Machine_Learning_Medical_Insurance

December 10, 2022

1 A Machine Learning Example for the Insurance Sector

Dataset: https://www.kaggle.com/datasets/mirichoi0218/insurance

[1]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

sns.set_style("darkgrid")
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (15, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'

import warnings
warnings.simplefilter(action='ignore')
```

2 Importing Data

[2]:
```
insurance_df = pd.read_csv('./insurance.csv')
insurance_df.head()
```

[2]:

	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

[3]:
```
insurance_df.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   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
```

[4]:
```
insurance_df[insurance_df.duplicated(keep=False)]
```

[4]:

	age	sex	bmi	children	smoker	region	charges
195	19	male	30.59	0	no	northwest	1639.5631
581	19	male	30.59	0	no	northwest	1639.5631

There is one duplicated value. It is better to drop it.

[5]:
```
insurance_df
```

[5]:

	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
…	…	…	…	…	…	…	…
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]

[6]:
```
insurance_df_no_dupl = insurance_df.drop_duplicates()
insurance_df_no_dupl
```

[6]:

	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
…	…	…	…	…	…	…	…
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

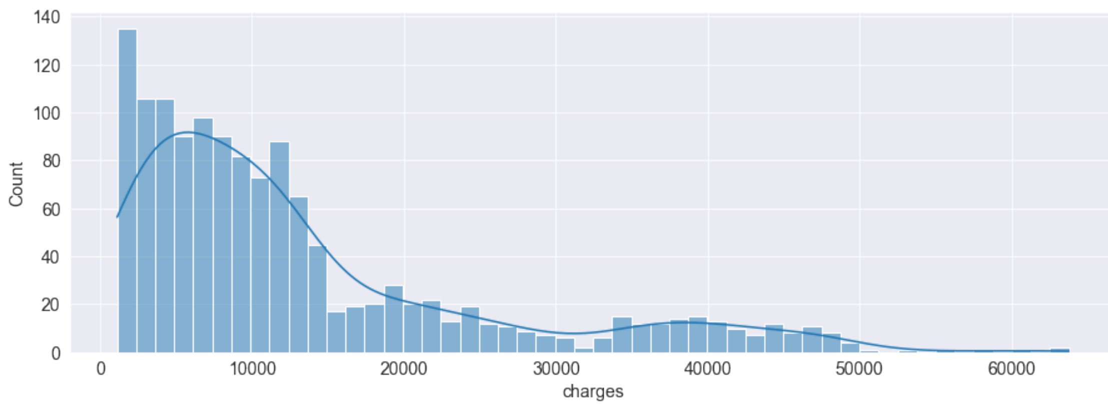
```
1336    21   female   25.800          0      no   southwest    2007.94500
1337    61   female   29.070          0     yes   northwest   29141.36030

