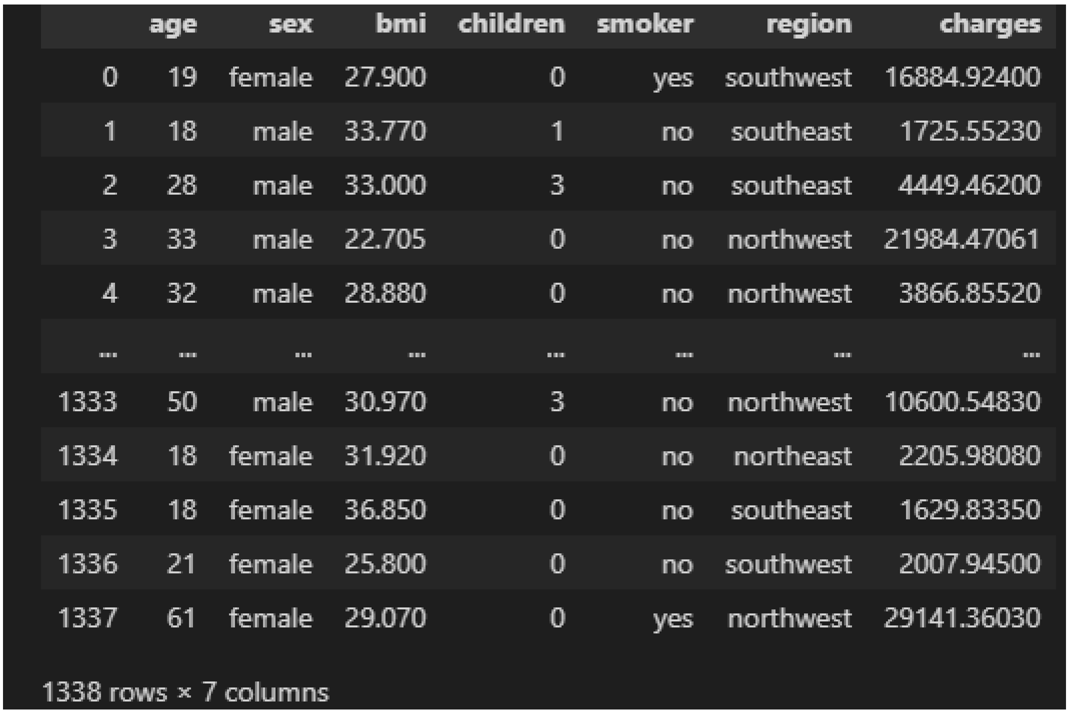
Prediction of Insurance Cost

Dataset Desbriction

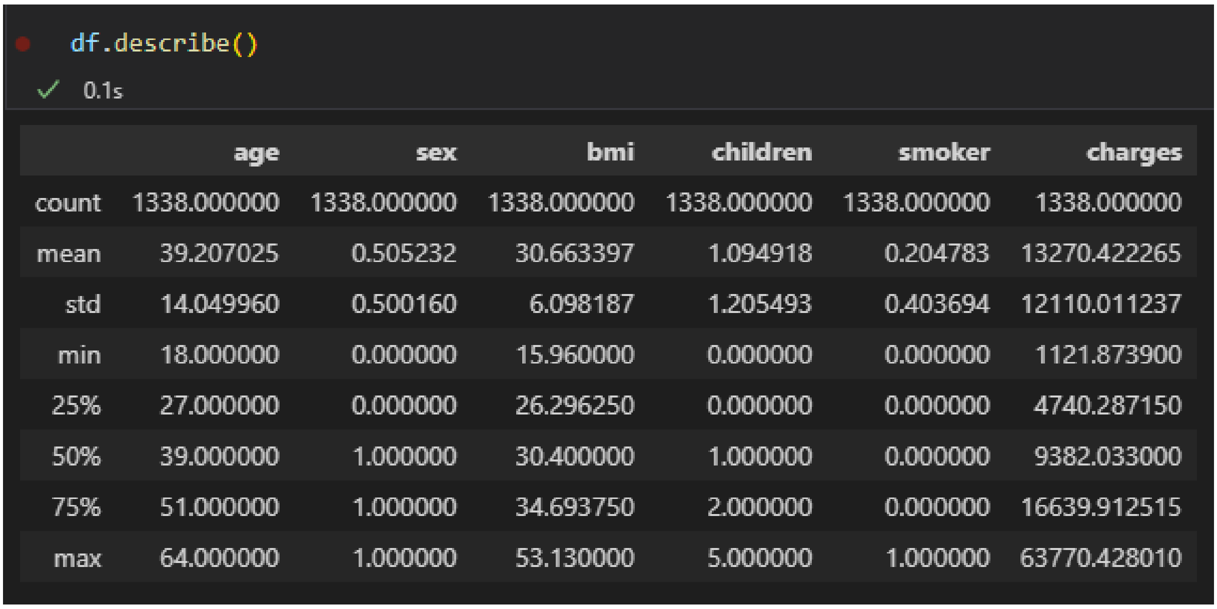
The problem about individual medical expenses for health insurance and to observe the effect of "age", "sex", "bmi", "children", "smoker", "region" on the invoice. The target of our problem is “charges”.



