

DATA MINING AND PREDICTIVE ANALYSIS

LAB PROJECT REPORT

PROJECT TITLE :
MEDICAL INSURANCE COST ANALYSIS

Project by:

Name	Registration Number	Roll Number	Semester and Branch	Section
Jay Shah	200953134	37	5th Sem, CCE	B
Vishist Bansal	200953142	38	5th Sem, CCE	B
Naman Khurana	200953146	39	5th Sem, CCE	B

Introduction

After the covid era, medical costs have increased manifold. Thus, we decided to make a project which considers factors such as age, BMI, smoking habits, sex etc. to accurately predict the medical insurance cost of an Individual. It involves the use of machine learning algorithms and open-source libraries to predict the insurance cost values, visualising the results and general analysis of the performance of the Algorithm. When an insurance company suggests a price for a person's medical insurance price, they can compare it with the price predicted by the Algorithm and check if they are being overcharged. It also aims to improve upon the pre-existing methods that have been previously used for prediction.

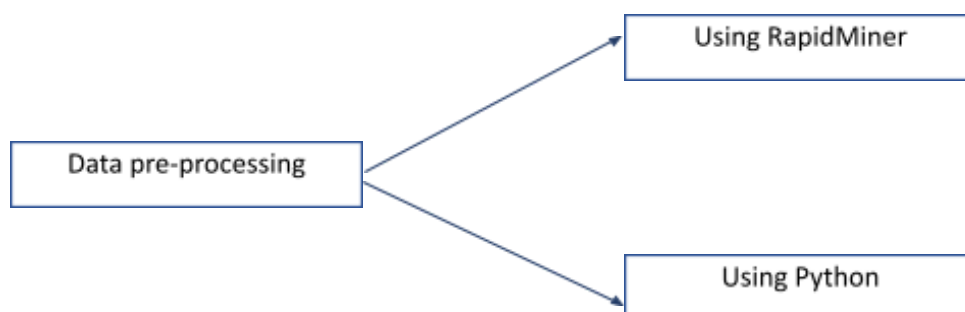
Methodology

1. Data pre-processing and warehousing

We used a dataset having the following attributes:

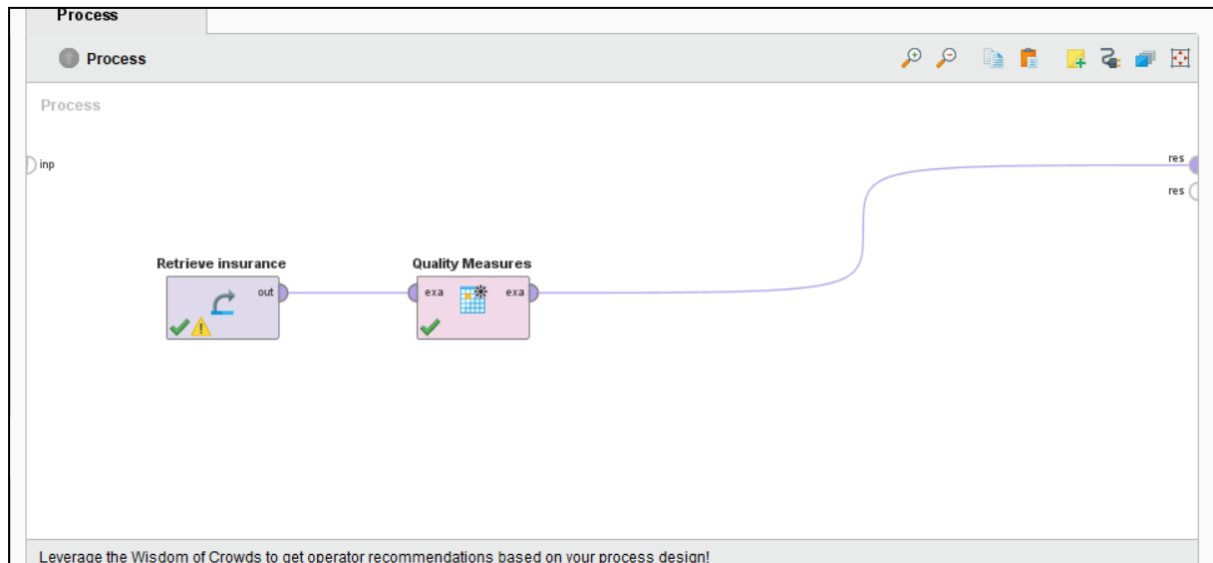
1. Age (Int)
2. Sex (Nominal String)
3. BMI (Int)
4. Children (Binary)
5. Smoker (Binary)
6. Region (Nominal String)
7. Charges (Float)

The tools that we used for pre-processing the dataset:



We first use **RapidMiner** to do the pre-processing:

Here the *Quality Measures* operator is used.



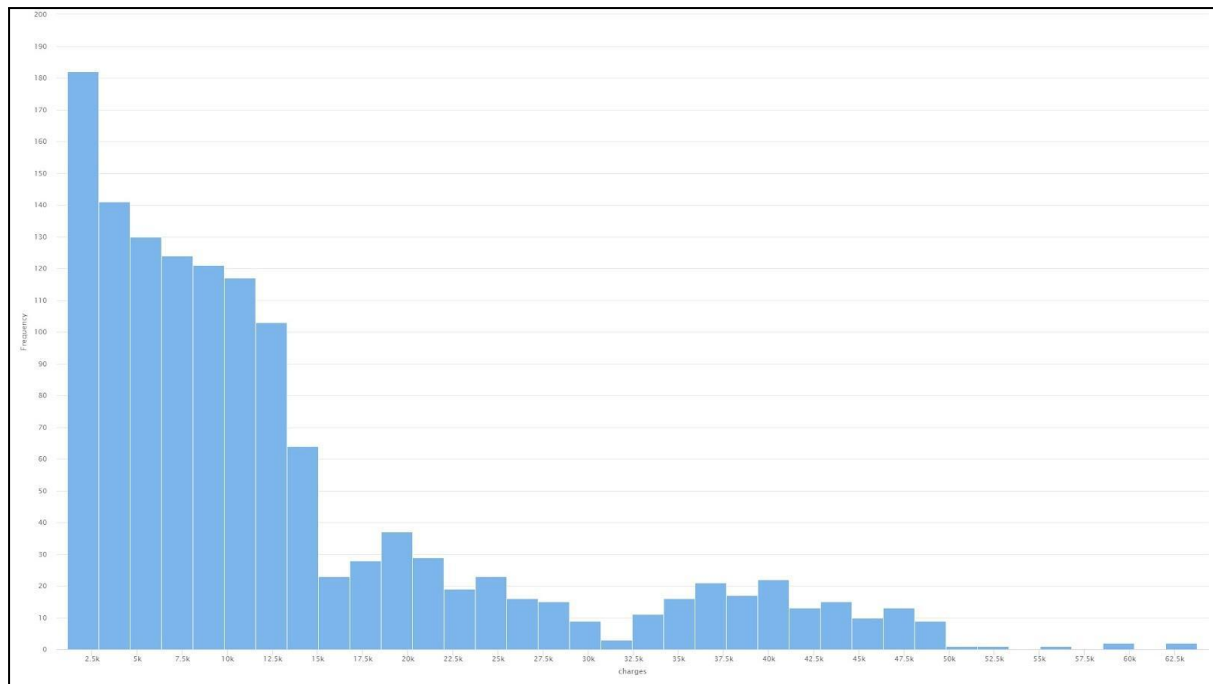
Row No.	Attribute	ID-ness	Stability	Missing	Text-ness
1	age	0.035	0.052	0	0
2	sex	0.001	0.505	0	0.023
3	bmi	0.011	0.010	0	0
4	children	0.004	0.429	0	0
5	smoker	0.001	0.795	0	0.010
6	region	0.003	0.272	0	0.041
7	charges	0.001	0.001	0	0