[1337 rows x 7 columns]
```

3 Exploratory Data Analysis

3.1 Exploring the Target Variable 'charges'

Before starting with the other variables it is important to analyze the distribution of the 'charges' feature.

[7]:
```
sns.histplot(data=insurance_df_no_dupl, x='charges', bins=50 ,kde=True);
```



[8]:
```
insurance_df_no_dupl.charges.describe()
```

[8]:
```
count     1337.000000
mean     13279.121487
std      12110.359656
min       1121.873900
25%       4746.344000
50%       9386.161300
75%      16657.717450
max      63770.428010
Name: charges, dtype: float64
```

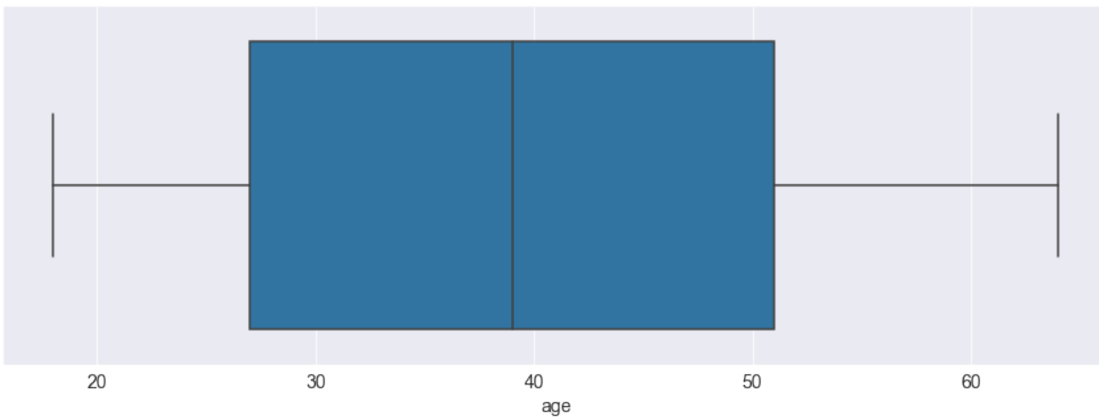
The distribution is right-skewed.
The majority of the fees are concentrated in the range between $0 and $10000.

3.2 Exploring the Variable Age

[9]:
```
insurance_df_no_dupl.age.describe()
```

[9]:
```
count    1337.000000
mean       39.222139
std        14.044333
min        18.000000
25%        27.000000
50%        39.000000
75%        51.000000
max        64.000000
Name: age, dtype: float64
```

[10]:
```
sns.boxplot(data=insurance_df_no_dupl, x='age');
```



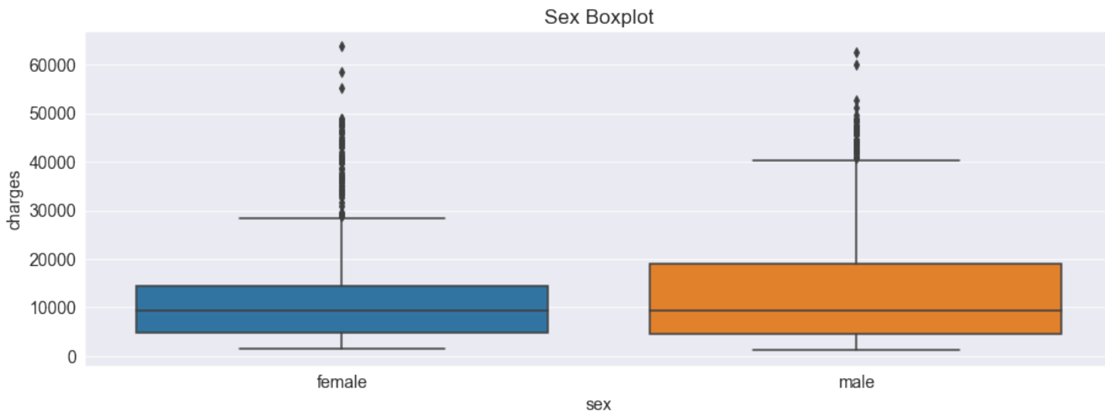
3.3 Exploring the Sex Variable

[11]:
```
insurance_df_no_dupl.sex.value_counts()
```

[11]:
```
male      675
female    662
Name: sex, dtype: int64
```

The distribution of sex is balanced. What is the amount of charges paid by sex?

[12]:
```
sns.boxplot(data=insurance_df_no_dupl, x='sex', y='charges')
plt.title('Sex Boxplot');
```



Males seem to pay quite more in charges than females.

[13]:
```
insurance_df_no_dupl.groupby('sex')['charges'].mean()
```

[13]:
```
sex
female    12569.578844
male      13974.998864
Name: charges, dtype: float64
```

[14]:
```
females_charges = insurance_df_no_dupl.groupby('sex')['charges'].mean()[0]
male_charges = insurance_df_no_dupl.groupby('sex')['charges'].mean()[1]
male_charges-females_charges
```

[14]:
```
1405.4200199276147
```

Indeed, on average, males pay $1405.42 more.

A more detailed description of the variable "charges" described by gender.

[15]:
```
insurance_df_no_dupl[insurance_df_no_dupl.sex=='male'].charges.describe()
```

[15]:
```
count      675.000000
mean     13974.998864
std      12971.958663
min       1121.873900
25%       4654.022675
50%       9377.904700
75%      19006.685500
max      62592.873090
Name: charges, dtype: float64
```

[16]:
```
insurance_df_no_dupl[insurance_df_no_dupl.sex=='female'].charges.describe()
```

[16]:
```
count      662.000000
mean     12569.578844
std      11128.703801
min       1607.510100
25%       4885.158700
50%       9412.962500
75%      14454.691825
max      63770.428010
Name: charges, dtype: float64
```

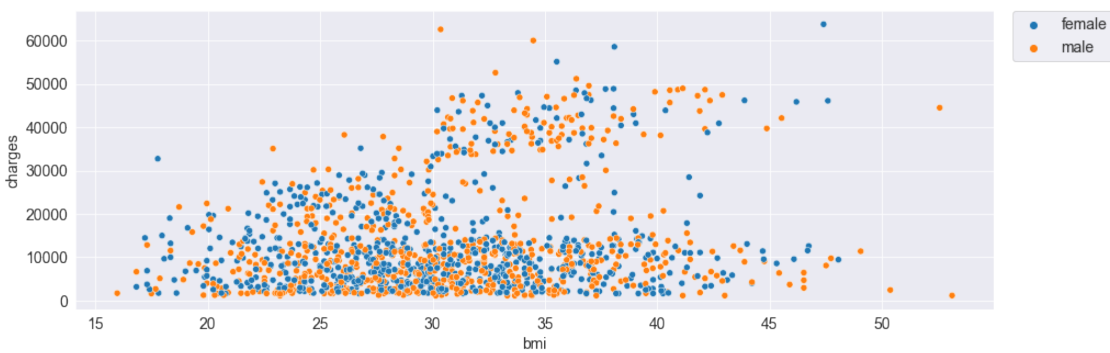
3.4 Exploring the BMI Variable

[17]:
```
insurance_df_no_dupl.bmi.describe()
```

[17]:
```
count    1337.000000
mean       30.663452
std         6.100468
min        15.960000
25%        26.290000
50%        30.400000
75%        34.700000
max        53.130000
Name: bmi, dtype: float64
```

Is there a correlation with "charges" feature?

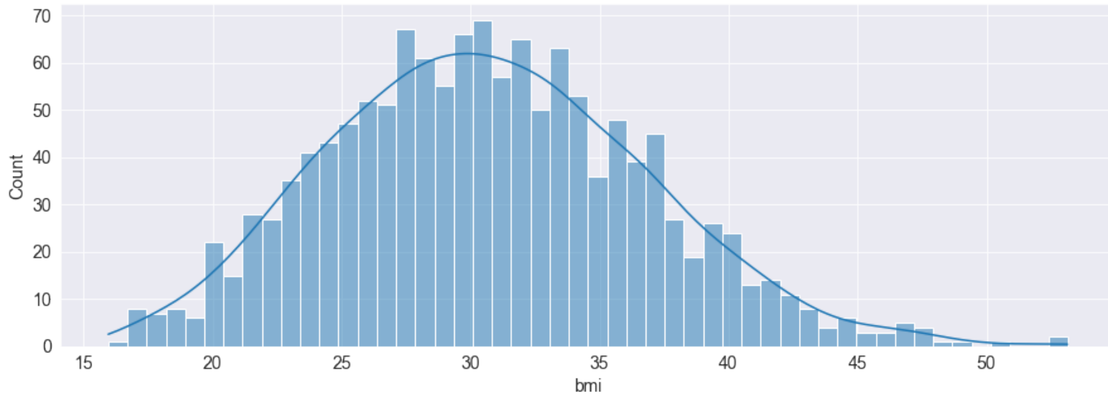
[18]:
```
sns.scatterplot(data=insurance_df_no_dupl, x='bmi', y='charges', hue='sex')
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0);
```



It does not seem to be a strong relationship, but it is positive for sure; moreover there are not two well defined clusters based on sex.

Checking the distribution of the variable BMI.

[19]:
```
sns.histplot(data=insurance_df_no_dupl, x='bmi', bins=50, kde=True);
```



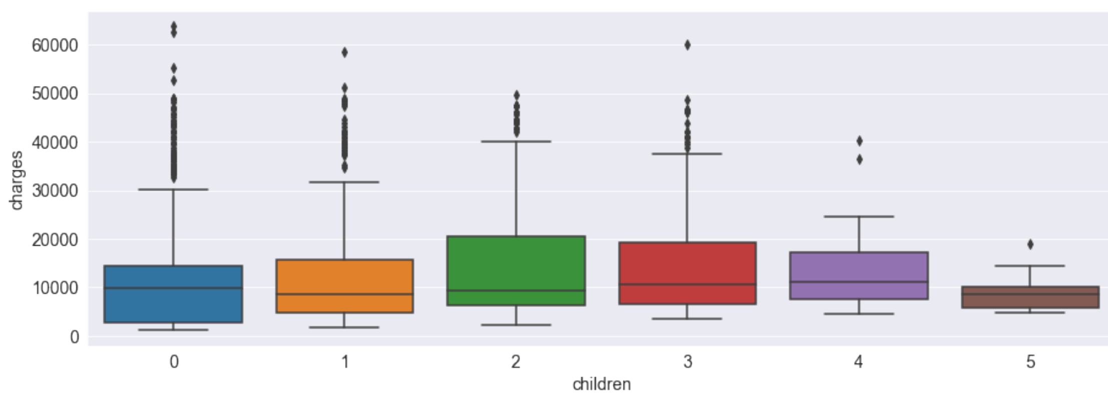
The “bmi” variable seems to be normally distributed.

3.5 Exploring the Children Variable

[20]:
```
insurance_df_no_dupl.children.describe()
```

[20]:
```
count    1337.000000
mean        1.095737
std         1.205571
min         0.000000
25%         0.000000
50%         1.000000
75%         2.000000
max         5.000000
Name: children, dtype: float64
```

[21]:
```
sns.boxplot(data=insurance_df_no_dupl, x='children', y='charges');
```



[22]:
```
insurance_df_no_dupl.groupby('children')['charges'].mean()
```

[22]:
```
children
0    12384.695344
1    12731.171832
2    15073.563734
3    15355.318367
4    13850.656311
5     8786.035247
Name: charges, dtype: float64
```

On average, who has 2 or 3 children pays spends more money in charges.

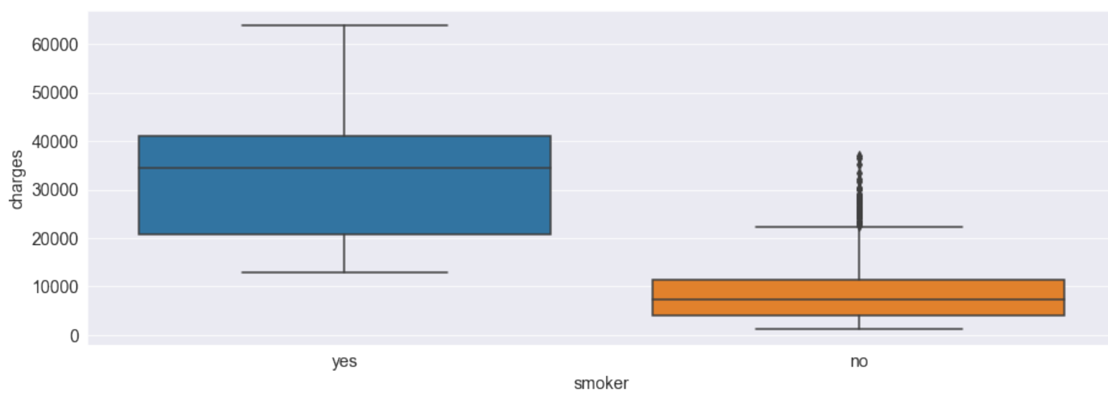
3.6 Exploring the Smoking Variable

[23]:
```
insurance_df_no_dupl.smoker.value_counts()
```

[23]:
```
no     1063
yes     274
Name: smoker, dtype: int64
```

There are not many smokers in the dataset.

[24]:
```
sns.boxplot(data=insurance_df_no_dupl, x='smoker', y='charges');
```



In general, smokers pay much more in insurance charges than no-smokers.

[25]:
```
insurance_df_no_dupl[insurance_df_no_dupl.smoker=='yes'].charges.describe()
```

[25]:
```
count      274.000000
mean     32050.231832
std      11541.547176
min      12829.455100
25%      20826.244213
```

