February, 2024

Health Insurance premium predictive analysis

Course: Quantitative Methods for Analytics
Instructor: Leslie Major

Prepared by
Battsetseg Tolya
Studend ID: A01407044

Table of contents

Introduction

Based on given hospital treatment charges of patients and other related information about patients, such as age, gender, BMI, region, number of dependents, and whether the patient is a smoker or not, health insurance companies can predict possible hospital charges of the patient based on factors mentioned above, so it can change its premium fee based on these factors and customize their premium fee to decrease their expenses. If predicted hospital charges tend to be higher, then health insurance companies can charge a higher premium fee when having a contract with the insured. When health insurance companies predict hospital charges and customize premium fees, it can save money and help the insurance company to work with a profit.

Understanding of data and data cleaning

The data has been taken from www.kaggle.com and the data has a following independent variables:

- age: age of primary beneficiary
- sex: insurance contractor gender: female or 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
- region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.
- charges: Individual medical costs billed by health insurance, which is the target variable.

In Python, I did the following commands to see my data:

```
print(data)
```

```
      age  gender     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
...   ...     ...     ...       ...    ...        ...          ...
1333   50    male  30.970         3     no  northwest  10600.54830
1334   18  female  31.920         0     no  northeast   2205.98080
1335   18  female  36.850         0     no  southeast   1629.83350
1336   21  female  25.800         0     no  southwest   2007.94500
1337   61  female  29.070         0    yes  northwest  29141.36030

[1338 rows x 7 columns]
```

```
data.info()
```

```
<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   gender    1338 non-null   object 
 2   bmi       1333 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
```

```
data.describe()
```

[59]:

	age	bmi	children	charges
count	1338.000000	1333.000000	1338.000000	1338.000000
mean	39.207025	30.658545	1.094918	13270.422265
std	14.049960	6.092785	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.315000	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.675000	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Based on the tables above, there are five BMI records has missing values. Handled the missing data by putting its averages in the place of empty cells.

[60]:

```
#handling missing values
count_nan = data.isnull().sum() # the number of missing values for every column
print(count_nan[count_nan > 0])
```

```
bmi    5
dtype: int64
```

[61]:

```
#fill in the missing values
data.fillna({'bmi':data['bmi'].mean()}, inplace=True)
```

