Introduction

The medical insurance dataset contains information about a number of factors that can affect medical expenses, including age, sex, BMI, smoking status, number of children, and region. Our job here is to derive insights from the datasets that contribute to higher insurance costs and help the company make more informed decisions regarding pricing and risk assessment.

Importing libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Importing dataset

In [2]:
```
Dataset = pd.read_csv("D:\Medical_insurance.csv")
```

Data preprocessing

In [3]:
```
Dataset.head(10)
```

Out[3]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960
7	37	female	27.740	3	no	northwest	7281.50560
8	37	male	29.830	2	no	northeast	6406.41070
9	60	female	25.840	0	no	northwest	28923.13692

In [4]:
```
Dataset.shape
```

Out[4]:
```
(2772, 7)
```

In [5]:
```
Dataset.isnull().sum()
```

Out[5]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

Now that we know,our dataset contains no null values and consists of 2772 records of data across 7 columns we proceed to the next steps

In [6]:
```
distinct_values = Dataset['region'].unique()
```

```
In [6]: distinct_values = Dataset['region'].unique()
        distinct_values
```

```
Out[6]: array(['southwest', 'southeast', 'northwest', 'northeast'], dtype=object)
```

The dataset consists of two string dtypes which needs to be encoded before splitting into training and testing set.

```
In [7]: from sklearn.compose import ColumnTransformer
        from sklearn.preprocessing import OneHotEncoder
        from sklearn.preprocessing import LabelEncoder
```

```
In [8]: le = LabelEncoder()
        Dataset['sex'] = le.fit_transform(Dataset['sex'])
        Dataset['smoker'] = le.fit_transform(Dataset['smoker'])
        Dataset.head()
```

Out[8]:

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

```
In [9]: ct = ColumnTransformer(transformers = [('encode',OneHotEncoder(),[5])], remainder = 'passthrough')
        d = ct.fit_transform(Dataset)
        d=pd.DataFrame(d)
        d
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	0.0	1.0	19.0	0.0	27.900	0.0	1.0	16884.92400
1	0.0	0.0	1.0	0.0	18.0	1.0	33.770	1.0	0.0	1725.55230
2	0.0	0.0	1.0	0.0	28.0	1.0	33.000	3.0	0.0	4449.46200
3	0.0	1.0	0.0	0.0	33.0	1.0	22.705	0.0	0.0	21984.47061
4	0.0	1.0	0.0	0.0	32.0	1.0	28.880	0.0	0.0	3866.85520
...	...	...	...	...	...	...	...	...	...	...
2767	0.0	0.0	1.0	0.0	47.0	0.0	45.320	1.0	0.0	8569.86180
2768	0.0	0.0	0.0	1.0	21.0	0.0	34.600	0.0	0.0	2020.17700
2769	0.0	1.0	0.0	0.0	19.0	1.0	26.030	1.0	1.0	16450.89470
2770	0.0	1.0	0.0	0.0	23.0	1.0	18.715	0.0	0.0	21595.38229
2771	0.0	0.0	0.0	1.0	54.0	1.0	31.600	0.0	0.0	9850.43200

2772 rows × 10 columns

The new one-hot encoded columns are added to the left side, and the original columns are appended on the right due to remainder='passthrough'. So, the order should be: one-hot encoded columns for 'region' (0, 1, 2, 3), followed by the other columns (age, sex, bmi, children, smoker, charges), maintaining the original order.

```
In [10]: X = d.iloc[:, 0:9]
         Y = d.iloc[:, -1]
         # X being the features and Y being the target.
```

```
In [11]: 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 = 0)
         # Splitting into training and test set, where training set contains 80% of data.
```

Now that we are done with the data preprocessing, we try implementing different kinds of regression models to predict the medical charges and find the accurate one using performance metrics.

Multiple Regression

```
In [12]: from sklearn.linear_model import LinearRegression
         reg = LinearRegression()
         reg.fit(X_train,Y_train)
```

```
Out[12]: ▾ LinearRegression
         LinearRegression()
```

```
In [13]: reg.coef_
```

Out[13]:
```
array([ 9.21246915e+15,  9.21246915e+15,  9.21246915e+15,  9.21246915e+15,
        2.53000000e+02, -1.41148438e+02,  3.16000000e+02,  5.42593750e+02,
        2.40674141e+04])
```

In [14]:
```
reg.intercept_
```

Out[14]:
```
-9212469151113900.0
```

In [15]:
```
Y_pred = reg.predict(X_test)
np.set_printoptions(precision = 2)
Y_pred_reshaped = Y_pred.reshape(-1, 1)
Y_test_reshaped = Y_test.values.reshape(-1, 1)

print(np.concatenate((Y_pred_reshaped, Y_test_reshaped), axis=1))
```

```
[[ 1596.    2221.56]
 [  -96.   21595.38]
 [ 7598.    5327.4 ]
 ...
 [ 9572.    9301.89]
 [13286.    7650.77]
 [ 4932.    6640.54]]
```

In [16]:
```
""" We try to predict the medical charges for a male belonging to the southeast region,with 23 years of age,
with a bmi of 32 with 0 kids,whose also a smoker """