```
50%      34456.348450
75%      41019.207275
max      63770.428010
Name: charges, dtype: float64
```

[26]:
```
insurance_df_no_dupl[insurance_df_no_dupl.smoker=='no'].charges.describe()
```

[26]:
```
count     1063.000000
mean      8440.660307
std       5992.973800
min       1121.873900
25%       3988.883500
50%       7345.726600
75%      11363.019100
max      36910.608030
Name: charges, dtype: float64
```

What is, on average, the amount paid by smokers and non-smokers?

[27]:
```
insurance_df_no_dupl.groupby('smoker')['charges'].mean()
```

[27]:
```
smoker
no      8440.660307
yes    32050.231832
Name: charges, dtype: float64
```

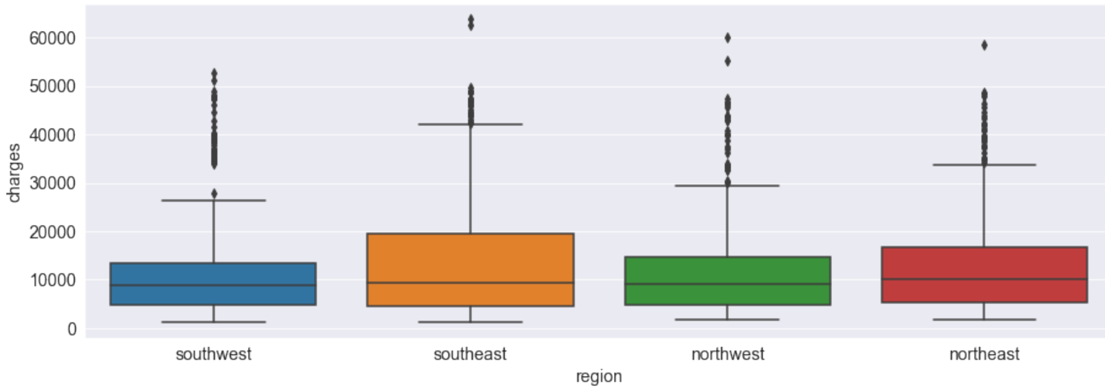
3.7 Exploring the Region Variable

[28]:
```
insurance_df_no_dupl.region.value_counts()
```

[28]:
```
southeast    364
southwest    325
northwest    324
northeast    324
Name: region, dtype: int64
```

Regions are almost perfectly balanced, only the southeast region appears more times.

[29]:
```
sns.boxplot(data=insurance_df_no_dupl, x='region', y='charges');
```



It does not seem to be a large difference between them. Only the southeast region has more outliers.

4 Feature Engineering

In this section are performed all the necessary steps to transform variables that will be used in the machine learning model.

4.1 Defining Inputs and Target

[30]:
```
input_df = insurance_df_no_dupl.drop(columns='charges')
target_df = insurance_df_no_dupl.charges
```

Showing the results

[31]:
```
input_df
```

[31]:
```
      age     sex     bmi  children smoker     region
0      19  female  27.900         0    yes  southwest
1      18    male  33.770         1     no  southeast
2      28    male  33.000         3     no  southeast
3      33    male  22.705         0     no  northwest
4      32    male  28.880         0     no  northwest
…     …       …       …         …      …          …
1333   50    male  30.970         3     no  northwest
1334   18  female  31.920         0     no  northeast
1335   18  female  36.850         0     no  southeast
1336   21  female  25.800         0     no  southwest
1337   61  female  29.070         0    yes  northwest

[1337 rows x 6 columns]
```

[32]:
```
target_df
```

[32]:
```
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: 1337, dtype: float64
```

4.2 Creating a Variable to Cluster the BMI

It will be added a new feature to cluster observations on their BMI values.
These are the different ranges:
- If BMI is less than 18.5, it falls within the underweight range.
- If BMI is 18.5 to 24.9, it falls within the normal or Healthy Weight range.
- If BMI is 25.0 to 29.9, it falls within the overweight range.
- If BMI is 30.0 or higher, it falls within the obese range.

[33]:
```python
def bmi_estimator(column):
    if column < 18.5:
        return 'underweight'
    elif (column >= 18.5) and (column<=24.9):
        return 'healthy weight'
    elif (column>=25) and (column<=29.9):
        return 'overweight'
    else:
        return 'obese'
```

[34]:
```python
input_df['bmi_class']=input_df.bmi.map(bmi_estimator)
input_df
```

[34]:

	age	sex	bmi	children	smoker	region	bmi_class
0	19	female	27.900	0	yes	southwest	overweight
1	18	male	33.770	1	no	southeast	obese
2	28	male	33.000	3	no	southeast	obese
3	33	male	22.705	0	no	northwest	healthy weight
4	32	male	28.880	0	no	northwest	overweight
…	…	…	…	…	…	…	…
1333	50	male	30.970	3	no	northwest	obese
1334	18	female	31.920	0	no	northeast	obese
1335	18	female	36.850	0	no	southeast	obese
1336	21	female	25.800	0	no	southwest	overweight
1337	61	female	29.070	0	yes	northwest	overweight

```
[1337 rows x 7 columns]
```

4.3 Encoding Sex and Smoking Variables

[35]:
```python
sex_dict = {'male':0, 'female':1}
smoker_dict = {'no':0, 'yes':1}

input_df['sex'] = input_df.sex.map(sex_dict)
input_df['smoker'] = input_df.smoker.map(smoker_dict)
```

4.4 Encoding the Region Variable

It is used the one-hot encoding procedure.

[36]:
```python
columns_to_encode = ['region', 'bmi_class']
```

[37]:
```python
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse=False, handle_unknown='ignore').
↪fit(input_df[columns_to_encode])

encoded_cols = list(encoder.get_feature_names_out(columns_to_encode))

input_df[encoded_cols] = encoder.transform(input_df[columns_to_encode])
```

[38]:
```python
input_df = input_df.drop(columns='region')
input_df
```

[38]:
```
      age  sex     bmi  children  smoker       bmi_class  region_northeast  \
0      19    1  27.900         0       1      overweight               0.0
1      18    0  33.770         1       0           obese               0.0
2      28    0  33.000         3       0           obese               0.0
3      33    0  22.705         0       0  healthy weight               0.0
4      32    0  28.880         0       0      overweight               0.0
…     …   …    …       …         …               …                …
1333   50    0  30.970         3       0           obese               0.0
1334   18    1  31.920         0       0           obese               1.0
1335   18    1  36.850         0       0           obese               0.0
1336   21    1  25.800         0       0      overweight               0.0
1337   61    1  29.070         0       1      overweight               0.0

      region_northwest  region_southeast  region_southwest  \
0                  0.0               0.0               1.0
1                  0.0               1.0               0.0
2                  0.0               1.0               0.0
3                  1.0               0.0               0.0
4                  1.0               0.0               0.0
```

```
…                  …                 …                 …
1333               1.0               0.0               0.0
1334               0.0               0.0               0.0
1335               0.0               1.0               0.0
1336               0.0               0.0               1.0
1337               1.0               0.0               0.0

      bmi_class_healthy weight  bmi_class_obese  bmi_class_overweight  \
0                          0.0              0.0                   1.0
1                          0.0              1.0                   0.0
2                          0.0              1.0                   0.0
3                          1.0              0.0                   0.0
4                          0.0              0.0                   1.0
…                          …                …                     …
1333                       0.0              1.0                   0.0
1334                       0.0              1.0                   0.0
1335                       0.0              1.0                   0.0
1336                       0.0              0.0                   1.0
1337                       0.0              0.0                   1.0

      bmi_class_underweight
0                       0.0
1                       0.0
2                       0.0
3                       0.0
4                       0.0
…                       …
1333                    0.0
1334                    0.0
1335                    0.0
1336                    0.0
1337                    0.0

[1337 rows x 14 columns]
```

4.5 Scaling Values

Values will be scaled by using MinMax Scaler.

[39]:
```
columns_to_scale = ['age','bmi','children']
input_df[columns_to_scale].head()
```

[39]:

	age	bmi	children
0	19	27.900	0
1	18	33.770	1
2	28	33.000	3
3	33	22.705	0

```
4   32  28.880         0
```

[40]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler().fit(input_df[columns_to_scale])

input_df[columns_to_scale] = scaler.transform(input_df[columns_to_scale])
```

[41]:
```
input_df = input_df.drop(columns='bmi_class')
input_df.head()
```

[41]:
```
        age  sex       bmi  children  smoker  region_northeast  \
0  0.021739    1  0.321227       0.0       1               0.0
1  0.000000    0  0.479150       0.2       0               0.0
2  0.217391    0  0.458434       0.6       0               0.0
3  0.326087    0  0.181464       0.0       0               0.0
4  0.304348    0  0.347592       0.0       0               0.0

   region_northwest  region_southeast  region_southwest  \
0               0.0               0.0               1.0
1               0.0               1.0               0.0
2               0.0               1.0               0.0
3               1.0               0.0               0.0
4               1.0               0.0               0.0

   bmi_class_healthy weight  bmi_class_obese  bmi_class_overweight  \
0                       0.0              0.0                   1.0
1                       0.0              1.0                   0.0
2                       0.0              1.0                   0.0
3                       1.0              0.0                   0.0
4                       0.0              0.0                   1.0

   bmi_class_underweight
0                    0.0
1                    0.0
2                    0.0
3                    0.0
4                    0.0
```

[42]:
```
input_df.describe().loc[['min','max']]
```

[42]:
```
     age  sex  bmi  children  smoker  region_northeast  region_northwest  \
min  0.0  0.0  0.0       0.0     0.0               0.0               0.0
max  1.0  1.0  1.0       1.0     1.0               1.0               1.0

     region_southeast  region_southwest  bmi_class_healthy weight  \
min               0.0               0.0                       0.0
```

```
max                 1.0                 1.0                          1.0

     bmi_class_obese  bmi_class_overweight  bmi_class_underweight
min              0.0                   0.0                    0.0
max              1.0                   1.0                    1.0
```

All the variables range now between 0 and 1.

