JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

Open Source Software Lab

Project Report

On

Medical Insurance Cost Prediction Using Machine Learning

SUBMITTED TO-

Purtee Kohli Ma'am

SUBMITTED BY-

Avisha Goyal - 21103057

Nikita Bansal - 21103069

Samyak Jain - 21103075

INTRODUCTION

People seek out various means of protecting themselves as well as their families when life is most at risk. Working hard and earning money satisfies merely their basic comforts of day-to-day existence. Risk cannot often be eliminated. By leveraging the money they have invested to safeguard them through financial support, financial companies have created a variety of solutions to shield both individuals and businesses from a variety of hazards. One cannot afford to instantly spend a large sum of money in times of emergency or unforeseen health concerns. Therefore, it makes sense to use insurance to save money for the future. The loss caused by various hazards is reduced or completely eliminated through insurance policies.

The amount covered by each policy that the customer must pay for must be precisely measured by the insurance provider. Due to the sensitive nature of the information, skilled individuals are employed for this purpose, but the possibility of mistakes is high. And thus ML is beneficial here. ML may generalize the effort or method to formulate the policy. These ML models can be learned by themselves. The model is trained on insurance data from the past. The model can then accurately predict insurance policy costs by using the necessary elements to measure the payments as its inputs. This decreases human effort and resources and improves the company's profitability. Thus the accuracy can be improved with ML.

Problem Statement

To build a model for Medical Insurance Cost Prediction Using Machine Learning. In this project machine learning algorithms such as linear regression, ridge regression, lasso regression and random forest regressor have been used.

Dataset Description

The collection consists of 1338 records with 7 attributes. The attributes are age, gender, bmi, children, smoker, and charges. The data was organised and kept in an insurance.csv file.

	age	sex	bmi	children	smoker	region	charges
1	19	female	27.9	0	yes	southwest	16884.924
2	18	male	33.77	1	no	southeast	1725.5523
3	28	male	33	3	no	southeast	4449.462
4	33	male	22.705	0	no	northwest	21984.47061
5	32	male	28.88	0	no	northwest	3866.8552
6	31	female	25.74	0	no	southeast	3756.6216
7	46	female	33.44	1	no	southeast	8240.5896
8	37	female	27.74	3	no	northwest	7281.5056
9	37	male	29.83	2	no	northeast	6406.4107
10	60	female	25.84	0	no	northwest	28923.13692
11	25	male	26.22	0	no	northeast	2721.3208
12	62	female	26.29	0	yes	southeast	27808.7251
13	23	male	34.4	0	no	southwest	1826.843
14	56	female	39.82	0	no	southeast	11090.7178
15	27	male	42.13	0	yes	southeast	39611.7577
16	19	male	24.6	1	no	southwest	1837.237
17	52	female	30.78	1	no	northeast	10797.3362
18	23	male	23.845	0	no	northeast	2395.17155
19	56	male	40.3	0	no	southwest	10602.385
20	30	male	35.3	0	yes	southwest	36837.467

The dataset is unsuitable for direct regression. In order to use the data with different regression techniques, dataset cleaning becomes crucial.Not every attribute in a dataset has an effect on the prediction. Some attributes even reduce accuracy, therefore it becomes necessary to delete them from the features of the code. Getting rid of these characteristics not only helps with accuracy but also with performance in general and speed. In health insurance many factors such as pre-existing body condition, family medical history, Body Mass Index (BMI), marital status, location, past insurances etc affects the amount. According to our dataset, age and smoking status has the maximum impact on the amount prediction with smoker being the one attribute with maximum effect. Children attribute had almost no effect on the prediction, therefore this attribute was removed from the input to the regression model to support better computation in less time.