- As shown, there are no missing values so there is no need to replace them.

Statistics	
Name	Value
Minimum	1121.874
Maximum	63770.428
Average	13270.422
Standard Deviation	12110.011

- These are the basic statistics for the charges column, showing the range.

DMPA Project Report: Medical Insurance Cost



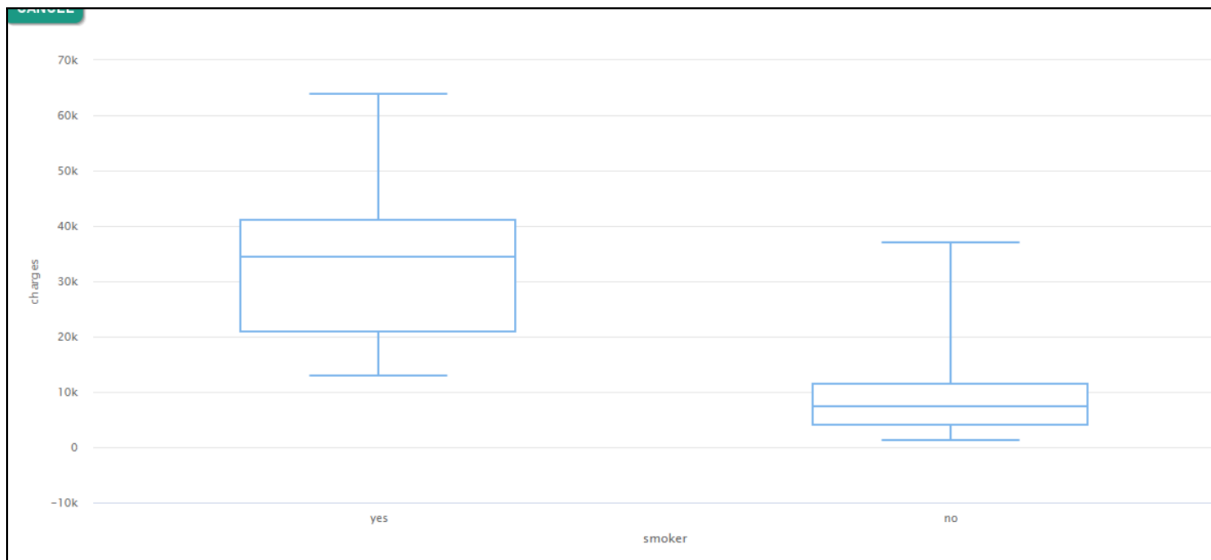
- A Histogram showing the distribution of charges

Next, we used the *Outlier Analysis* tool

Distance-based Outliers - Outlier Data								
Row No.	outlier	age	bmi	charges	children	region	sex	smoker
1	true	64	31.300	47291.055	2	southwest	female	yes
2	true	61	29.920	30942.192	3	southeast	female	yes
3	true	63	32.200	47305.305	2	southwest	female	yes
4	true	52	46.750	12592.534	5	southeast	female	no
5	true	21	25.700	17942.106	4	southwest	male	yes
6	true	54	47.410	63770.428	0	southeast	female	yes
7	true	33	35.530	55135.402	0	northwest	female	yes
8	true	37	47.600	46113.511	2	southwest	female	yes
9	true	61	33.330	36580.282	4	southeast	female	no
10	true	22	52.580	44501.398	1	southeast	male	yes
11	true	39	18.300	19023.260	5	southwest	female	yes
12	true	23	42.750	40904.200	1	northeast	female	yes
13	true	52	34.485	60021.399	3	northwest	male	yes

- It detects that 13 rows are outliers but since insurance values can be at extreme ends and this needs to be accounted for, we will keep the values as it is.

Another interesting observation directly shows that *smokers* have a higher insurance *charge* in general, this is illustrated in the boxplot below.