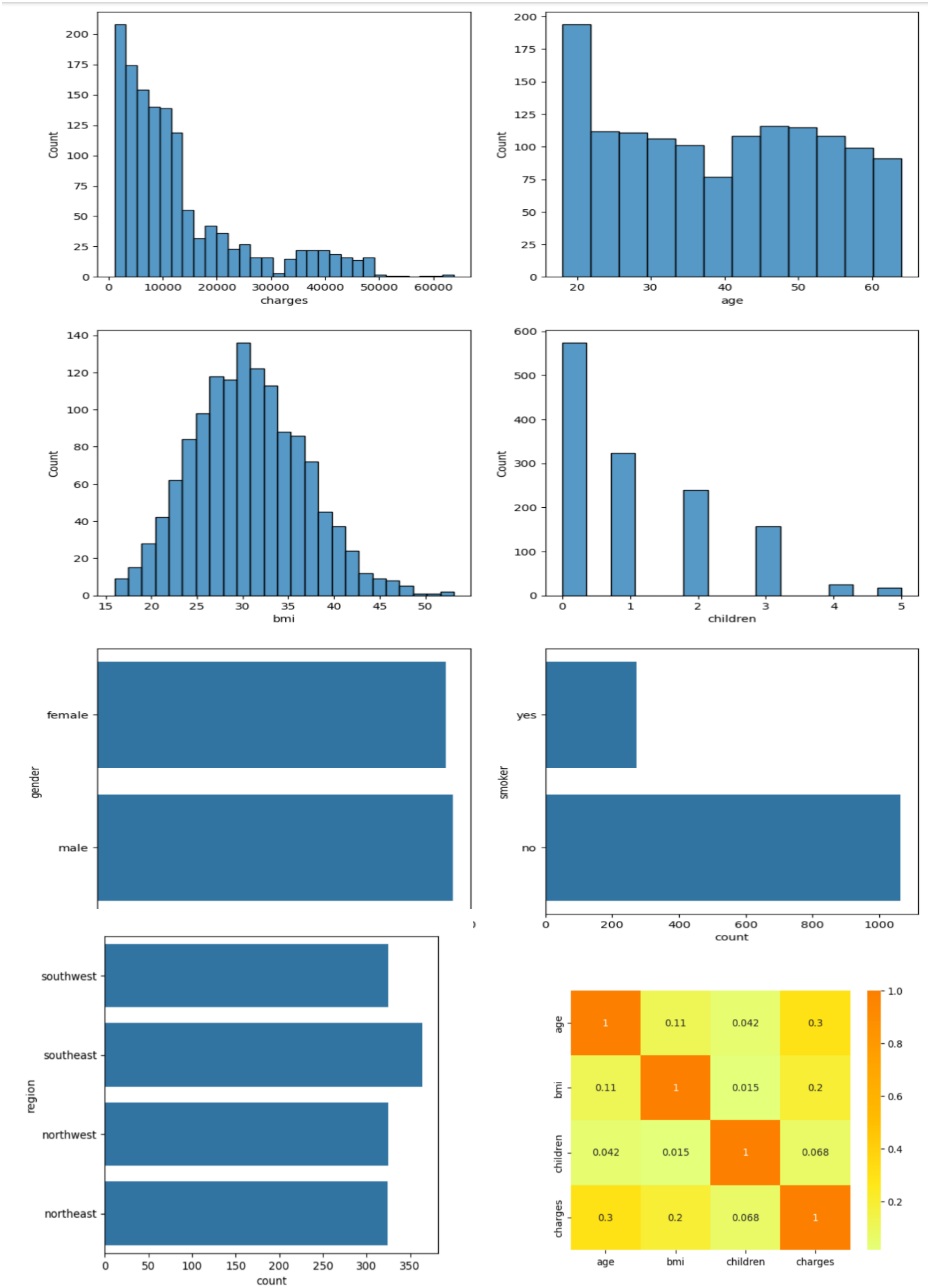
[62]:

```
#check how many values are missing (NaN) - after we filled in the NaN
count_nan = data.isnull().sum() # the number of missing values for every column
print(count_nan[count_nan > 0])
```

