.600731	0.594253	0.341500	0.395200	0.584400	0.628600	0.439373	1.133524
10	MLR Full Model	0.659725	0.600709	0.594231	0.341500	0.395200	0.584400	0.628600	0.439066	1.136038
11	Ridge gridsearch	0.659660	0.600624	0.594144	0.341500	0.395300	0.584400	0.628700	0.439555	1.133520
12	Ridge (alpha=1)	0.659660	0.600624	0.594144	0.341500	0.395300	0.584400	0.628700	0.439555	1.133520
13	Ridge (alpha=2)	0.659477	0.600435	0.593952	0.341700	0.395500	0.584600	0.628900	0.440080	1.131099
14	MLR Model FWD	0.658665	0.599367	0.594660	0.342500	0.396600	0.585200	0.629800	0.440398	1.132491
15	GD model	0.582790	0.582790	0.578508	0.360100	0.413000	0.600100	0.642700	0.452824	1.125142
16	Adaboost base model	0.620138	0.573788	0.566874	0.381200	0.421900	0.617400	0.649500	0.500329	1.246109
17	Adaboost base_est=rf	0.620138	0.573788	0.769289	0.039600	0.224700	0.199000	0.474000	0.302104	0.764084
18	Adaboost grid	0.620138	0.573788	0.830681	0.100500	0.164900	0.317000	0.406100	0.145693	0.290656

1. Ridge Regression:

Most of the times our data can show multicollinearity in the variables. To analyze such data we can use Ridge Regression. It uses the L2 norm for regularization.

After applying the ridge regression with alpha equal to one (which is the optimal value using GridSearchCV), we get 0.6287 as the RMSE value.

After applying the ridge regression with alpha equal to two, the RMSE value changed to 0.6289. The coefficients obtained from ridge regression have smaller values as compared to the coefficients obtained from the previous model.

2. Lasso Regression:

Lasso regression shrinks the less important variable's coefficient to zero which makes this technique more useful when we are dealing with large number of variables.

It is a type of regularization technique that uses L1 norm for regularization. After applying the lasso regression with alpha equal to 0.01 (which is the optimal value using the GridSearchCV), the RMSE value is 0.6285.

3. Elastic Net Regression:

This technique is a combination of Ridge and Lasso regression techniques. It considers

the linear combination of penalties for L1 and L2 regularization.
With the elastic-net regression with optimal value using the GridSearchCV we get 0.6286 as the RMSE value.

GridSearchCV:

Hyperparameters are the parameters in the model that are preset by the user. GridSearch considers all the combinations of hyperparameters and returns the best hyperparameter values. Following are some of the parameters that GridSearchCV takes:

estimator: pass the machine learning algorithm model

param_grid: takes a dictionary having parameter names as keys and list of parameters as values

cv: number of folds for k-fold cross validation

Decision Tree:

Decision Tree is a non-parametric supervised learning method. It builds a model in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets, which is called splitting. A decision node is a node on which a decision of split is to be made. A node that can not be split further is known as the terminal/leaf node. A leaf node represents the decision. A decision tree can work with both numerical and categorical variables.

The decision tree is said to be over-fitted if it tries to perfectly fit all the observations in the training data. We can calculate the difference between the train and test accuracy to identify if there is over-fitting.

If we tune the hyperparameters in the decision tree, it helps to avoid the over-fitting of the tree.

Hence we used the hyperparameter tuning to get the best results from the Decision Tree.

Random Forest:

It is the method of constructing multiple decision trees on randomly selected data samples. We can use the bootstrap sampling method to select the random samples of the same size from the dataset to construct multiple trees. This method is used for both regression and classification analysis. The random forest returns the prediction based on all the individual decision trees prediction. For regression, it returns the average of all the predicted values; and for classification, it returns the class, which is the mode of all the predicted classes.

It avoids the over-fitting problem as it considers a random data sample to construct a decision tree.

To avoid the problem of Overfitting of the Decision Tree, we have used the RandomForestRegressor.

We have fine tuned the hyperparameters of the RandomForestRegressor to get the best results.

For $n_estimators = 50$, we got the Maximum value of RMSE as 0.8904 and also we can observe that the MSE, RMSE values have significantly reduced using this method

From this table we can see that the Model arrived using stacking has the best overall score out of all the models built. It even has the least difference between RMSE:train and RMSE:test and R2:train and R2:test meaning that overfitting is not present. The difference between the adjusted R2 and R2 is also the least.

Results & Discussion

Stacking is a machine learning technique that takes several classification or regression models and uses their predictions as the input for the meta-classifier (final classifier) or meta-regressor (final regressor).

(Test, Train) : (0.8321771417819457, 0.8682966980917672)

```
Out[749]: StackingRegressor(cv=5,
                        estimators=[('rf2',
                                     RandomForestRegressor(criterion='mse',
                                                             max_depth=9,
                                                             min_samples_split=9,
                                                             n_estimators=70,
                                                             random_state=10)),
                                     ('rf1',
                                     RandomForestRegressor(criterion='mse',
                                                             max_depth=9,
                                                             min_samples_split=9,
                                                             n_estimators=50,
                                                             random_state=10))],
                        final_estimator=RandomForestRegressor(criterion='mse',
                                                             max_depth=9,
                                                             min_samples_split=9,
                                                             n_estimators=50,
                                                             random_state=10),
                        n_jobs=-1)
```

In [751]: m

```
Out[751]: Model_Name      Stacking
R-squared:train    0.869657
R-squared:test     0.838324
Adj. R-squared     0.835701
MSE train         0.130800
MSE test          0.160000
RMSE train        0.361700
RMSE test         0.400000
MAE               0.145225
MAPE              0.346632
dtype: object
```

In [752]: s.score(X_test,y_test),s.score(X_train,y_train)

```
Out[752]: (0.8383235917441632, 0.8696572318471694)
```

Conclusion:

In this paper we have successfully used different models like Bivariate linear regression using least square method, gradient descent, ridge, lasso and elasticnet regressor, decision tree regressor as well as Randomforest regressor. For obtaining the final model we used

GridsearchCV for hyperparameter tuning, boosting techniques and Stacking Regressor to improve the model for determining the medical insurance. The task was to predict the medical insurance amount for the patient. The analysis was implemented in the supervised regression model and performance metrics like r-squared, Adjusted r-squared, RMSE for train and test were calculated. This paper provides an effective basis for medical insurance premium in order to identify the amount for medical insurance for the patients based on the medical issues.