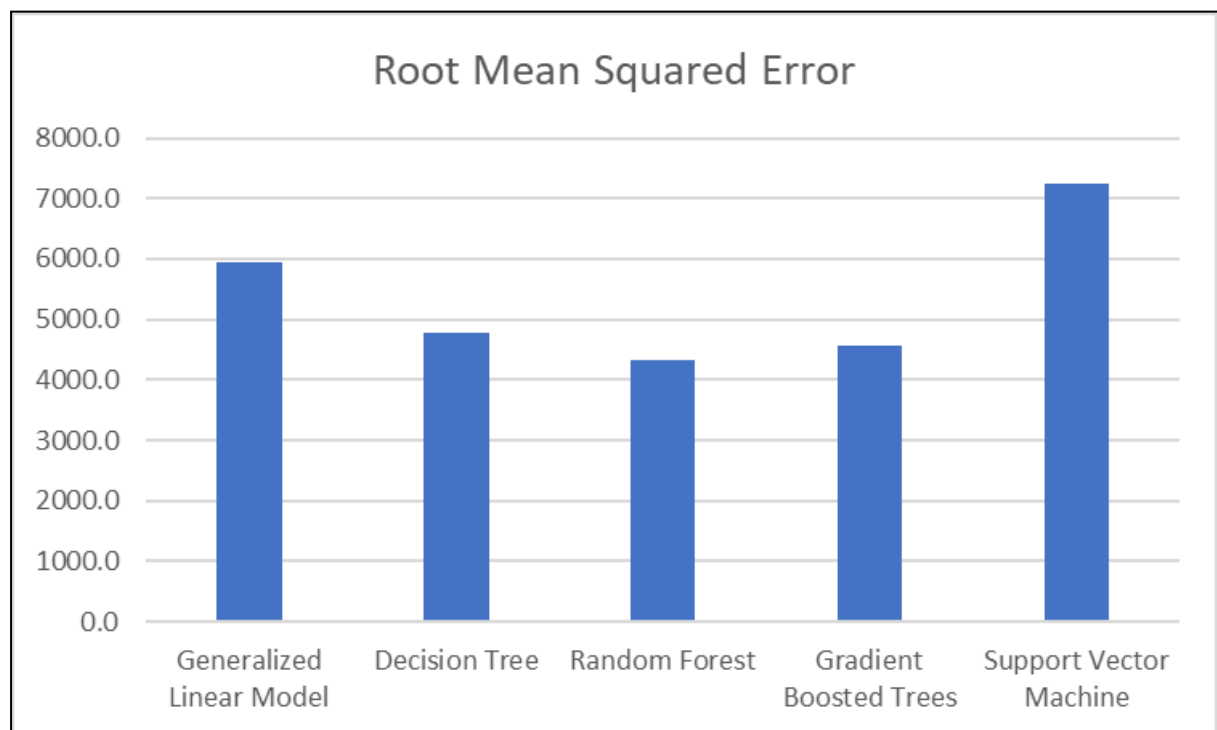


- Boxplot showing how smokers generally have higher charges

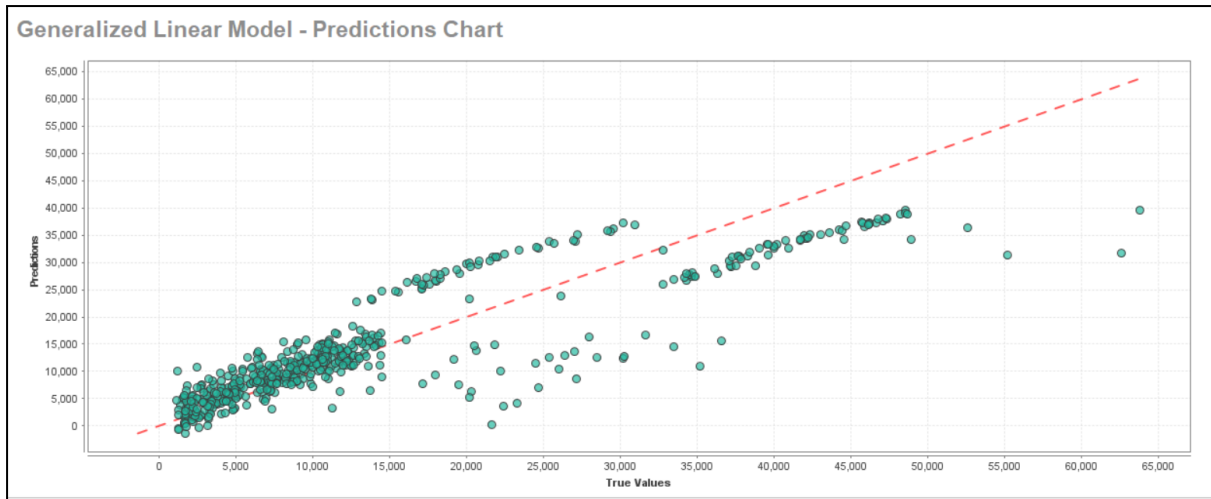
Instead of directly applying the Algorithms, we used RapidMiner's auto model feature to get a general idea on the performance of these Algorithms and gauge the applicability of each one

	A	B	C	D	E	F
1	Model	Root Mean Squared Error	Standard Deviation	Total Time	Training Time (1,000 Rows)	Scoring Time (1,000 Rows)
2	Generalized Linear Model	5934.3	795.8	219.0	21.7	7.5
3	Decision Tree	4788.5	762.9	195.0	2.2	7.5
4	Random Forest	4312.9	480.5	4532.0	17.2	76.6
5	Gradient Boosted Trees	4552.0	687.8	7819.0	162.2	31.8
6	Support Vector Machine	7232.8	1377.9	5070.0	341.6	50.5

- Results in tabular form



- Graph showing RMSE values for the Algorithms applied



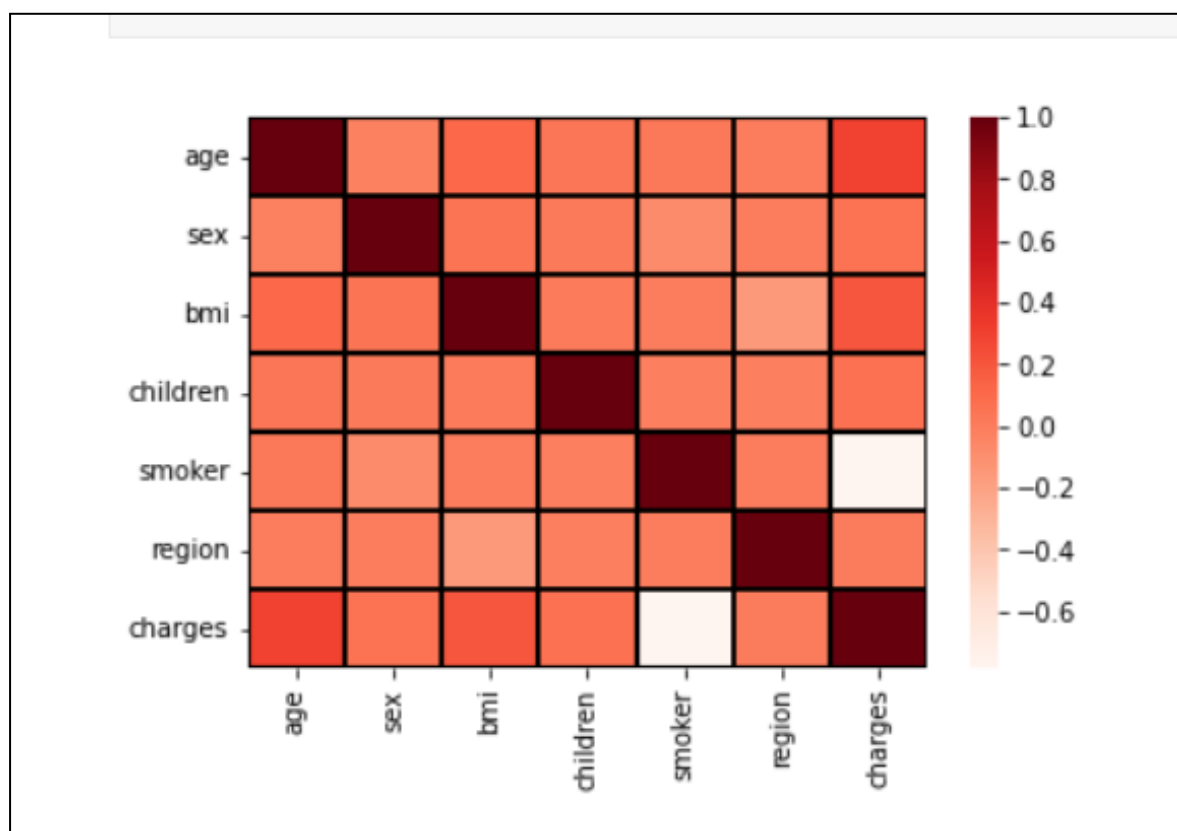
Example for the performance of the Generalized Linear Model