4.6 Splitting the Input Data Frame into a Train and Validation One

[43]:
```python
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(input_df, target_df,␣
↪test_size=0.2, random_state=42)
```

Showing the result.

[44]:
```python
X_train
```

[44]:
```
           age  sex       bmi  children  smoker  region_northeast  \
1114  0.108696    0  0.230024       0.0       0               1.0
968   0.065217    0  0.263250       0.4       0               1.0
599   0.739130    1  0.580172       0.4       0               0.0
170   0.978261    0  0.686306       0.0       0               0.0
275   0.630435    1  0.286252       0.4       0               1.0
…          …   …        …         …       …                 …
1096  0.717391    1  0.511165       0.4       1               1.0
1131  0.195652    0  0.805488       0.4       0               0.0
1295  0.043478    0  0.162497       0.2       0               0.0
861   0.434783    1  0.323917       0.6       0               0.0
1127  0.369565    1  0.535378       0.4       0               0.0

      region_northwest  region_southeast  region_southwest  \
1114               0.0               0.0               0.0
968                0.0               0.0               0.0
599                1.0               0.0               0.0
170                0.0               1.0               0.0
275                0.0               0.0               0.0
…                    …                 …                 …
1096               0.0               0.0               0.0
1131               0.0               0.0               1.0
1295               0.0               0.0               1.0
861                0.0               0.0               1.0
1127               0.0               1.0               0.0

      bmi_class_healthy weight  bmi_class_obese  bmi_class_overweight  \
1114                       1.0              0.0                   0.0
968                        0.0              0.0                   1.0
```

```
599                          0.0              1.0                    0.0
170                          0.0              1.0                    0.0
275                          0.0              0.0                    1.0
…                           …               …                    …
1096                         0.0              1.0                    0.0
1131                         0.0              1.0                    0.0
1295                         1.0              0.0                    0.0
861                          0.0              0.0                    1.0
1127                         0.0              1.0                    0.0

      bmi_class_underweight
1114                    0.0
968                     0.0
599                     0.0
170                     0.0
275                     0.0
…                     …
1096                    0.0
1131                    0.0
1295                    0.0
861                     0.0
1127                    0.0

[1069 rows x 13 columns]
```

[45]:
```
X_val
```

[45]:
```
           age  sex       bmi  children  smoker  region_northeast  \
900   0.673913    0  0.176352       0.0       0               1.0
1064  0.239130    1  0.259349       0.8       0               0.0
1256  0.717391    1  0.549502       0.6       0               0.0
298   0.282609    0  0.495830       0.6       1               0.0
237   0.282609    0  0.603444       0.4       0               0.0
…         …  …       …       …     …               …
534   1.000000    0  0.659672       0.0       0               0.0
542   0.978261    1  0.547215       0.0       0               0.0
760   0.086957    1  0.500942       0.4       0               1.0
1284  0.934783    0  0.547215       0.2       1               0.0
1285  0.630435    1  0.224913       0.0       0               1.0

      region_northwest  region_southeast  region_southwest  \
900                0.0               0.0               0.0
1064               0.0               0.0               1.0
1256               1.0               0.0               0.0
298                1.0               0.0               0.0
237                0.0               1.0               0.0
…                …               …               …
```

```
534                0.0               1.0               0.0
542                0.0               1.0               0.0
760                0.0               0.0               0.0
1284               0.0               0.0               1.0
1285               0.0               0.0               0.0

      bmi_class_healthy weight  bmi_class_obese  bmi_class_overweight  \
900                        1.0              0.0                   0.0
1064                       0.0              0.0                   1.0
1256                       0.0              1.0                   0.0
298                        0.0              1.0                   0.0
237                        0.0              1.0                   0.0
…                        …              …                   …
534                        0.0              1.0                   0.0
542                        0.0              1.0                   0.0
760                        0.0              1.0                   0.0
1284                       0.0              1.0                   0.0
1285                       1.0              0.0                   0.0

      bmi_class_underweight
900                     0.0
1064                    0.0
1256                    0.0
298                     0.0
237                     0.0
…                     …
534                     0.0
542                     0.0
760                     0.0
1284                    0.0
1285                    0.0

[268 rows x 13 columns]
```

[46]:

```
y_train
```

[46]:

```
1114     2396.09590
968      3279.86855
599     33471.97189
170     13405.39030
275      9715.84100
            …
1096    44641.19740
1131     3693.42800
1295     1964.78000
861      7151.09200
1127     5836.52040
```

```
Name: charges, Length: 1069, dtype: float64
```

[47]:
```
y_val
```

[47]:
```
900      8688.85885
1064     5708.86700
1256    11436.73815
298     38746.35510
237      4463.20510
           …
534     13831.11520
542     13887.20400
760      3925.75820
1284    47403.88000
1285     8534.67180
Name: charges, Length: 268, dtype: float64
```