Data Preprocessing

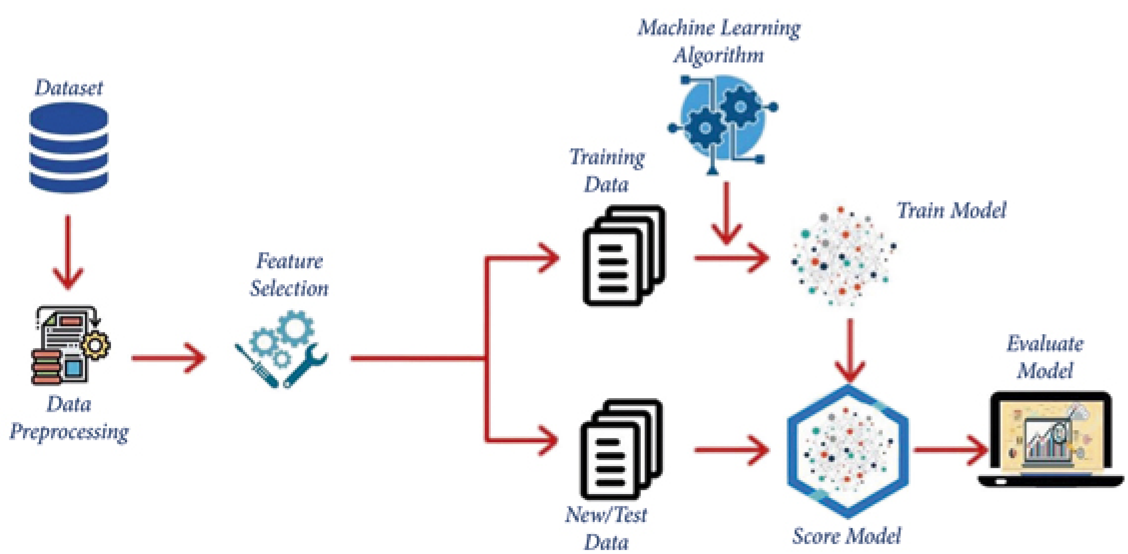
The dataset includes 7 variables. From these variables each one of these attributes has some contribution to estimate the cost of the insurance, which is our dependent variable. In this stage, the data is scrutinized and updated properly to efficiently apply the data to the ML algorithms.

Now the categorical variables are converted into numeric or binary values to represent either 0 or 1. For example, instead of "sex" with males or females, the "Male" variable would be considered as false (0) if the person is male. And "female" would be (1) following this phase now, we can apply this data to all regression models.

Age	Age of client
sex	Male / Female 0=Male 1=Female
BMI	Body mass index
children	Number of children the client have
smoker	Whether or not a client smokes 0=yes 1=no

region	Whether the client lives in southwest, northwest, southeast or northeast 0=southeast 1=southwest 2=northeast 3=northwest
Charges(Target Variable)	Medical Cost the client pay

Methodology



Implementation

Implementation Set-up (software used) -

In this section of the report, we are mentioning different tools and libraries which are used in our working model.

- Jupyter notebook
- Python 3.7 version is used for this project. Python is a very useful programming language.It is object oriented and interpreted.It is a high level language.There are lots of in-built libraries in Python for machine learning purposes which we can use easily.
- Windows Version Colab and python 3 can be used in all the operating systems including Windows, iOS and Linux. It is most useful in Linux but can be used in windows as well.It can be run on windows xp,vista,7,8 and the latest version windows 10 as well.

Tools & Libraries -

- Numpy:- Used to support Panda frameworks.
- Panda:- To Create Dataframe of the Image Pixel Values.
- Sklearn:- Python Library used for machine learning and statistical modelling including classification.
- Matplotlib:-Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication quality plots.
- Seaborn:- Data visualization library that is commonly used for data science and machine learning tasks. It is used to create interactive plots to answer questions about data.

Exploratory Data Analysis-

```
[126]: #getting information about dataset
       insurance_dataset.info()
```

Result:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

Description of various columns of dataset:-

```
[133]: #statistical measures of the dataset
       insurance_dataset.describe()
```

[133]:

	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

```
[134]: #distribution of age value
       sns.set()
       plt.figure(figsize=(8,8))
       sns.displot(insurance_dataset['age'])
       plt.title('Age Distribution')
       plt.show()
```

[135]:
```python
# Gender column
plt.figure(figsize=(8,8))
sns.countplot(x='sex', data=insurance_dataset)
plt.title('Sex Distribution')
plt.show()
```

[137]:
```python
# bmi distribution
plt.figure(figsize=(8,8))
sns.displot(insurance_dataset['bmi'])
plt.title('BMI Distribution')
plt.show()

"""Normal BMI Range --> 18.5 to 24.9"""
```