*According to our initial analysis, we can already see that **Random Forest Regression** is performing better than the other algorithms, given the correct parameters are given to the model*

Correlation Analysis of the Attributes:**Correlations**

Attribut...	age	bmi	charges	children	region =...	region =...	region =...	sex = m...	smoker...
age	1	0.109	0.299	0.042	-0.000	-0.012	0.010	-0.021	0.025
bmi	0.109	1	0.198	0.013	-0.136	0.270	-0.006	0.046	-0.004
charges	0.299	0.198	1	0.068	-0.040	0.074	-0.043	0.057	-0.787
children	0.042	0.013	0.068	1	0.025	-0.023	0.022	0.017	-0.008
region = ...	-0.000	-0.136	-0.040	0.025	1	-0.346	-0.321	-0.011	0.037
region = ...	-0.012	0.270	0.074	-0.023	-0.346	1	-0.346	0.017	-0.068
region = ...	0.010	-0.006	-0.043	0.022	-0.321	-0.346	1	-0.004	0.037
sex = ma...	-0.021	0.046	0.057	0.017	-0.011	0.017	-0.004	1	-0.076
smoker ...	0.025	-0.004	-0.787	-0.008	0.037	-0.068	0.037	-0.076	1

- Values for correlation



- Heatmap showing the correlation between the attributes

Rapid Miner has given us a rough idea on how to move on with the analysis.

Now we use **python and its libraries** to further pre-process the dataset:

First, we change the nominal attributes *sex*, *smoker* and *region* to numeric form

```
clean_data = {'sex': {'male' : 0 , 'female' : 1} ,  
              'smoker': {'no': 0 , 'yes' : 1},  
              'region' : {'northwest':0, 'northeast':1,'southeast':2,'southwest':3}  
             }  
data_copy = data.copy()  
data_copy.replace(clean_data, inplace=True)
```

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	3	16884.92400
1	18	0	33.770	1	0	2	1725.55230
2	28	0	33.000	3	0	2	4449.46200
3	33	0	22.705	0	0	0	21984.47061
4	32	0	28.880	0	0	0	3866.85520

- Preview after changing

Since the visualization part is already done in rapid miner, we move on to scaling/normalization.

Age, BMI and Charges need to be normalized to avoid erroneous results. We will use Standard Scaler for this

```

tempbmi = data_pre["bmi"]
tempbmi = tempbmi.values.reshape(-1,1)
data_pre["bmi"] = StandardScaler().fit_transform(tempbmi)
data_pre["bmi"]

```

[35] ✓ 0.1s

- Scaling for the bmi attribute

Similarly, we scale the age and the charges

	age	sex	bmi	children	smoker	region	charges
0	-1.438764	1	-0.453320	0	1	3	0.298584
1	-1.509965	0	0.509621	1	0	2	-0.953689
2	-0.797954	0	0.383307	3	0	2	-0.728675
3	-0.441948	0	-1.305531	0	0	0	0.719843
4	-0.513149	0	-0.292556	0	0	0	-0.776802
...
1333	0.768473	0	0.050297	3	0	0	-0.220551
1334	-1.509965	1	0.206139	0	0	1	-0.914002
1335	-1.509965	1	1.014878	0	0	2	-0.961596
1336	-1.296362	1	-0.797813	0	0	3	-0.930362
1337	1.551686	1	-0.261388	0	1	0	1.311053

- Preview of the data after scaling

Predictive analysis:

Now that the data has been normalized/scaled suitably, the preprocessing part is over. In the next step, we are separating the data, now we have a testing dataset and a training dataset. This allows us to train our models and also test their performance.

```

X = data_pre.drop('charges',axis=1).values
y = data_pre['charges'].values.reshape(-1,1)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)

print("Dimensions after splitting the data:")
print('X_train : ', X_train.shape)
print('y_train : ', y_train.shape)
print('X_test : ', X_test.shape)
print('y_test : ', y_test.shape)

```

✓ 0.6s

```

Dimensions after splitting the data:
X_train : (1070, 6)
y_train : (1070, 1)
X_test : (268, 6)
y_test : (268, 1)

```

- Here we have split the data with a test size of 20%. The dimensions of the testing and training data have been illustrated.

Next we are importing the essential modules that will allow us to apply the algorithms and also get the metrics for analysis.

```

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import cross_val_score, GridSearchCV
#comparision = {}

```

[59] ✓ 0.7s

- After this, we will still require more modules but they will be imported as and when they are required.

Algorithm 1: Linear Regression

Implementation:

```
> ✓  
linear_reg = LinearRegression()  
linear_reg.fit(X_train, y_train)  
cv_linear_reg = cross_val_score(estimator = linear_reg, X = X, y = y, cv = 10)  
  
y_pred_linear_reg_train = linear_reg.predict(X_train)  
r2_score_linear_reg_train = r2_score(y_train, y_pred_linear_reg_train)  
  
y_pred_linear_reg_test = linear_reg.predict(X_test)  
r2_score_linear_reg_test = r2_score(y_test, y_pred_linear_reg_test)  
  
import numpy as np  
rmse_linear = (np.sqrt(mean_squared_error(y_test, y_pred_linear_reg_test)))  
60] ✓ 0.2s
```

Performance:

```
Cross Validation Score: 0.745  
R Squared score (train) : 0.741  
R Squared score (test) : 0.783  
RMSE : 0.480
```