5 Creating the Model

It will be used the XGBRegressor model

[48]:
```
from xgboost import XGBRegressor
```

[49]:
```
model = XGBRegressor(n_jobs=-1, n_estimators=1000, early_stopping_rounds=50,␣
 ↪random_state=42)
model.fit(X_train, y_train, eval_set=[(X_train, y_train),(X_val,y_val)])
```

```
[0]     validation_0-rmse:12740.82085   validation_1-rmse:14354.43509
[1]     validation_0-rmse:9486.62138    validation_1-rmse:10616.10467
[2]     validation_0-rmse:7287.93828    validation_1-rmse:8240.67071
[3]     validation_0-rmse:5864.49266    validation_1-rmse:6655.85239
[4]     validation_0-rmse:4970.66799    validation_1-rmse:5662.25928
[5]     validation_0-rmse:4397.13963    validation_1-rmse:5106.75982
[6]     validation_0-rmse:4057.24959    validation_1-rmse:4771.82219
[7]     validation_0-rmse:3822.63958    validation_1-rmse:4610.10440
[8]     validation_0-rmse:3645.34812    validation_1-rmse:4541.40784
[9]     validation_0-rmse:3512.55671    validation_1-rmse:4492.49500
[10]    validation_0-rmse:3375.72474    validation_1-rmse:4488.07927
[11]    validation_0-rmse:3311.14597    validation_1-rmse:4474.10073
[12]    validation_0-rmse:3222.20924    validation_1-rmse:4471.14367
[13]    validation_0-rmse:3123.12721    validation_1-rmse:4490.20394
[14]    validation_0-rmse:3050.29756    validation_1-rmse:4498.37645
[15]    validation_0-rmse:3030.27200    validation_1-rmse:4494.12635
[16]    validation_0-rmse:2954.61214    validation_1-rmse:4496.12827
[17]    validation_0-rmse:2892.54924    validation_1-rmse:4521.35732
[18]    validation_0-rmse:2764.14245    validation_1-rmse:4546.57569
[19]    validation_0-rmse:2696.99813    validation_1-rmse:4542.33188
[20]    validation_0-rmse:2632.43704    validation_1-rmse:4530.55994
```

```
[21]	validation_0-rmse:2584.61584	validation_1-rmse:4535.77625
[22]	validation_0-rmse:2527.93047	validation_1-rmse:4543.95639
[23]	validation_0-rmse:2468.29461	validation_1-rmse:4568.30553
[24]	validation_0-rmse:2403.46871	validation_1-rmse:4583.96935
[25]	validation_0-rmse:2365.15049	validation_1-rmse:4597.30591
[26]	validation_0-rmse:2328.78542	validation_1-rmse:4612.73905
[27]	validation_0-rmse:2276.17594	validation_1-rmse:4621.37732
[28]	validation_0-rmse:2257.53377	validation_1-rmse:4630.06126
[29]	validation_0-rmse:2227.57430	validation_1-rmse:4632.32797
[30]	validation_0-rmse:2158.27133	validation_1-rmse:4633.59277
[31]	validation_0-rmse:2122.02279	validation_1-rmse:4643.16265
[32]	validation_0-rmse:2108.64767	validation_1-rmse:4642.75280
[33]	validation_0-rmse:2056.91867	validation_1-rmse:4671.44408
[34]	validation_0-rmse:2033.70646	validation_1-rmse:4684.71168
[35]	validation_0-rmse:2029.62547	validation_1-rmse:4684.88997
[36]	validation_0-rmse:1994.54510	validation_1-rmse:4693.26534
[37]	validation_0-rmse:1981.56759	validation_1-rmse:4691.73150
[38]	validation_0-rmse:1926.58067	validation_1-rmse:4695.43395
[39]	validation_0-rmse:1874.85411	validation_1-rmse:4717.92004
[40]	validation_0-rmse:1857.27879	validation_1-rmse:4718.50099
[41]	validation_0-rmse:1830.16744	validation_1-rmse:4709.73574
[42]	validation_0-rmse:1767.69701	validation_1-rmse:4724.83488
[43]	validation_0-rmse:1703.13915	validation_1-rmse:4778.27119
[44]	validation_0-rmse:1656.20379	validation_1-rmse:4788.64363
[45]	validation_0-rmse:1609.83477	validation_1-rmse:4777.29533
[46]	validation_0-rmse:1572.18702	validation_1-rmse:4781.85206
[47]	validation_0-rmse:1567.22527	validation_1-rmse:4783.89930
[48]	validation_0-rmse:1539.31888	validation_1-rmse:4808.19374
[49]	validation_0-rmse:1532.71819	validation_1-rmse:4809.19389
[50]	validation_0-rmse:1509.51840	validation_1-rmse:4819.17982
[51]	validation_0-rmse:1470.37402	validation_1-rmse:4818.69505
[52]	validation_0-rmse:1448.02958	validation_1-rmse:4845.66365
[53]	validation_0-rmse:1442.19708	validation_1-rmse:4843.40563
[54]	validation_0-rmse:1393.94653	validation_1-rmse:4846.84045
[55]	validation_0-rmse:1372.76705	validation_1-rmse:4841.93288
[56]	validation_0-rmse:1338.61655	validation_1-rmse:4873.32958
[57]	validation_0-rmse:1299.63161	validation_1-rmse:4886.52445
[58]	validation_0-rmse:1292.29898	validation_1-rmse:4888.67500
[59]	validation_0-rmse:1282.16949	validation_1-rmse:4890.51711
[60]	validation_0-rmse:1273.10161	validation_1-rmse:4890.99547
[61]	validation_0-rmse:1256.99184	validation_1-rmse:4891.56390
```

```
[49]: XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
                   colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
                   early_stopping_rounds=50, enable_categorical=False,
                   eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
                   importance_type=None, interaction_constraints='',
```

```
learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
missing=nan, monotone_constraints='()', n_estimators=1000,
n_jobs=-1, num_parallel_tree=1, predictor='auto', random_state=42,
reg_alpha=0, reg_lambda=1, …)
```

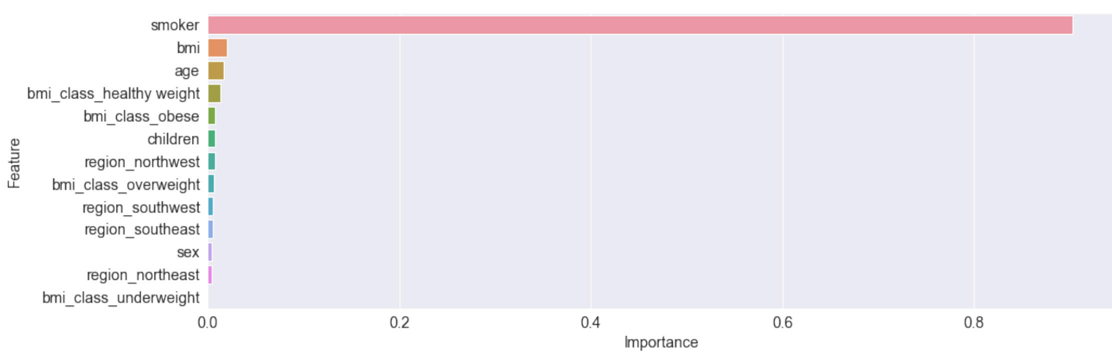
6 Checking the Importance of Each Feature

[50]:
```python
feature_importance_df = pd.DataFrame({
    'Feature':input_df.columns,
    'Importance':model.feature_importances_
})

feature_importance_df.sort_values('Importance', ascending=False)
```

[50]:

	Feature	Importance
4	smoker	0.903173
2	bmi	0.020471
0	age	0.016071
9	bmi_class_healthy weight	0.012697
10	bmi_class_obese	0.007735
3	children	0.007344
6	region_northwest	0.007163
11	bmi_class_overweight	0.006668
8	region_southwest	0.005663
7	region_southeast	0.005130
1	sex	0.004088
5	region_northeast	0.003797
12	bmi_class_underweight	0.000000

[51]:
```python
sns.barplot(data=feature_importance_df.sort_values('Importance',␣
↪ascending=False), y='Feature', x='Importance');
```



The smoker feature appears to be the most important one in determining the charges.

To check the best number of trees.

[52]:
```
model.best_ntree_limit
```

[52]:
```
13
```

13 is the best number of trees use in the model.

Checking the R^2 score for train and validation sets.

[53]:
```
from sklearn.metrics import r2_score

print('The R^2 Score for the Training Set is: {}'.format(r2_score(y_train,␣
 ↪model.predict(X_train))))
print('The R^2 Score for the Validation Set is: {}'.format(r2_score(y_val,␣
 ↪model.predict(X_val))))
```

```
The R^2 Score for the Training Set is: 0.92416713065968
The R^2 Score for the Validation Set is: 0.8912083506265275
```

Since it ranges from 0 to 1, where 0 is the minimum and 1 the maximum, the model seems to work properly.

Moreover, another way to check the goodness of a model is by performing NRMSE = RMSE/(y__max-y__min) –> https://www.statology.org/what-is-a-good-rmse/

The closer to 0 the better.

[54]:
```
from sklearn.metrics import mean_squared_error

rmse = mean_squared_error(y_val, model.predict(X_val), squared=False)

nrmse = rmse/(max(target_df)-min(target_df))

print('The RMSE is: {}'.format(rmse))
print('The NRMSE is: {}'.format(nrmse))
```

```
The RMSE is: 4471.143622095877
The NRMSE is: 0.07136866421921444
```

Now it is better to check some examples of the difference between the predicted and the actual values.

[55]:
```
y_pred = model.predict(X_val)
y_pred[:10]
```

[55]:
```
array([ 9457.824 ,  5656.2124, 13328.179 , 37901.95  ,  5033.9526,
        9560.842 , 37780.    ,  2835.1318,  8372.604 , 10332.222 ],
      dtype=float32)
```

[56]:
```
y_val[:10]
```

[56]:
```
900      8688.85885
1064     5708.86700
1256    11436.73815
298     38746.35510
237      4463.20510
481      9304.70190
240     38511.62830
277      2150.46900
415      7345.72660
707     10264.44210
Name: charges, dtype: float64
```

The predictions, except certain cases, do not seem to be so far from reality.

7 Hyperparameter Tuning

In this section are provided some techniques to improve the score of the model.