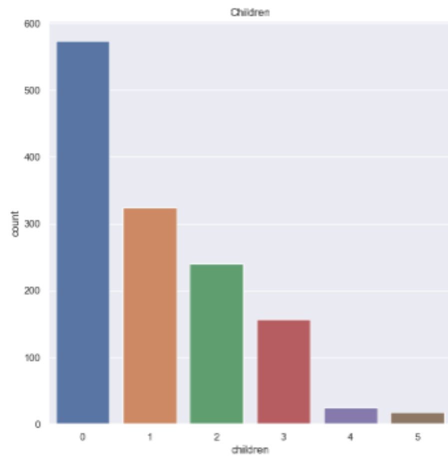
[138]:
```python
# children column
plt.figure(figsize=(8,8))
sns.countplot(x='children', data=insurance_dataset)
plt.title('Children')
plt.show()
```

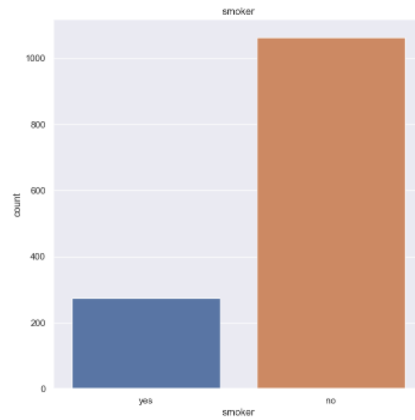
[140]:
```python
# smoker column
plt.figure(figsize=(8,8))
sns.countplot(x='smoker', data=insurance_dataset)
plt.title('smoker')
plt.show()
```

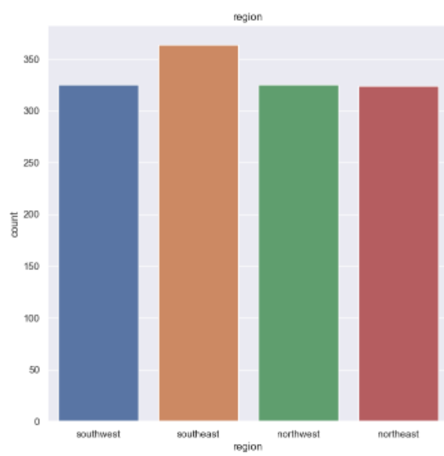
[142]:
```python
# region column
plt.figure(figsize=(8,8))
sns.countplot(x='region', data=insurance_dataset)
plt.title('region')
plt.show()
```

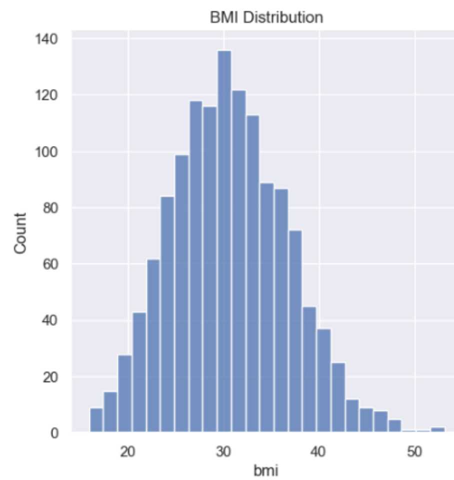
[144]:
```python
# distribution of charges value
plt.figure(figsize=(8,8))
sns.displot(insurance_dataset['charges'])
plt.title('Charges Distribution')
plt.show()
```

Result:-


Children
count
children
0
100
200
300
400
500
600
0 1 2 3 4 5


smoker
count
smoker
yes
no
0
200
400
600
800
1000


region
count
region
southwest
southeast
northwest
northeast
0
50
100
150
200
250
300
350


BMI Distribution
Count
bmi
0
20
40
60
80
100
120
140
20 30 40 50


Charges Distribution
Count
charges
0
25
50
75
100
125
150
175
200
0 10000 20000 30000 40000 50000 60000

Preprocessing Analysis-

```
[145]: # encoding sex column
       insurance_dataset.replace({'sex':{'male':0,'female':1}}, inplace=True)

[146]: 3 # encoding 'smoker' column
       insurance_dataset.replace({'smoker':{'yes':0,'no':1}}, inplace=True)

[147]: # encoding 'region' column
       insurance_dataset.replace({'region':{'southeast':0,'southwest':1,'northeast':2,'northwest':3}}, inplace=True)

[148]: """Splitting the Features and Target"""

       X = insurance_dataset.drop(columns='charges', axis=1)
       Y = insurance_dataset['charges']

       print(X)

       print(Y)
```