reg.predict([[0.0,0.0,1.0,0.0,23,1.0,32,0,1.0]])
```

Out[16]:
```
array([27842.])
```

The R2 score indicates the percentage of the target variable's variance that can be predicted by the features in our regression model. A higher R2 score suggests that our model is better at explaining the variability in the target variable.

In [17]:
```
from sklearn.metrics import r2_score
r2_score(Y_test,Y_pred)
```

Out[17]:
```
0.7459911226805709
```

However, we use Mean Absolute Error to evaluate the performance of your model. These metrics provide information about the average magnitude of errors in your predictions.

In [18]:
```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(Y_test, Y_pred)
```

Out[18]:
```
4177.043841637839
```

The MAE value of 4177.04 means that, on average, our model's predictions are off by approximately $4177.04 in terms of the medical charges, In the context of our regression model, a lower MAE is generally better.

Polynomial Regression

In [19]:
```
from sklearn.preprocessing import PolynomialFeatures
```

In [20]:
```
# fitting polynomial features of degree 2
pf = PolynomialFeatures(degree = 2)
X2_Train = pf.fit_transform(X_train)
X2_test = pf.fit_transform(X_test)
```

In [21]:
```
Lr2 = LinearRegression()
Lr2.fit(X2_Train,Y_train)
Y_pred2 = Lr2.predict(X2_test)
```

In [22]:
```
"""We try to predict the medical charges for a male belonging to the southeast region,with 23 years of age,with
   with 0 kids,
   whose also a smoker,
   which is the same case as before, to compare model accuracy"""

Lr2.predict(pf.fit_transform([[0.0,0.0,1.0,0.0,23,1.0,32,0,1.0]]))
```

Out[22]:
```
array([28285.1])
```

In [23]:
```
Lr2.coef_
```

```
Out[23]: array([ 1.28e+13,  2.16e+12, -7.58e+12, -1.41e+13, -2.37e+12,  6.12e+13,
               -1.81e+12, -9.47e+13, -2.39e+12,  1.07e+11, -9.43e+12, -3.76e+12,
                9.00e+11,  1.79e+12, -6.12e+13, -1.73e+11,  9.47e+13,  2.39e+12,
                1.13e+11,  3.11e+11, -1.35e+11, -1.08e+10, -6.12e+13, -1.73e+11,
                9.47e+13,  2.39e+12,  1.13e+11,  6.87e+12,  1.03e+11, -6.12e+13,
               -1.73e+11,  9.47e+13,  2.39e+12,  1.13e+11, -4.90e+12, -6.12e+13,
               -1.73e+11,  9.47e+13,  2.39e+12,  1.13e+11,  4.28e+00,  3.42e+01,
                4.57e-01, -5.95e+00, -3.89e+00,  1.98e+12, -1.75e+01, -2.26e+02,
               -5.38e+02, -6.77e+00,  9.09e+00,  1.51e+03, -8.34e+01, -2.93e+02,
               -2.20e+11])
```

```
In [24]: Lr2.intercept_
```

```
Out[24]: -5570980671982.907
```

Checking R2 score.

```
In [25]: r2_score(Y_test, Y_pred2)
```

```
Out[25]: 0.8399747496882675
```

Checking Mean Absolute error for polynomial regression.

```
In [26]: mean_absolute_error(Y_test, Y_pred2)
```

```
Out[26]: 2815.430853217793
```

Decision Tree Regression

```
In [27]: from sklearn.tree import DecisionTreeRegressor
```

```
In [28]: regressor = DecisionTreeRegressor(random_state = 0)
```

```
In [29]: regressor.fit(X_train,Y_train)
```

```
Out[29]: ▾       DecisionTreeRegressor
         DecisionTreeRegressor(random_state=0)
```

```
In [30]: Y_pred3 = regressor.predict(X_test)
```

```
In [31]: """predicting the medical charges for a male belonging to the southeast region,with 23 years of age,with a bmi
            with 0 kids,
            whose is also a smoker"""

         regressor.predict([[0.0,0.0,1.0,0.0,23,1.0,32,0,1.0]])
```

```
Out[31]: array([33732.69])
```

Checking R2 score.

```
In [32]: r2_score(Y_test, Y_pred3)
```

```
Out[32]: 0.9486444975108179
```

Checking MAE for Decision tree regression.

```
In [33]: mean_absolute_error(Y_test, Y_pred3)
```

```
Out[33]: 555.9411333153155
```

Random Forest Regression

```
In [34]: from sklearn.ensemble import RandomForestRegressor
         regressor2 = RandomForestRegressor(n_estimators = 10,random_state = 0)
         regressor2.fit(X_train,Y_train)
```

```
Out[34]: ▾              RandomForestRegressor
         RandomForestRegressor(n_estimators=10, random_state=0)
```

```
In [35]: Y_pred4 = regressor2.predict(X_test)
```

```
In [36]: """predicting the medical charges for a male belonging to the southeast region,with 23 years of age,with a bmi
            with 0 kids,
   whose is also a smoker"""

regressor2.predict([[0.0,0.0,1.0,0.0,23,1.0,32,0,1.0]])
```

Out[36]:
```
array([34286.91])
```

Checking R2 score

In [37]:
```
r2_score(Y_test, Y_pred4)
```

Out[37]:
```
0.9533192794278544
```

Checking MAE for Random forest regression.

In [38]:
```
mean_absolute_error(Y_test, Y_pred4)
```

Out[38]:
```
1238.4003685541356
```

Finally, we see that out of all the regression models, Decision tree regression and Random forest regression have the least Mean absolute arror and highest R2 scores, therefore if our aim was to find a model in which the proportion of the variance in the medical charges explained by the models is more, then higher R2 suggests a better fit. However, our aim is to attain higher prediction accuracy i.e., closeness of predictions to actual values. Therefore, Decision tree regression is the better suited model.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js