Algorithm 2: Ridge Regression

- For ridge regression, we are not using the pre normalized data, instead we will scale the train and test split data separately.

```

X_c = data_copy.drop('charges',axis=1).values
y_c = data_copy['charges'].values.reshape(-1,1)

X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_c,y_c,test_size=0.2, random_state=42)

X_train_scaled = StandardScaler().fit_transform(X_train_c)
y_train_scaled = StandardScaler().fit_transform(y_train_c)
X_test_scaled = StandardScaler().fit_transform(X_test_c)
y_test_scaled = StandardScaler().fit_transform(y_test_c)

```

[43] ✓ 0.5s

- Next we are creating a Pipeline for the ridges

```

from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import Ridge

steps = [ ('scalar', StandardScaler()),
          ('poly', PolynomialFeatures(degree=2)),
          ('model', Ridge()) ]
ridge_pipe = Pipeline(steps)

```

[44] ✓ 0.3s

- Now we will use Grid Search to get the best parameters

```

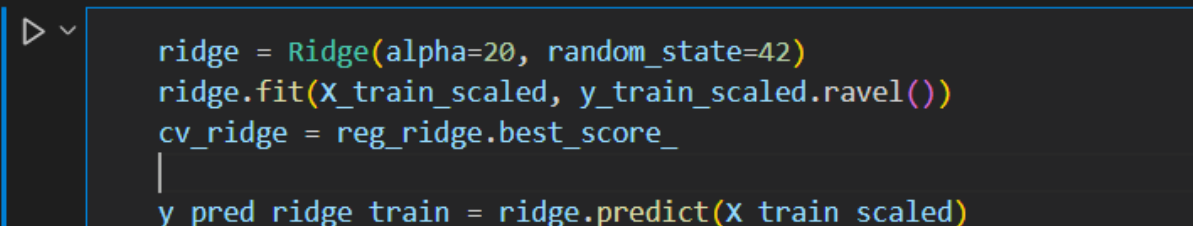
parameters = {
    'model__alpha': [1e-15, 1e-10, 1e-8, 1e-3, 1e-2, 1, 2, 5, 10, 20, 25, 35, 43, 55, 100],
    'model__random_state' : [42]}
reg_ridge = GridSearchCV(ridge_pipe, parameters, cv=10)
reg_ridge = reg_ridge.fit(X_train, y_train.ravel())
reg_ridge.best_estimator_, reg_ridge.best_score_

```

[61] ✓ 0.5s

- Obtained best parameters and best score:

```
(Pipeline(steps=[('scalar', StandardScaler()), ('poly',  
PolynomialFeatures()),  
                ('model', Ridge(alpha=20, random_state=42))]),  
0.8259990140429396)
```



```
ridge = Ridge(alpha=20, random_state=42)  
ridge.fit(X_train_scaled, y_train_scaled.ravel())  
cv_ridge = reg_ridge.best_score_  
|  
y_pred_ridge_train = ridge.predict(X_train_scaled)
```

- Here we have applied Ridge Regression using the best values of the parameters found

Performance:

```
Cross Validation Score: 0.826  
R Squared score (train) : 0.741  
R Squared score (test) : 0.784  
RMSE : 0.465
```

Algorithm 3: Random Forest Regressor

- First we are finding the best parameters again using Grid Search

```

reg_rf = RandomForestRegressor()
parameters = { 'n_estimators': [600, 1000, 1200],
               'max_features': ["auto"],
               'max_depth': [40, 50, 60],
               'min_samples_split': [5, 7, 9],
               'min_samples_leaf': [7, 10, 12],
               'criterion': ['mse']}

reg_rf_gscv = GridSearchCV(estimator=reg_rf, param_grid=parameters, cv=10, n_jobs=-1)
reg_rf_gscv = reg_rf_gscv.fit(X_train_scaled, y_train_scaled.ravel())
[49] ✓ 0.4s

```

- The estimator found these parameters to be the best:

```

(bootstrap=True, criterion='mse', max_depth=50, max_features='auto', max_leaf_nodes=None,
 min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=12, min_samples_split=7,
 min_weight_fraction_leaf=0.0, n_estimators=1200, n_jobs=None, oob_score=False,
 random_state=None, verbose=0, warm_start=False)

```

Best Score = 0.8483687880955955

- Now we apply the Algorithm using these parameters

```

rf_reg = RandomForestRegressor(max_depth=50, min_samples_leaf=12,
                               min_samples_split=7, n_estimators=1200)
rf_reg.fit(X_train_scaled, y_train_scaled.ravel())
[50] ✓ 2.5s

```

- Performance:

```

Cross Validation Score: 0.848
R Squared score (train) : 0.885
R Squared score (test) : 0.879
RMSE : 0.348

```

Further, we shall compare the performance of the algorithms. The comparison is illustrated in the oncoming results section of the report.

#Note: Grid Search took around 10 minutes to find the best parameters for RFR which is a huge disadvantage.