The hyperparameters can be found here: https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgb

7.1 Max Depth

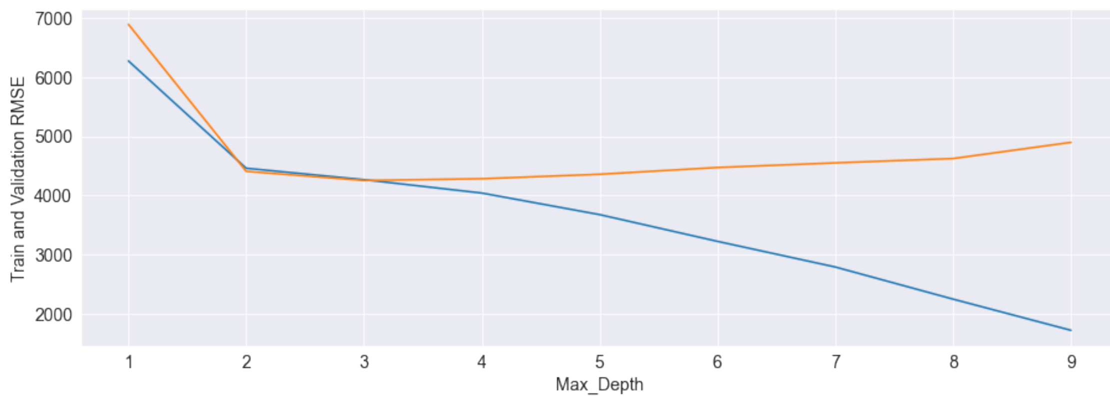
[57]:
```
def optimal_max_depth(number):
    max_depth_model = XGBRegressor(n_jobs=-1, n_estimators=13, max_depth=number␣
 ↪,random_state=42)
    max_depth_model.fit(X_train, y_train)
    train_rmse = mean_squared_error(y_train, max_depth_model.predict(X_train),␣
 ↪squared=False)
    val_rmse = mean_squared_error(y_val, max_depth_model.predict(X_val),␣
 ↪squared=False)
    return {'Max_Depth':number, 'Train_Rmse':train_rmse, 'Validation_Rmse':
 ↪val_rmse}
```

[58]:
```
max_depth_df = pd.DataFrame([optimal_max_depth(number) for number in␣
 ↪range(1,10)]).sort_values('Validation_Rmse')
max_depth_df.head()
```

[58]:

	Max_Depth	Train_Rmse	Validation_Rmse
2	3	4264.770476	4251.996995
3	4	4038.957467	4280.987639
4	5	3674.277444	4356.719852
1	2	4458.462012	4405.776079
5	6	3222.209225	4471.143622

[59]:
```
sns.lineplot(data=max_depth_df, x='Max_Depth', y='Train_Rmse')
sns.lineplot(data=max_depth_df, x='Max_Depth', y='Validation_Rmse')
plt.ylabel('Train and Validation RMSE');
```



At "max_depth"=3 the validation rmse starts increasing.

7.2 Learning Rate

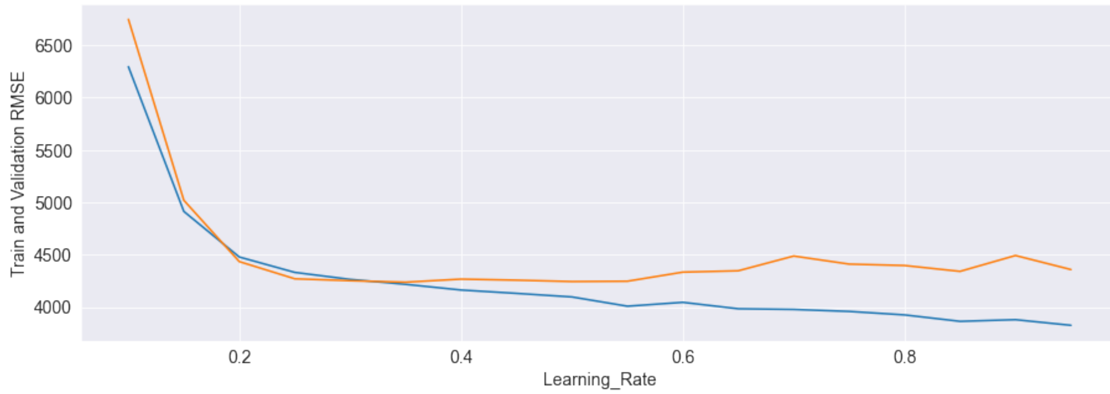
[60]:
```
def optimal_learning_rate(number):
    model = XGBRegressor(n_jobs=-1, n_estimators=13, max_depth=3,␣
 ↪learning_rate=number, random_state=42)
    model.fit(X_train, y_train)
    train_rmse = mean_squared_error(y_train, model.predict(X_train),␣
 ↪squared=False)
    val_rmse = mean_squared_error(y_val, model.predict(X_val), squared=False)
    return {'Learning_Rate':number, 'Train_Rmse':train_rmse, 'Validation_Rmse':
 ↪val_rmse}
```

[61]:
```
learning_rate_df = pd.DataFrame([optimal_learning_rate(number) for number in np.
 ↪arange(0.1,1,0.05)]).sort_values('Validation_Rmse')
learning_rate_df.head()
```

[61]:

	Learning_Rate	Train_Rmse	Validation_Rmse
5	0.35	4217.498960	4238.039332
8	0.50	4097.225842	4244.259593
9	0.55	4008.885655	4247.309357
4	0.30	4264.770476	4251.996995
7	0.45	4131.279452	4257.320246

[62]:
```
sns.lineplot(data=learning_rate_df, x='Learning_Rate', y='Train_Rmse')
sns.lineplot(data=learning_rate_df, x='Learning_Rate', y='Validation_Rmse')
plt.ylabel('Train and Validation RMSE');
```



0.35 is the best value.

7.3 Min Child Weight

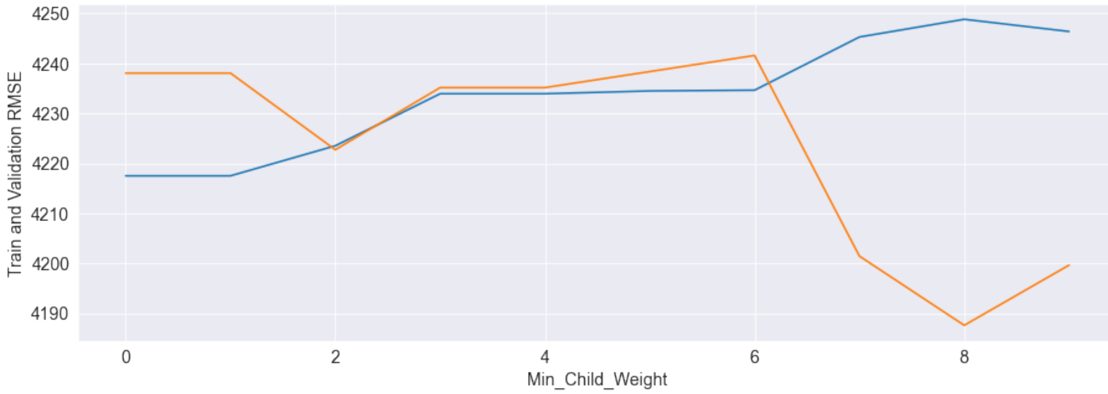
[63]:
```
def optimal_child_weight(number):
    model = XGBRegressor(n_jobs=-1, n_estimators=13, max_depth=3,␣
↪learning_rate=0.35, min_child_weight=number ,random_state=42)
    model.fit(X_train, y_train)
    train_rmse = mean_squared_error(y_train, model.predict(X_train),␣
↪squared=False)
    val_rmse = mean_squared_error(y_val, model.predict(X_val), squared=False)
    return {'Min_Child_Weight':number, 'Train_Rmse':train_rmse,␣
↪'Validation_Rmse':val_rmse}
```

[64]:
```
child_weight_df = pd.DataFrame([optimal_child_weight(number) for number in␣
↪range(0,10)]).sort_values('Validation_Rmse')
child_weight_df.head()
```

[64]:

	Min_Child_Weight	Train_Rmse	Validation_Rmse
8	8	4248.802687	4187.605787
9	9	4246.365003	4199.642400
7	7	4245.259970	4201.440936
2	2	4223.506507	4222.713290
3	3	4233.919525	4235.137942

[65]:
```
sns.lineplot(data=child_weight_df, x='Min_Child_Weight', y='Train_Rmse')
sns.lineplot(data=child_weight_df, x='Min_Child_Weight', y='Validation_Rmse')
plt.ylabel('Train and Validation RMSE');
```



At level 8 the rmse decreases a lot.