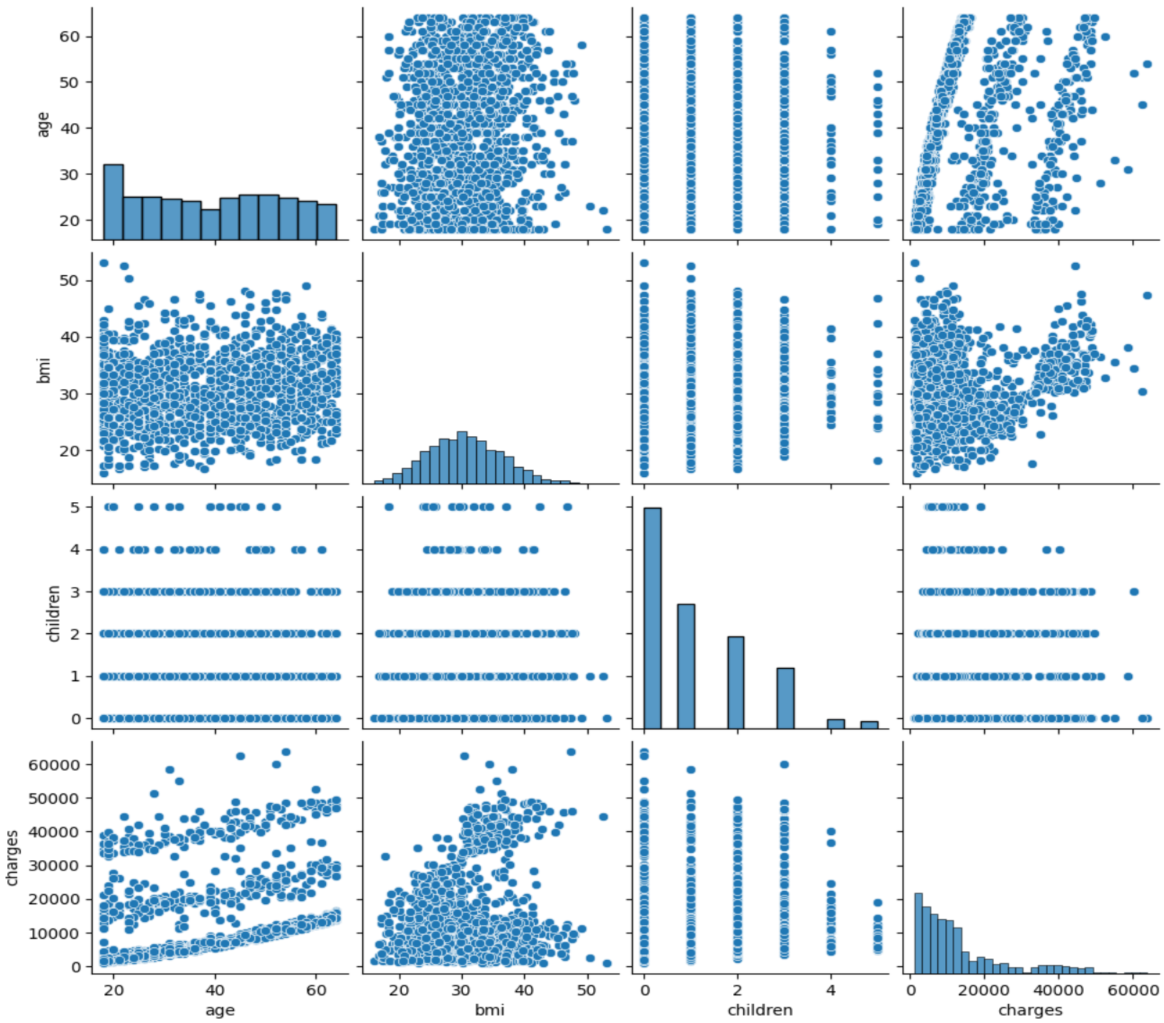
```
Series([], dtype: int64)
```

Exploratory Data Analysis

First the report visualized independent variables by count to get some insights. As in following graphs, there are almost equal number of women and men were counted, almost half of people had no children. There are more non-smokers detected as well. At the following page, correlation has been plotted between numerical variables.

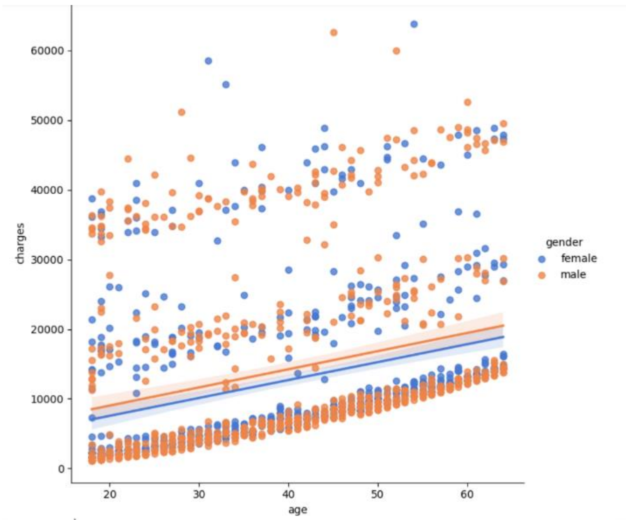

Count
charges
Count
age
Count
bmi
Count
children
female
male
gender
yes
no
smoker
count
southwest
southeast
northwest
northeast
region
count
age
bmi
children
charges
1
0.11
0.042
0.3
0.11
1
0.015
0.2
0.042
0.015
1
0.068
0.3
0.2
0.068
1