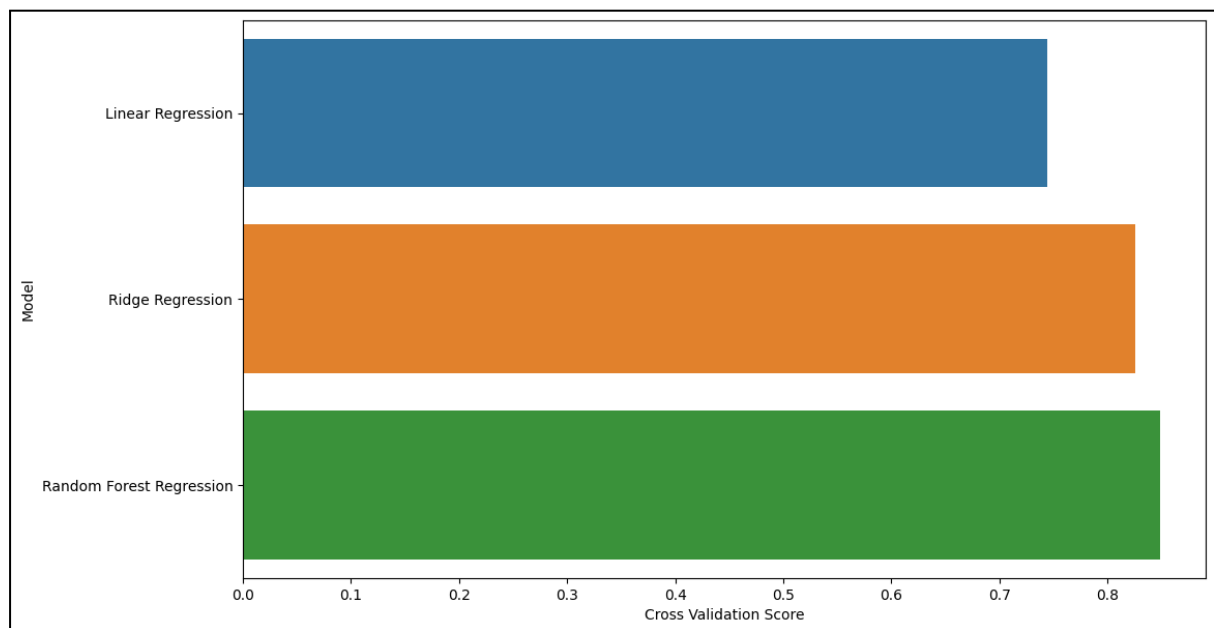
Results and Discussions

- Now we start comparing the metrics of the three algorithms

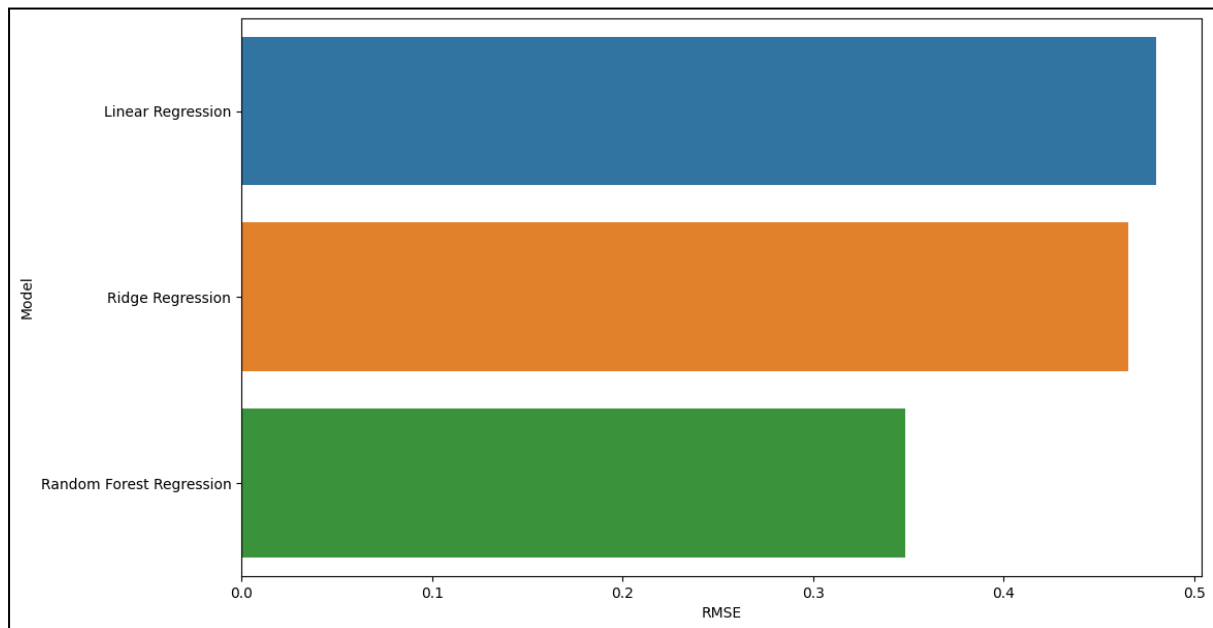
Model	RMSE	R2_Score(training)	R2_Score(test)	Cross-Validation
Linear Regression	0.479808	0.74141	0.782694	0.744528
Ridge Regression	0.465206	0.74115	0.783800	0.825999
Random Forest Regression	0.348313	0.88453	0.878678	0.848369

- To better understand our final results, we are plotting them using a seaborn barplot

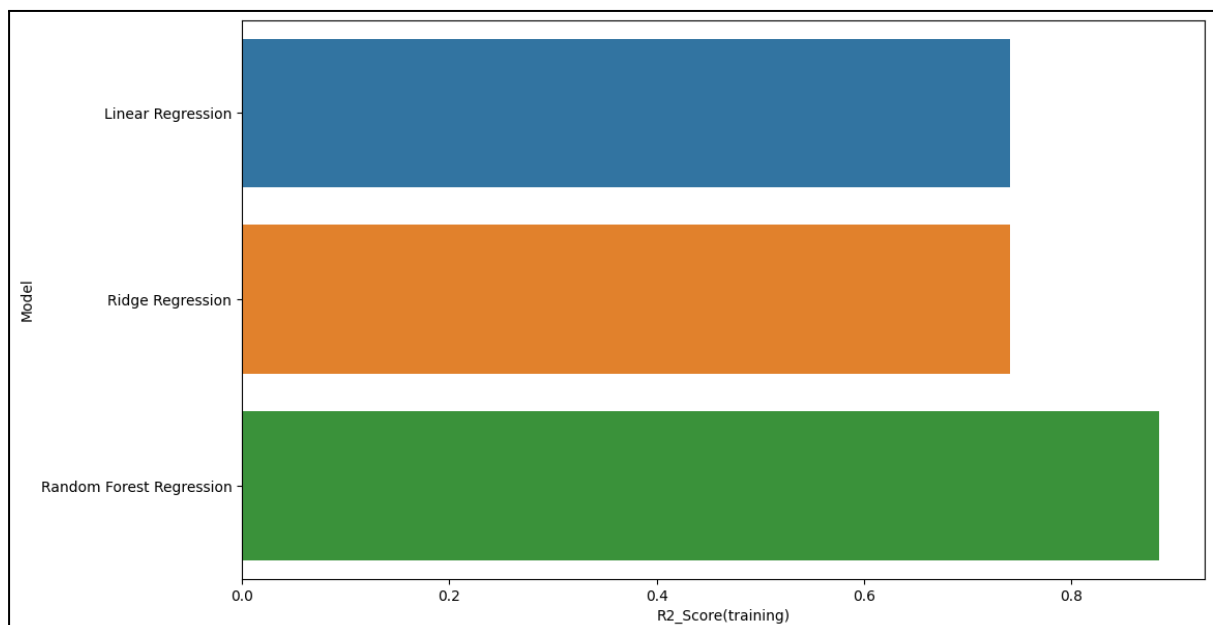
Barplot for Cross Validation Score:



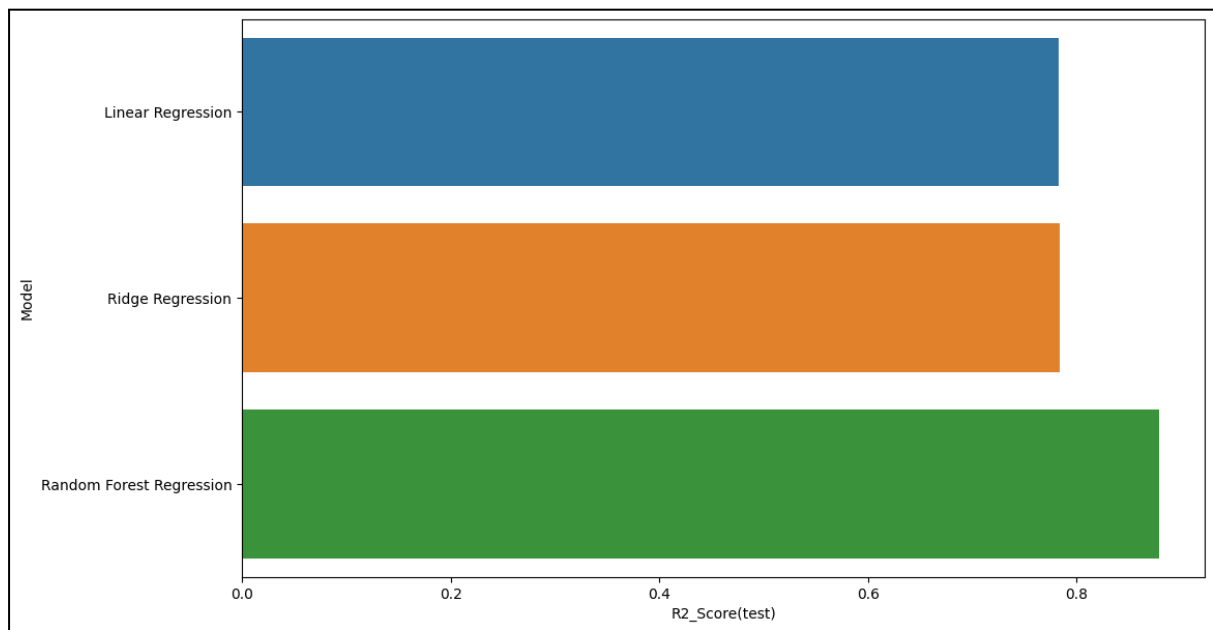
Barplot for RMSE:



Barplot for R Squared (Training):



Barplot for R Squared (Test):



- Random Forest Regression shows the best performance for all the metrics, so the final results using RFR are:

```
Cross Validation Score: 0.848
R Squared score (train) : 0.885
R Squared score (test)  : 0.879
RMSE : 0.348
```

Conclusions

Using our results, we can conclude the following:

1. Attribute *Smoker* has the highest correlation with the medical charges. Whether a person Smokes or not affects the medical costs the most. A person who smokes generally has a higher Insurance cost rate than a person who doesn't. As portrayed in the initial boxplot
2. The other attributes don't have a large amount of correlation with the charges attribute; no conclusion can be formed for them. We had expected the bmi attribute to have a high effect on the charges but our results showed that there wasn't a large amount of correlation.
3. The results clearly show that the **Random Forest Regressor** algorithm has the best performance making it the optimal model for a patient to predict insurance costs. It has the lowest RMSE value out of all the algorithms and the highest value of Cross Validation Score and R Squared Score
4. The problem we faced whilst applying the Random Forest Regressor algorithm is that the parameters have huge ranges and unless we are using the best possible values for the parameters, the performance is very average. The Grid Search that found the best parameters took over 10 minutes, making the time performance of the algorithm relatively bad
5. The Ridge Regressor shows lower metrics than RFR but the best parameters can be found quickly (in a fraction of a second) so it is still a good option for a patient to quickly predict their insurance cost.

Our project can be easily used by a patient to predict his or her insurance costs. They can enter their details and our Random Forest Regressor will produce a close estimate of how much their medical insurance will be.

Literature Survey

Yang et al. [1] in this paper provides the number of days that a patient will be admitted in the hospital for. Dataset used was insurance claim data from more than 200000 individuals over three years. Regression decision tree algorithm is used. AUC score \Rightarrow 0.843. RMSE of 0.428 and proposed RMSE of 0.381. Root mean square error of 0.428 is achieved with the proposed model having RMSE of 0.381 which is lower than the top team of the HHP competition which has a RMSE of 0.461. But the dataset has many missing values. Important details like diagnosis codes can be missing which can make the prediction model less effective.

Venkata et al. [2] in this paper shows stacking regressor models which are used to analyse the price of medical insurance which is affected by a number of factors. The model implemented also recommends lifestyle changes to a person i.e., what long term changes they should incorporate in their life so that the amount they will pay for medical insurance will decrease. Dataset used is from Kaggle and it involves 1338 records with attributes such as 'age', 'gender', 'BMI' etc. Stacking Regressor algorithm with Cat boost at level 1 is used. RMSE \Rightarrow 0.352 and Explained variance score \Rightarrow 0.85 is observed. In addition to predicting the insurance cost, it also gives advice to an individual so that they can reduce the cost. For example, if the person is a smoker, the algorithm gives the cost of the insurance if was a non-smoker and shows a graph for comparison. Authors had to try numerous regression algorithms and parameters to arrive at the best performing one.

Janet et al. [3] has developed a prediction model that will compare various machine learning algorithms that will be used to predict insurance costs for medical patients. It uses another dataset from Kaggle. In addition to this functionality, the project also implements end to end AES encryption so that the system becomes secure and potentially sensitive data of the patient doesn't get misused. So, the model is now reliable and secure at the same time. Support Vector regression and AES encryption algorithms are used. RMSE of SVR \Rightarrow 0.34, R squared value of SVR \Rightarrow 0.87. This paper uses AES algorithm which provides additional layers of security. The personal information thus remains safe. Though the performance of the algorithm and the visualization made is not displayed in a convenient way in the research paper.

Akhil et al. [4] tries to find a relation between factors such as Smoking habits, Age etc and the medical insurance cost of patients. Then he used prominent factors, applied Linear Regression and then tried to predict the correlation between the factors and Medical Costs. Multiple Linear regression approaches are used. Multiple R squared value observed is 0.7509. This paper is effective in finding factors that effects medical cost the most, like gender and region have little to no impact whereas BMI, Smoking habits, Age affect majorly. This model has good cross-validity. The successful percentage prediction is achieved is just 75% which is low as compared to other methods of prediction.

Xingyi et al. [5] in his research showed that instead of focusing on efficiency of building decision trees, we should use a new splitting attributes condition which can be used. The results of this paper portray a lower misclassification cost. CSL, Regression Decision Tree algorithms are used. Gain Ratio = 0.82, Misclassification Ratio = 0.2 is obtained. This uses a ground breaking tool to accurately predict Multiple Cost Scales using splitting criterion which gives better prediction results than the conventional decision tree approach. Missing values are appropriately taken into consideration thus reducing the fluctuations in the dataset. Still the splitting criterion requires in-depth knowledge of the factors taken in consideration so as to find the splitting attributes.