Result:-

```
      age  sex     bmi  children  smoker  region
0      19    1  27.900         0       0       1
1      18    0  33.770         1       1       0
2      28    0  33.000         3       1       0
3      33    0  22.705         0       1       3
4      32    0  28.880         0       1       3
...   ...  ...     ...       ...     ...     ...
1333   50    0  30.970         3       1       3
1334   18    1  31.920         0       1       2
1335   18    1  36.850         0       1       0
1336   21    1  25.800         0       1       1
1337   61    1  29.070         0       0       3

[1338 rows x 6 columns]
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

Observations:- In this code snippet, we are using the replace method to replace the character values with numeric values and drop method has been used to split the features and target.

Machine Learning Algorithms Used:-

Here we have used multiple machine learning algorithms to test and predict the medical insurance cost. R squared value is also calculated for each model to determine which of the used algorithms has the highest accuracy.

<u>Multi-Linear Regression</u> -

This is a more complex version of the simpler linear regression. For example, we use it when we want to predict the output of a single variable from a large number of inputs or when the expected value of one variable is based on the values of two or more other variables. Given the large number of independent variables, multiple linear regression has been applied in this case. The gender, smoker, and region columns in this table are encoded for calculating purposes. In column gender, 0 for male and 1 for female and in column smoker, 0 for yes and 1 for no. Similarly in the region, southeast:0, southwest:1, northeast:2, northwest:3. The mathematical equation for cost estimation,

$$Y = \alpha o+\alpha 1\times x1+\alpha 2\times x2+\alpha 3\times x3+\alpha 4\times x4+\alpha 5\times x5+\alpha 6\times x6$$

Y = Dependent variable and

x1, x2, x3, …… xn = multiple independent variables

The result Y will be the final cost of the insurance.

For training the model 80% of data is used and rest is used for testing the model as shown below-

```
[149]: """Splitting the data into Training data & Testing Data"""

       X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)

       print(X.shape, X_train.shape, X_test.shape)

       (1338, 6) (1070, 6) (268, 6)
```

```
[150]: """Model Training

       Linear Regression
       """

       # loading the Linear Regression model
       regressor = LinearRegression()

       regressor.fit(X_train, Y_train)

[150]: ▾ LinearRegression
       LinearRegression()
```

```
[151]: """Model Evaluation"""

       # prediction on training data
       training_data_prediction =regressor.predict(X_train)

[152]: # R squared value
       r2_train = metrics.r2_score(Y_train, training_data_prediction)
       print('R squared vale : ', r2_train)
```

Result for training data and error computation:-

```
R squared vale :  0.751505643411174
```

```
[153]: # prediction on test data
       test_data_prediction =regressor.predict(X_test)
       test_data_prediction

[153]: array([ 1520.59242161, 11570.5920178 , 10082.43849883,  2246.21754312,
         7881.28362035, 11081.50227956,  3538.24791808,   698.03224036,
        12223.4851558 ,  9611.93217623, 11657.51046259,  4891.0539656 ,
        29947.50192274,  -370.8384887 , 12401.36048618, 13243.21522903,
         3814.42216541,  7883.39384825, 29431.34485576,  2362.83672121,
        12505.50452609,  2256.75277238, 34468.01948464, 31742.4859866 ,
        30306.19118561,  9027.76110059,  1923.87420399, 15247.09503907,
         6542.61302531,  2104.79910554,  9484.36642532,  5794.91649267,
         4425.26853454,  5015.3811241 ,  9579.4545934 ,  4601.74838962,
        29875.58083252,  6797.04084444, 27239.25811383, 13999.0938259 ,
          313.55184653, 28415.75044713,  7886.54751277,  1478.09056648,
        10273.28966107,  8003.09003405, 11612.15283896,  8175.95966058,
        10753.45200738, 13802.18082647,  5740.90172027,  -737.13333209,
        26346.21771217, 37192.66032995,  7364.09646118, 17845.51752284,
         1412.63748094, 11042.48090545,  2159.33597148, 34066.1609094 ,
        11646.83178834,   874.98548929,  4020.66706965, 35913.0386546 ,
        -1034.71506651, 13963.49470486, 14840.86595147,  3395.11689253,
        12935.74119039, 11199.38639761, 11579.90265947, 16132.93772732,
        10183.88439249,  9888.34374983, 15157.35586536, 12377.94812939,
         4387.77863628,  3680.0942183 ,  5347.06219182, 13291.0174177 ,
         9158.24253865, 11935.82529104,  9522.10094863, 27668.10801212,
        12639.34008179,  3989.82506218, 38550.3600665 , 11191.86138788,
         8088.76475698, 11068.02157864, 10956.54972199, 15139.01708371,
        11077.7652618 , 13045.02707757,  5283.33522041, 25958.0327765 ,
         4962.43983078, 10543.57361001,  2709.95649343, 29007.79585973,
         6350.41196404,  3478.11303549,  2661.5079005 , 15990.91366368,
         7905.79980945, 10304.73937225,  9962.86575973,  5066.24762376,
        14869.35897203, 33752.1676117 ,  3761.88660755, 11521.18346955,
        24631.42819661, 14803.95189475,  1734.60861523, 10401.39588933,
         9202.60416666,  6288.03801508, 11838.14846799, 28871.88920869,
```

Result for testing data and error computation:-