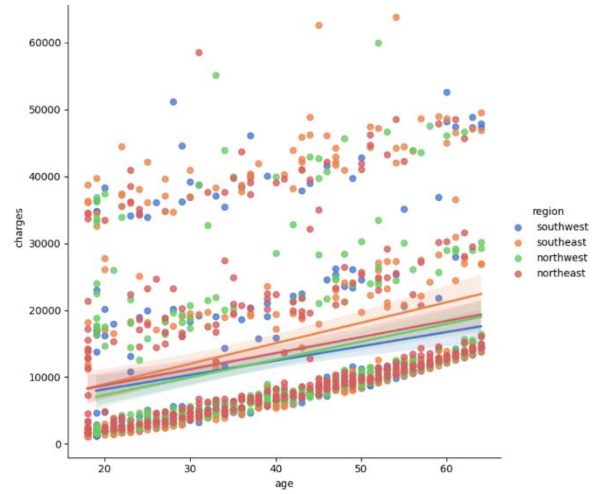
Next, the report further used pairplot between numeric variables in Python to see which numerical variables are affecting the charges more.

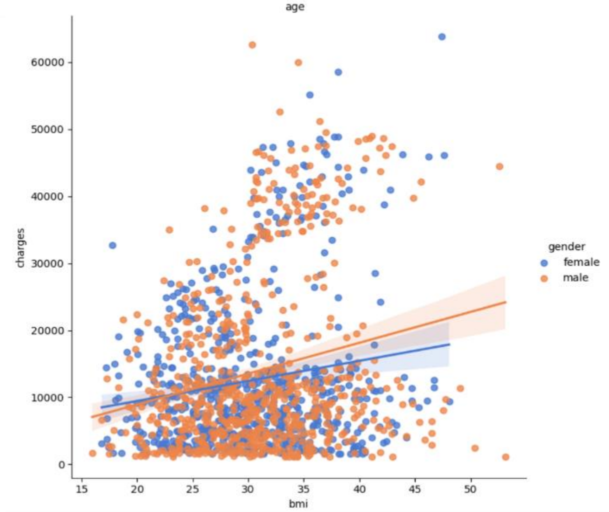


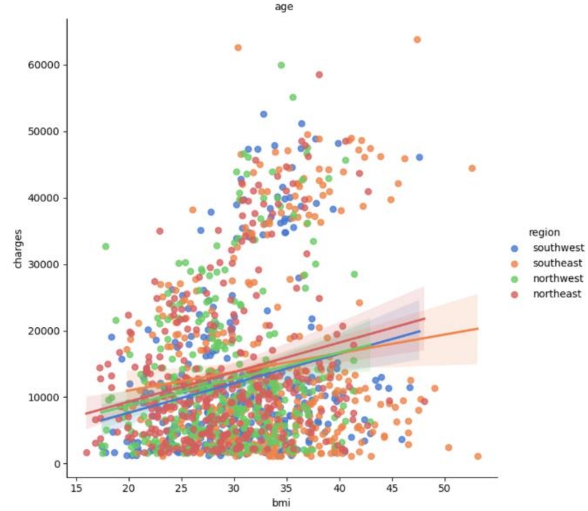
Looking at graphs above and correlation matrix, age vs charges and BMI vs charges has some patterns and higher correlations that can be further investigated by adding third categorical variables: smoker, region, and gender. Therefore, it added third categorical variables to see their effects on charges.