Guiduo et al. [6] in his research showed that conventional methods of analysis of Medical insurance companies are lacking in both flexibility and efficiency.. He proposed that correlation between medical costs and relevant factors need to be mined using FP-Growth algorithms. This paper proposes an improved version of Decision tree algorithms which aims to provide improved prediction of costs and improve decision making. Improved FP-Growth algorithm used. Classification Accuracy = 0.8178, MDMP (Medical Decision Model with Pruning) = 8% - 15%. An Improved Method of FP-Growth Algorithm is framed which proves to have better accuracy than conventional methods. Missing Values can cause huge fluctuations in the accuracy and efficiency of the prediction.

Ravishankar et al. [7] in his research focused on three areas: Big Data, machine learning, and healthcare. Machine learning is a central part of our business. The three model classes are based on multiple linear regression, the permutation method and time series modeling were used to train on the SPACRS dataset. It was found that the best results were given by DNN's followed by Regression Tree. Multiple linear regression, regression trees and deep neural networks (DNNs) algorithms are used. Model - R^2 score for various algorithms were observed as: Linear regression - 0.64, Regression tree - 0.68, DNN - 0.71. Deep Learning algorithms can learn high-level features from data in an incremental manner. This eliminates the need to have any domain expertise, which saves time and resources for data extraction. Though DNN can still be computationally expensive and more complex than conventional machine learning methods.

Krittika et al. [8] in this paper, has performed a assessment among the real and anticipated prices of predicting rate and then a graph has been plotted in this foundation if you want to enlighten us to select the first-class-perfect regression set of rules for the coverage prediction. Algorithms including selection tree, random wooded area, linear and polynomial regression regression are carried out to get the prediction analysis. After executing those algorithms for prediction, correctness was measured with the help of the Coefficient of Root Mean Squared Error and determination of every set of rules to test for the first-class-perfect set of rules. Coefficient of determination (r^2_score) obtained: Linear Regression - 0.754904, Decision-Tree Regression 0.746422, Random-Forest Regression 0.862533, Polynomial Regression 0.843962. It is very accurate machine learning algorithm which is very efficient in giving best results with minimal effort. The speed for training the dataset is faster and reduces the issue of overfitting.

Mohamed et al. [9] has shown in his research that Machine learning (ML) for the coverage enterprise zone could make the wording of coverage regulations extra efficient. This observe demonstrates how exclusive fashions of regression can forecast coverage costs. And we are able to evaluate the outcomes of the models, for example, Multiple Linear Regression, Support Vector Machine, Random Forest Regressor, CART, k-Nearest Neighbours and Deep Neural Network using their algorithms. Model - R^2 values observed: Random Forest Regressor - 0.849299, Support Vector Machine - 0.842307, Decision Tree (CART) - 0.833493, DNN - 0.809799, Multiple Linear Regression - 0.755813, K Nearest neighbours - 0.318513. Random Forest is faster to train than decision trees since we work only on a subset of features in this model. A large number of trees can make the algorithm too slow which is a limitation. Thus it becomes ineffective for real-time predictions. They are fast to train but difficult to predict algorithms.

Xianghai et al. [10] took the dataset from a medical study from China involving people from ages 20- 74, conducted between June 2007 and May 2008. Her type 2 diabetes was determined based on fasting plasma glucose criteria and 2-hour plasma glucose criteria for subjects with no history of diabetes. New Chinese diabetes risk scores were calculated using

Beta coefficients made from multiple logistic regression models. AUC observed= 0.748. The logistic regression model is fast to train. Another advantage is that there are many papers on diabetes detection so the results can be easily compared. No actual data from blood samples was used.

References:

1. Prabhdeep Singh, Kiran Deep Singh, Vikas Tripathi, Vaibhav Chaudhari, "Use of Ensemble Based Approach to Predict Health Insurance Premium at Early Stage", 2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), pp.566-569, 2022.
2. Keshav Kaushik, Akashdeep Bhardwaj, Ashutosh Dhar Dwivedi, Rajani Singh, "Machine Learning-Based Regression Framework to Predict Health Insurance Premiums", International Journal of Environmental Research and Public Health, vol.19, no.13, pp.7898, 2022.

3. J. B, A. Ghosh and J. A. Kumar R, "End-to-End Encryption and Prediction of Medical Insurance Cost," 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI), 2022, pp. 846-850, doi: 10.1109/ICOEI53556.2022.9777238.
4. Kodiyan, Akhil Alfons, and Kirthy Francis. "Linear regression model for predicting medical expenses based on insurance data."
5. X. Liu, "Cost-sensitive Decision Tree with Missing Values and Multiple Cost Scales," 2009 International Joint Conference on Artificial Intelligence, 2009, pp. 294-297, doi: 10.1109/JCAI.2009.118.
6. G. Duan, D. Ding, Y. Tian and X. You, "An Improved Medical Decision Model Based on Decision Tree Algorithms," 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), 2016, pp. 151-156, doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.33.
7. A. R. Rao and D. Clarke, "A comparison of models to predict medical procedure costs from open public healthcare data," 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1-8, doi: 10.1109/IJCNN.2018.8489257.
8. K. Dutta, S. Chandra, M. K. Gourisaria and H. GM, "A Data Mining based Target Regression-Oriented Approach to Modelling of Health Insurance Claims," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 1168-1175, doi: 10.1109/ICCMC51019.2021.9418038.
9. Por otra parte, Mohamed H. realizó una comparación de diferentes modelos de ML y deep learning para predecir los gastos de atención medica en las aseguradoras, encontrando que el algoritmo Stochastic Gradient Boosting tuvo el mejor rendimiento para esta tarea (4). Contrario a lo reportado por Belisario P (5), quien sugirió que el mejor predictor se basó en un modelo de redes neuronales (6) generado a partir de los datos históricos del Hospital Tsuyama Chuo. ...
10. Zhou X, Qiao Q, Ji L, Ning F, Yang W, Weng J, Shan Z, Tian H, Ji Q, Lin L, Li Q, Xiao J, Gao W, Pang Z, Sun J. Nonlaboratory-based risk assessment algorithm for undiagnosed type 2 diabetes developed on a nation-wide diabetes survey. *Diabetes Care*. 2013 Dec;36(12):3944-52. doi: 10.2337/dc13-0593. Epub 2013 Oct 21. PMID: 24144651; PMCID: PMC3836161.