```
[155]: # R squared value
       r2_test = metrics.r2_score(Y_test, test_data_prediction)
       print('R squared vale : ', r2_test)

       R squared vale :  0.7447273869684077

[156]: """Building a Predictive System"""

       input_data = (31,1,25.74,0,1,0)

       # changing input_data to a numpy array
       input_data_as_numpy_array = np.asarray(input_data)

[157]: # reshape the array
       input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

[158]: prediction = regressor.predict(input_data_reshaped)
       print(prediction)

       [3760.0805765]
       F:\Python\Python310\lib\site-packages\sklearn\base.py:465: Use
       e names
         warnings.warn(

[159]: print('The insurance cost is USD ', prediction[0])

       The insurance cost is USD  3760.0805764960496
```

Ridge Regression -

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values.

```
[160]: """Prediction using Ridge regression


       """

       # Ridge:
       from sklearn.linear_model import Ridge
       Ridge = Ridge()
       Ridge = Ridge.fit(X_train, Y_train)

[161]: # Prediction:
       y_pred = Ridge.predict(X_test)

[162]: # Scores:
       print(r2_score(Y_test, y_pred))

       0.7448008334274916
```

Lasso Regression-

Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

```
[163]: """Prediction using Lasso regression"""

# Lasso:
from sklearn.linear_model import Lasso
Lasso = Lasso()
Lasso = Lasso.fit(X_train, Y_train)

[164]: # Prediction:
y_pred = Lasso.predict(X_test)

[165]: # Scores:
print(r2_score(Y_test, y_pred))

0.7447245444913575
```

Random Forest Regressor-

The Random Forest Classifier algorithm is suitable for both classification and regression. The basic concept behind this algorithm is ensemble learning which means that combining multiple classifiers to solve a particular complex problem leads to the improvement in performance. Instead of creating a single decision tree, it creates multiple decision trees based on the dataset and the average is taken to predict the output.

```
[166]: """Prediction using Random Forest Regressor

"""

from sklearn.ensemble import RandomForestRegressor
RandomForestRegressor = RandomForestRegressor()
RandomForestRegressor = RandomForestRegressor.fit(X_train, Y_train)

[167]: # Prediction:
y_pred = RandomForestRegressor.predict(X_test)

[168]: # Scores:
print(r2_score(Y_test, y_pred))

0.8355660456707901
```

Result:-

```
[169]: input_data = (31,1,25.74,0,1,0)

[170]: # changing input_data to a numpy array
       input_data_as_numpy_array = np.asarray(input_data)

[171]: # reshape the array
       input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

[172]: prediction = RandomForestRegressor.predict(input_data_reshaped)
       print(prediction)

       [3764.1201605]
       F:\Python\Python310\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feat
       eature names
         warnings.warn(

[173]: print('The insurance cost is USD ', prediction[0])

       The insurance cost is USD  3764.1201605000074
```

CONCLUSION

An investigation into individual health insurance data is conducted using two regression models.Out of the four models used above, Random forest regressor has better accuracy. Any unneeded attribute was removed from each of the features. Premiums are determined by a person's health rather than the terms and conditions of another insurance provider. Some other algorithms can be employed to predict premiums based on data and improve accuracy. People and insurance companies can work together to deliver better and more health-focused coverage as a result of this.

REFERENCES

[1] Pesantez-Narvaez, J., Guillen, M., & Alcañiz, M. (2019). Predicting motor insurance claims using telematics data—XGBoost versus logistic regression. Risks, 7(2), 70

[2] Gupta, S., & Tripathi, P. (2016, February). An emerging trend of big data analytics with health insurance in India. In 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH) (pp. 64-69). IEEE.

[3] C. C. a. A. Semanskee, "Analysis of UnitedHealth Group's Premiums and Participation in ACA Marketplaces," 2016.