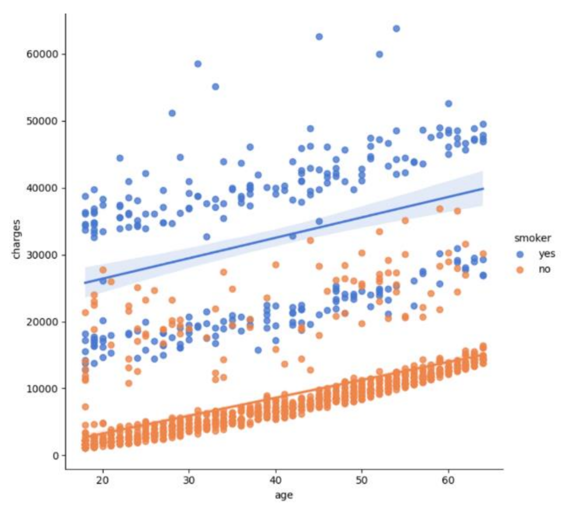
When region and gender were added to age vs charges and BMI vs charges plots, the report hasn't found clear linear distinction. However, smoker categorical variable was affecting these two plots and showed clear linear relationships which will improve our predictive model well. (Graphs are following)

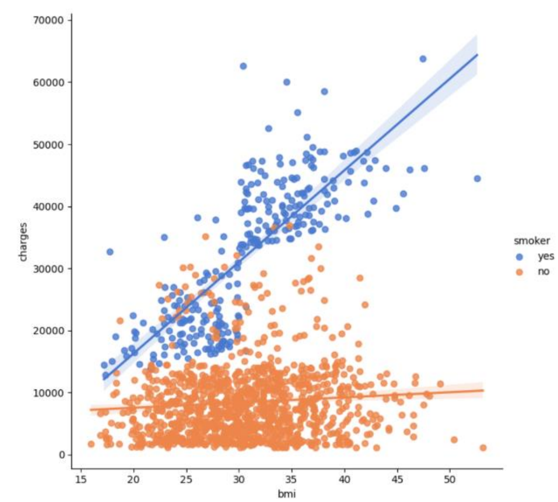












Medical bills can be significantly higher regardless of patient's age for smokers and for smokers as BMI increases medical costs were increased. For non-smokers as BMI increases, its medical costs haven't increased. Therefore, being a smoker can increase charges and having a higher BMI and smoking also can increase charges.

Comparison of predictive models

The report ran five regression models and see which one performs better to predict the y variable.

Each model's training and testing followed these steps:

1. Splits the data into training and testing sets. Testing portion was set at 33 percent on each model, which prevents overfit and using separate parts of the data for training and testing helps to ensure the reliability and generalizability of your machine learning models.
2. Scales the features using Standard Scaler.
3. Fit the model
4. Makes predictions on the training and testing sets.
5. Prints the training and testing scores.
6. Evaluates the model using Mean Squared Error and R-squared.
7. Visualizes the predicted vs. true values for both the training and test sets.

In order to train and test data all categorical variables were turned into dummy variables since regression algorithms only accept numerical variables. Following encoding was done on Python:

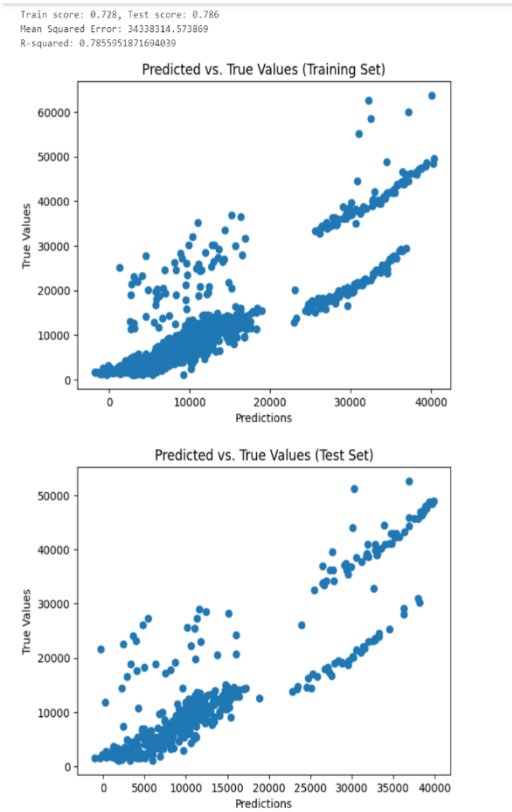
```
#sklearn one hot encoding: maps each category to 0 (cold) or 1 (hot)

#one hot encoder = ohe
#create ndarray for one hot encodoing (sklearn)
region = data.iloc[:,5:6].values #ndarray

## ohe for region
ohe = OneHotEncoder()

region = ohe.fit_transform(region).toarray()
region = pd.DataFrame(region)
region.columns = ['northeast', 'northwest', 'southeast', 'southwest']
print("Sklearn one hot encoder results for region:")
print(region[:10])


#sklearn label encoding: maps each category to a different integer

#create ndarray for label encodoing (sklearn)
gender = data.iloc[:,1:2].values
smoker = data.iloc[:,4:5].values

#label encoder = le
## le for gender
le = LabelEncoder()
gender[:,0] = le.fit_transform(gender[:,0])
gender = pd.DataFrame(gender)
gender.columns = ['gender']
le_gender_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Sklearn label encoder results for gender:")
print(le_gender_mapping)
print(gender[:10])

## le for smoker
le = LabelEncoder()
smoker[:,0] = le.fit_transform(smoker[:,0])
smoker = pd.DataFrame(smoker)
smoker.columns = ['smoker']
le_smoker_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Sklearn label encoder results for smoker:")
print(le_smoker_mapping)
print(smoker[:10])
```