8 Creating a New Model

[66]:
```
final_model = XGBRegressor(n_jobs=-1, n_estimators=13, max_depth=3,␣
    ↪learning_rate=0.35, min_child_weight=8, random_state=42)
final_model.fit(X_train, y_train)
```

[66]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.35, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=3, max_leaves=0, min_child_weight=8,
             missing=nan, monotone_constraints='()', n_estimators=13, n_jobs=-1,
             num_parallel_tree=1, predictor='auto', random_state=42,
             reg_alpha=0, reg_lambda=1, …)
```

Checking new scores.

[67]:
```
print('The R^2 Score for the Training Set is: {}'.format(r2_score(y_train,␣
    ↪final_model.predict(X_train))))
print('The R^2 Score for the Validation Set is: {}'.format(r2_score(y_val,␣
    ↪final_model.predict(X_val))))
```

```
The R^2 Score for the Training Set is: 0.8681490868783452
The R^2 Score for the Validation Set is: 0.9045689059547112
```

The R^2 score of the validation set has improved.

[68]:
```
rmse = mean_squared_error(y_val, final_model.predict(X_val), squared=False)

nrmse = rmse/(max(target_df)-min(target_df))
```

```
print('The RMSE is: {}'.format(rmse))
print('The NRMSE is: {}'.format(nrmse))
```

```
The RMSE is: 4187.605787188936
The NRMSE is: 0.0668428161939097
```

The rmse and the nrmse have decreased, the hyperparameter tuning was completed successfully.

9 Making Predictions on New Inputs

[80]:
```
def make_new_predictions(single_input):
    single_input_df = pd.DataFrame([single_input])
    single_input_df['bmi_class']=single_input_df.bmi.map(bmi_estimator)
    single_input_df['sex'] = single_input_df.sex.map(sex_dict)
    single_input_df['smoker'] = single_input_df.smoker.map(smoker_dict)
    single_input_df[encoded_cols] = encoder.
 ↪transform(single_input_df[columns_to_encode])
    single_input_df[columns_to_scale] = scaler.
 ↪transform(single_input_df[columns_to_scale])
    single_input_df = single_input_df.drop(columns=['region', 'bmi_class'])
    pred = final_model.predict(single_input_df)[0]
    return 'The charge is: ${}'.format(pred)
```

[70]:
```
new_input = {
    'age':23,
    'sex':'male',
    'bmi':28.1,
    'children':0,
    'smoker':'no',
    'region':'northwest'
}
```

[81]:
```
make_new_predictions(new_input)
```

[81]:
```
'The charge is: $4507.5537109375'
```

10 Extra: Using an Artificial Neural Network

[ ]:
```
import tensorflow as tfl
from tensorflow import keras
```

[247]:
```
ann = keras.Sequential([
    keras.layers.Dense(10, input_shape=(13,), activation='relu'),
    keras.layers.Dense(3, activation='relu'),
```

```
    keras.layers.Dense(1, activation='linear')
])

ann.compile(
    optimizer='sgd',
    loss='mse',
    metrics=[tfl.keras.metrics.RootMeanSquaredError()]
)

ann.fit(X_train, y_train, epochs=100)
```

```
Epoch 1/100
34/34 [==============================] - 0s 2ms/step - loss: 1023015744.0000 -
root_mean_squared_error: 31984.6172
Epoch 2/100
34/34 [==============================] - 0s 2ms/step - loss: 176589888.0000 -
root_mean_squared_error: 13288.7129
Epoch 3/100
34/34 [==============================] - 0s 2ms/step - loss: 147162272.0000 -
root_mean_squared_error: 12131.0459
Epoch 4/100
34/34 [==============================] - 0s 2ms/step - loss: 139552272.0000 -
root_mean_squared_error: 11813.2246
Epoch 5/100
34/34 [==============================] - 0s 2ms/step - loss: 137603088.0000 -
root_mean_squared_error: 11730.4346
Epoch 6/100
34/34 [==============================] - 0s 2ms/step - loss: 137115984.0000 -
root_mean_squared_error: 11709.6533
Epoch 7/100
34/34 [==============================] - 0s 2ms/step - loss: 136999296.0000 -
root_mean_squared_error: 11704.6699
Epoch 8/100
34/34 [==============================] - 0s 2ms/step - loss: 136994272.0000 -
root_mean_squared_error: 11704.4551
Epoch 9/100
34/34 [==============================] - 0s 2ms/step - loss: 136996112.0000 -
root_mean_squared_error: 11704.5342
Epoch 10/100
34/34 [==============================] - 0s 2ms/step - loss: 136972224.0000 -
root_mean_squared_error: 11703.5137
Epoch 11/100
34/34 [==============================] - 0s 2ms/step - loss: 137018096.0000 -
root_mean_squared_error: 11705.4727
Epoch 12/100
34/34 [==============================] - 0s 2ms/step - loss: 136983792.0000 -
root_mean_squared_error: 11704.0078
Epoch 13/100
```

```
34/34 [==============================] - 0s 2ms/step - loss: 136982960.0000 -
root_mean_squared_error: 11703.9717
Epoch 14/100
34/34 [==============================] - 0s 2ms/step - loss: 137011376.0000 -
root_mean_squared_error: 11705.1855
Epoch 15/100
34/34 [==============================] - 0s 2ms/step - loss: 137011952.0000 -
root_mean_squared_error: 11705.2109
Epoch 16/100
34/34 [==============================] - 0s 2ms/step - loss: 136983872.0000 -
root_mean_squared_error: 11704.0107
Epoch 17/100
34/34 [==============================] - 0s 2ms/step - loss: 136980240.0000 -
root_mean_squared_error: 11703.8555
Epoch 18/100
34/34 [==============================] - 0s 2ms/step - loss: 136984624.0000 -
root_mean_squared_error: 11704.0430
Epoch 19/100
34/34 [==============================] - 0s 2ms/step - loss: 137001376.0000 -
root_mean_squared_error: 11704.7588
Epoch 20/100
34/34 [==============================] - 0s 2ms/step - loss: 137019328.0000 -
root_mean_squared_error: 11705.5254
Epoch 21/100
34/34 [==============================] - 0s 2ms/step - loss: 136977648.0000 -
root_mean_squared_error: 11703.7451
Epoch 22/100
34/34 [==============================] - 0s 2ms/step - loss: 137017584.0000 -
root_mean_squared_error: 11705.4512
Epoch 23/100
34/34 [==============================] - 0s 2ms/step - loss: 137005824.0000 -
root_mean_squared_error: 11704.9482
Epoch 24/100
34/34 [==============================] - 0s 2ms/step - loss: 137007344.0000 -
root_mean_squared_error: 11705.0137
Epoch 25/100
34/34 [==============================] - 0s 2ms/step - loss: 136976592.0000 -
root_mean_squared_error: 11703.7002
Epoch 26/100
34/34 [==============================] - 0s 2ms/step - loss: 136965056.0000 -
root_mean_squared_error: 11703.2070
Epoch 27/100
34/34 [==============================] - 0s 2ms/step - loss: 137021824.0000 -
root_mean_squared_error: 11705.6318
Epoch 28/100
34/34 [==============================] - 0s 2ms/step - loss: 136975072.0000 -
root_mean_squared_error: 11703.6348
Epoch 29/100
```

```
34/34 [==============================] - 0s 2ms/step - loss: 137020176.0000 -
root_mean_squared_error: 11705.5615
Epoch 30/100
34/34 [==============================] - 0s 2ms/step - loss: 137025360.0000 -
root_mean_squared_error: 11705.7832
Epoch 31/100
34/34 [==============================] - 0s 2ms/step - loss: 137002784.0000 -
root_mean_squared_error: 11704.8184
Epoch 32/100
34/34 [==============================] - 0s 2ms/step - loss: 136961808.0000 -
root_mean_squared_error: 11703.0684
Epoch 33/100
34/34 [==============================] - 0s 2ms/step - loss: 136978816.0000 -
root_mean_squared_error: 11703.7949
Epoch 34/100
34/34 [==============================] - 0s 1ms/step - loss: 136985920.0000 -
root_mean_squared_error: 11704.0986
Epoch 35/100
34/34 [==============================] - 0s 2ms/step - loss: 136969104.0000 -
root_mean_squared_error: 11703.3799
Epoch 36/100
34/34 [==============================] - 0s 2ms/step - loss: 136965648.0000 -
root_mean_squared_error: 11703.2324
Epoch 37/100
34/34 [==============================] - 0s 2ms/step - loss: 136972256.0000 -
root_mean_squared_error: 11703.5146
Epoch 38/100
34/34 [==============================] - 0s 2ms/step - loss: 136984384.0000 -
root_mean_squared_error: 11704.0332
Epoch 39/100
34/34 [==============================] - 0s 2ms/step - loss: 136975968.0000 -
root_mean_squared_error: 11703.6729
Epoch 40/100
34/34 [==============================] - 0s 2ms/step - loss: 136966528.0000 -
root_mean_squared_error: 11703.2695
Epoch 41/100
34/34 [==============================] - 0s 2ms/step - loss: 136998064.0000 -
root_mean_squared_error: 11704.6172
Epoch 42/100
34/34 [==============================] - 0s 2ms/step - loss: 137011952.0000 -
root_mean_squared_error: 11705.2109
Epoch 43/100
34/34 [==============================] - 0s 2ms/step - loss: 136979376.0000 -
root_mean_squared_error: 11703.8193
Epoch 44/100
34/34 [==============================] - 0s 2ms/step - loss: 136968944.0000 -
root_mean_squared_error: 11703.3730
Epoch 45/100
```