All coded variables and machine learning ready data are printed in Python as following:

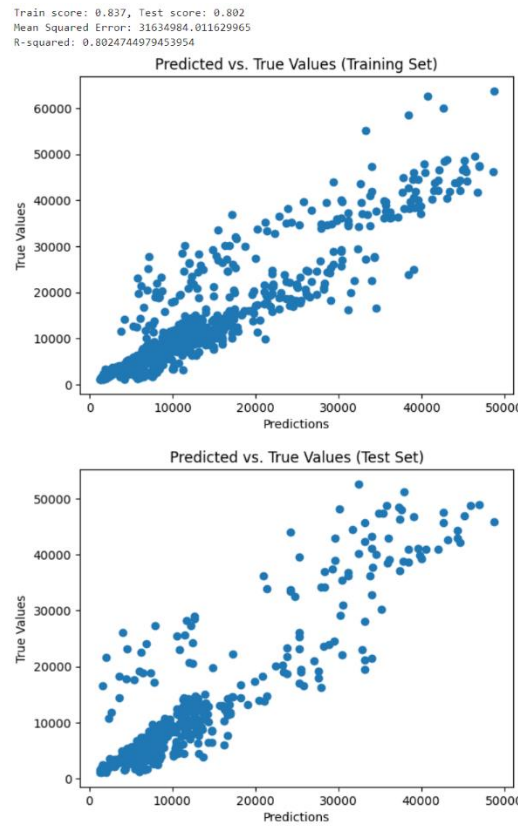
```
[26]: #putting the data together:

      ##take the numerical data from the original data
      X_num = data[['age', 'bmi', 'children']].copy()

      ##take the encoded data and add to numerical data
      X_final = pd.concat([X_num, region, gender, smoker], axis = 1)

      #define y as being the "charges column" from the original dataset
      y_final = data[['charges']].copy()

      #print all columns that will be used for testing
      print(X_final,y_final)

      #Test train split
      X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size = 0.33, random_state = 0 )
      #X_train, X_test, y_train, y_test = train_test_split(data[['age']], y_final, test_size = 0.33, random_state = 0 )
```

```
      age     bmi  children  northeast  northwest  southeast  southwest  \
0      19  27.900         0        0.0        0.0        0.0        1.0
1      18  33.770         1        0.0        0.0        1.0        0.0
2      28  33.000         3        0.0        0.0        1.0        0.0
3      33  22.705         0        0.0        1.0        0.0        0.0
4      32  28.880         0        0.0        1.0        0.0        0.0
...   ...     ...       ...        ...        ...        ...        ...
1333   50  30.970         3        0.0        1.0        0.0        0.0
1334   18  31.920         0        1.0        0.0        0.0        0.0
1335   18  36.850         0        0.0        0.0        1.0        0.0
1336   21  25.800         0        0.0        0.0        0.0        1.0
1337   61  29.070         0        0.0        1.0        0.0        0.0

     gender  smoker
0         0       1
1         1       0
2         1       0
3         1       0
4         1       0
...     ...     ...
1333      1       0
1334      0       0
1335      0       0
1336      0       0
1337      0       1

[1338 rows x 9 columns]            charges
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

[1338 rows x 1 columns]
```

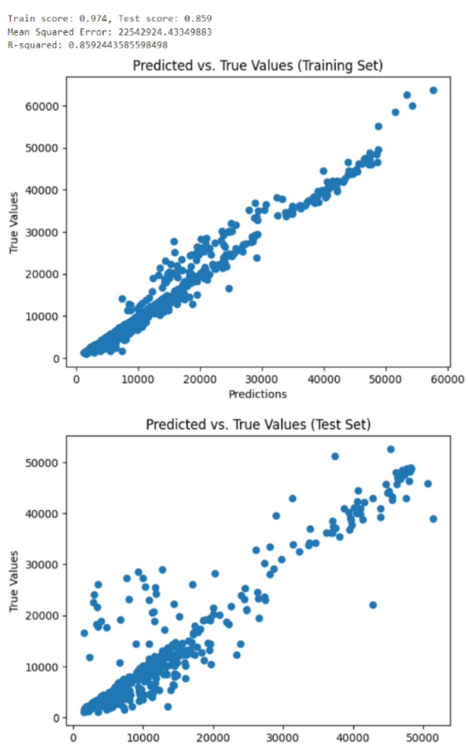
Next, the report trained and tested these models and evaluated on which one performs best to predict y variable. Each model has been visualized with train score and test score, mean squared error and r squared printed at the top of the plots.