```
34/34 [==============================] - 0s 2ms/step - loss: 137008480.0000 -
root_mean_squared_error: 11705.0625
Epoch 46/100
34/34 [==============================] - 0s 2ms/step - loss: 136984288.0000 -
root_mean_squared_error: 11704.0283
Epoch 47/100
34/34 [==============================] - 0s 2ms/step - loss: 137016272.0000 -
root_mean_squared_error: 11705.3945
Epoch 48/100
34/34 [==============================] - 0s 1ms/step - loss: 136949136.0000 -
root_mean_squared_error: 11702.5273
Epoch 49/100
34/34 [==============================] - 0s 1ms/step - loss: 136987040.0000 -
root_mean_squared_error: 11704.1465
Epoch 50/100
34/34 [==============================] - 0s 1ms/step - loss: 137002672.0000 -
root_mean_squared_error: 11704.8145
Epoch 51/100
34/34 [==============================] - 0s 2ms/step - loss: 137007120.0000 -
root_mean_squared_error: 11705.0039
Epoch 52/100
34/34 [==============================] - 0s 1ms/step - loss: 137017168.0000 -
root_mean_squared_error: 11705.4336
Epoch 53/100
34/34 [==============================] - 0s 2ms/step - loss: 136956448.0000 -
root_mean_squared_error: 11702.8389
Epoch 54/100
34/34 [==============================] - 0s 2ms/step - loss: 137008176.0000 -
root_mean_squared_error: 11705.0488
Epoch 55/100
34/34 [==============================] - 0s 1ms/step - loss: 136995856.0000 -
root_mean_squared_error: 11704.5225
Epoch 56/100
34/34 [==============================] - 0s 2ms/step - loss: 137006960.0000 -
root_mean_squared_error: 11704.9971
Epoch 57/100
34/34 [==============================] - 0s 1ms/step - loss: 136971888.0000 -
root_mean_squared_error: 11703.4990
Epoch 58/100
34/34 [==============================] - 0s 2ms/step - loss: 137011424.0000 -
root_mean_squared_error: 11705.1875
Epoch 59/100
34/34 [==============================] - 0s 2ms/step - loss: 136971648.0000 -
root_mean_squared_error: 11703.4883
Epoch 60/100
34/34 [==============================] - 0s 1ms/step - loss: 136953808.0000 -
root_mean_squared_error: 11702.7266
Epoch 61/100
```

```
34/34 [==============================] - 0s 1ms/step - loss: 137024688.0000 -
root_mean_squared_error: 11705.7549
Epoch 62/100
34/34 [==============================] - 0s 2ms/step - loss: 137001072.0000 -
root_mean_squared_error: 11704.7461
Epoch 63/100
34/34 [==============================] - 0s 2ms/step - loss: 137007056.0000 -
root_mean_squared_error: 11705.0010
Epoch 64/100
34/34 [==============================] - 0s 2ms/step - loss: 136997936.0000 -
root_mean_squared_error: 11704.6113
Epoch 65/100
34/34 [==============================] - 0s 1ms/step - loss: 136983520.0000 -
root_mean_squared_error: 11703.9961
Epoch 66/100
34/34 [==============================] - 0s 1ms/step - loss: 136999584.0000 -
root_mean_squared_error: 11704.6826
Epoch 67/100
34/34 [==============================] - 0s 2ms/step - loss: 136985328.0000 -
root_mean_squared_error: 11704.0732
Epoch 68/100
34/34 [==============================] - 0s 2ms/step - loss: 136983984.0000 -
root_mean_squared_error: 11704.0156
Epoch 69/100
34/34 [==============================] - 0s 1ms/step - loss: 137004432.0000 -
root_mean_squared_error: 11704.8896
Epoch 70/100
34/34 [==============================] - 0s 1ms/step - loss: 136978064.0000 -
root_mean_squared_error: 11703.7627
Epoch 71/100
34/34 [==============================] - 0s 1ms/step - loss: 136978176.0000 -
root_mean_squared_error: 11703.7676
Epoch 72/100
34/34 [==============================] - 0s 1ms/step - loss: 136978096.0000 -
root_mean_squared_error: 11703.7646
Epoch 73/100
34/34 [==============================] - 0s 1ms/step - loss: 136965952.0000 -
root_mean_squared_error: 11703.2451
Epoch 74/100
34/34 [==============================] - 0s 1ms/step - loss: 137002048.0000 -
root_mean_squared_error: 11704.7871
Epoch 75/100
34/34 [==============================] - 0s 2ms/step - loss: 136999568.0000 -
root_mean_squared_error: 11704.6816
Epoch 76/100
34/34 [==============================] - 0s 1ms/step - loss: 137017312.0000 -
root_mean_squared_error: 11705.4395
Epoch 77/100
```

```
34/34 [==============================] - 0s 1ms/step - loss: 136952224.0000 -
root_mean_squared_error: 11702.6592
Epoch 78/100
34/34 [==============================] - 0s 2ms/step - loss: 136991488.0000 -
root_mean_squared_error: 11704.3359
Epoch 79/100
34/34 [==============================] - 0s 2ms/step - loss: 137023936.0000 -
root_mean_squared_error: 11705.7227
Epoch 80/100
34/34 [==============================] - 0s 2ms/step - loss: 136962896.0000 -
root_mean_squared_error: 11703.1152
Epoch 81/100
34/34 [==============================] - 0s 1ms/step - loss: 136982448.0000 -
root_mean_squared_error: 11703.9502
Epoch 82/100
34/34 [==============================] - 0s 2ms/step - loss: 136994624.0000 -
root_mean_squared_error: 11704.4707
Epoch 83/100
34/34 [==============================] - 0s 1ms/step - loss: 137003904.0000 -
root_mean_squared_error: 11704.8662
Epoch 84/100
34/34 [==============================] - 0s 1ms/step - loss: 136984304.0000 -
root_mean_squared_error: 11704.0293
Epoch 85/100
34/34 [==============================] - 0s 2ms/step - loss: 136977696.0000 -
root_mean_squared_error: 11703.7471
Epoch 86/100
34/34 [==============================] - 0s 2ms/step - loss: 136982464.0000 -
root_mean_squared_error: 11703.9512
Epoch 87/100
34/34 [==============================] - 0s 2ms/step - loss: 137010544.0000 -
root_mean_squared_error: 11705.1504
Epoch 88/100
34/34 [==============================] - 0s 2ms/step - loss: 136988256.0000 -
root_mean_squared_error: 11704.1982
Epoch 89/100
34/34 [==============================] - 0s 2ms/step - loss: 137008288.0000 -
root_mean_squared_error: 11705.0537
Epoch 90/100
34/34 [==============================] - 0s 1ms/step - loss: 136993968.0000 -
root_mean_squared_error: 11704.4424
Epoch 91/100
34/34 [==============================] - 0s 2ms/step - loss: 136968336.0000 -
root_mean_squared_error: 11703.3477
Epoch 92/100
34/34 [==============================] - 0s 2ms/step - loss: 136985888.0000 -
root_mean_squared_error: 11704.0967
Epoch 93/100
```

```
34/34 [==============================] - 0s 1ms/step - loss: 137019552.0000 -
root_mean_squared_error: 11705.5352
Epoch 94/100
34/34 [==============================] - 0s 2ms/step - loss: 137007712.0000 -
root_mean_squared_error: 11705.0293
Epoch 95/100
34/34 [==============================] - 0s 2ms/step - loss: 137005808.0000 -
root_mean_squared_error: 11704.9482
Epoch 96/100
34/34 [==============================] - 0s 2ms/step - loss: 137000880.0000 -
root_mean_squared_error: 11704.7373
Epoch 97/100
34/34 [==============================] - 0s 2ms/step - loss: 137021120.0000 -
root_mean_squared_error: 11705.6025
Epoch 98/100
34/34 [==============================] - 0s 2ms/step - loss: 136983904.0000 -
root_mean_squared_error: 11704.0127
Epoch 99/100
34/34 [==============================] - 0s 2ms/step - loss: 136972720.0000 -
root_mean_squared_error: 11703.5342
Epoch 100/100
34/34 [==============================] - 0s 2ms/step - loss: 136958528.0000 -
root_mean_squared_error: 11702.9277
```

```
[247]: <keras.callbacks.History at 0x181203b57b0>
```

It does not seem to perform so well in this case, for sure it needs further implementation.