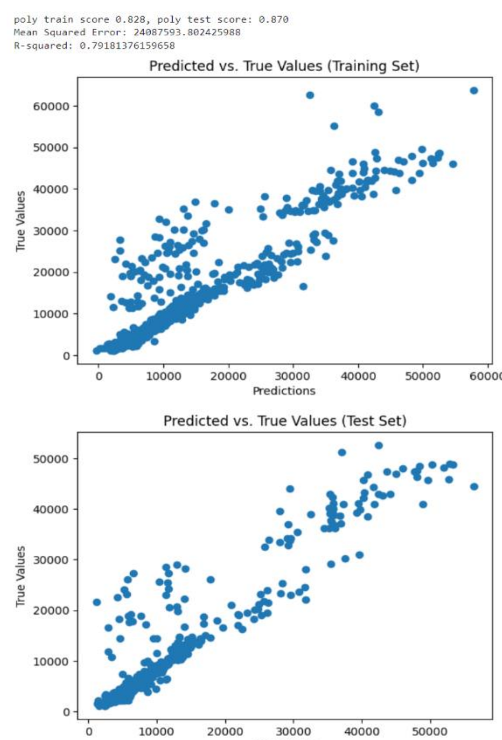
Linear regression



K nearest neighbors

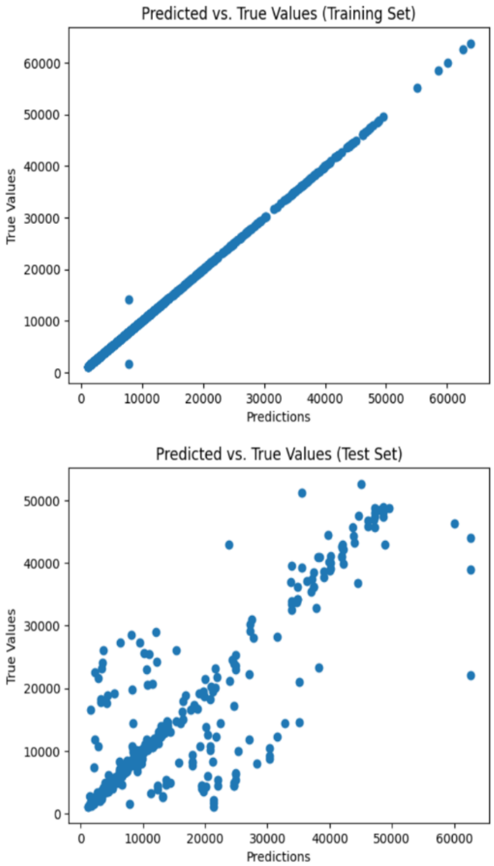


Random forest regression



Polynomial regression

```
Train score: 0.999, Test score: 0.701
Mean Squared Error: 47808139.881123416
R-squared: 0.7014910192828288
```



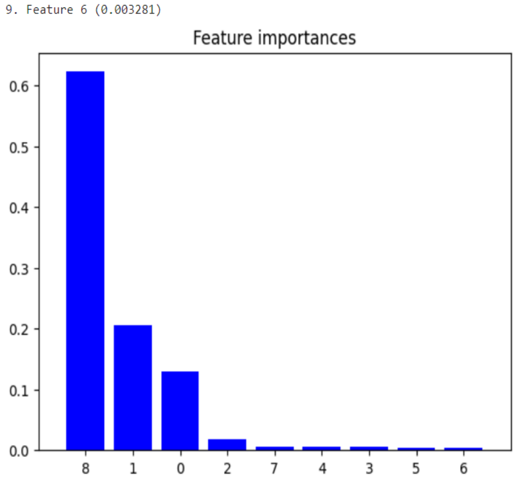
Decision Tree

	Model	Train Score	Test Score	Mean Squared Error	\
0	Linear Regression	0.728	0.786	3.433831e+07	
1	KNN	0.837	0.802	3.163498e+07	
2	Random Forest	0.974	0.859	2.254292e+07	
3	Polynomial Regression	0.828	0.870	2.408759e+07	
4	Decision Tree	0.999	0.701	4.780814e+07	

	R-squared
0	0.7856
1	0.8024
2	0.8592
3	0.7918
4	0.7014

At the chart above, it summarized all scores in comparison between models. It looks like random forest regression best predicts y variable at 86 percent with less error. Another good predicter can be polynomial regression model but it has higher errors, lower r squared, and lower training score.

```
Feature ranking:
1. Feature 8 (0.623069)
2. Feature 1 (0.205051)
3. Feature 0 (0.130234)
4. Feature 2 (0.018765)
5. Feature 7 (0.005492)
6. Feature 4 (0.005227)
7. Feature 3 (0.005140)
8. Feature 5 (0.003740)
9. Feature 6 (0.003281)
```



On random forest model, the report visualized features importance and ranked it. Since Python starts counting from zero, #8 is the smoker feature, #1 is the BMI feature, #0 is the age, #2 is number of dependents feature (Page 8) which affected y variable prediction higher than other variables such as region and gender.

Executive summary

The aim of this analysis was to investigate the predictability of medical costs (denoted as 'y' variable) based on various factors including age, BMI, gender, geographical regions, number of dependents, and smoking habits. The study explores how accurately these factors can forecast medical expenses, which could assist health insurance companies in effectively predicting costs and tailoring premium fees accordingly, thus optimizing cost efficiency.

By employing predictive modeling techniques, particularly Random Forest Regression, it was determined that the model achieved a prediction accuracy of 86 percent. This indicates a high level of predictability for medical costs based on the specified input variables. Therefore, health insurance companies can utilize these findings to enhance their risk assessment strategies, better manage financial resources, and offer more customized premium fees to their customers.

In conclusion, the insights derived from this analysis provide valuable guidance for health insurance companies to make informed decisions, optimize their operations, and ultimately improve the overall efficiency of their business processes.

Appendix

1. Data: https://www.kaggle.com/datasets/mirichoi0218/insurance/data

I hasn't followed the codes thoroughly, but took some ideas from the codes listed and ran Python codes to best illustrate the data and best illustrate prediction models.

2. Python codes: Final project in Python.html