	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 × 7 columns

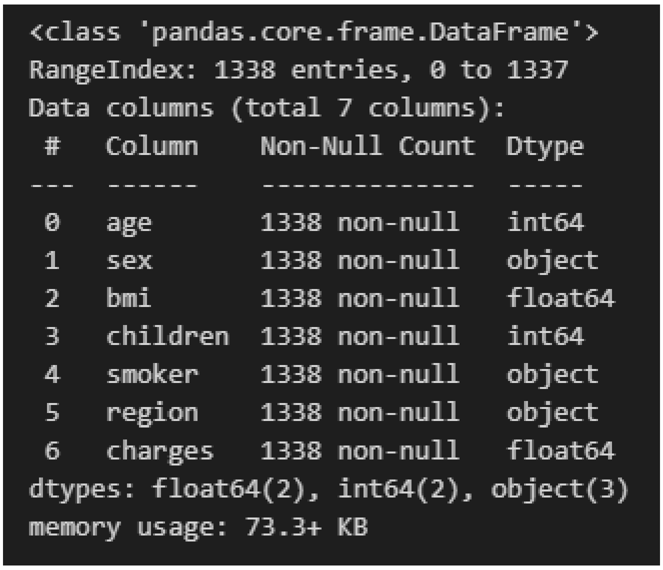
The dataset have 7 columns and 1338 rows.



```
df.describe()
```

✓ 0.1s

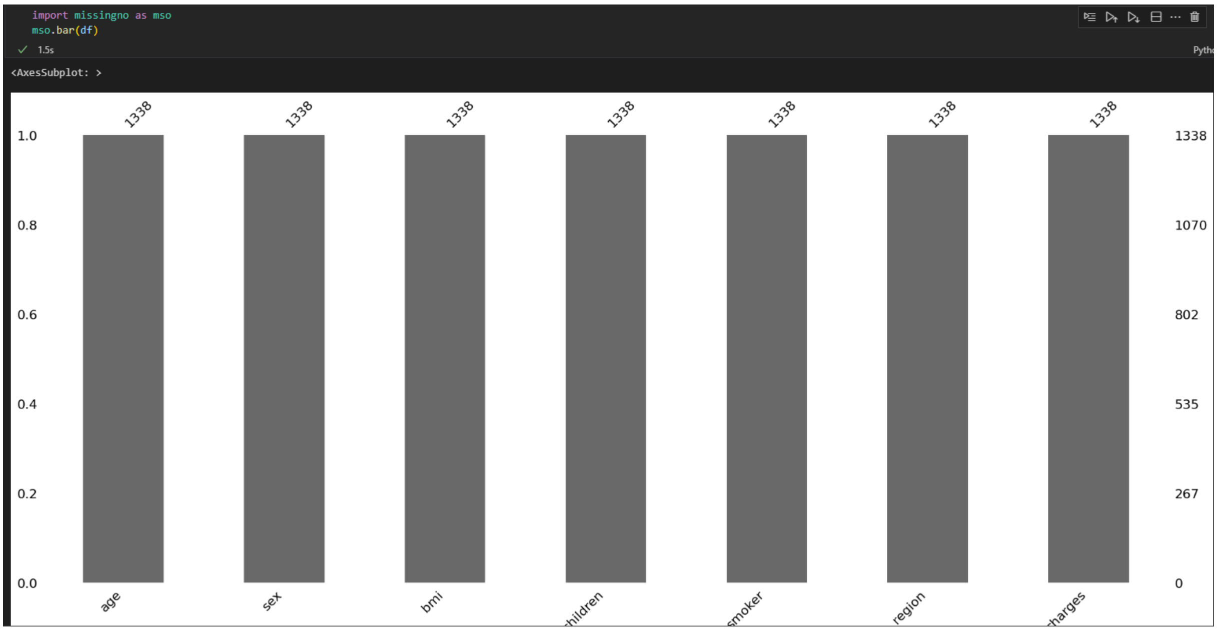
	age	sex	bmi	children	smoker	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.204783	13270.422265
std	14.049960	0.500160	6.098187	1.205493	0.403694	12110.011237
min	18.000000	0.000000	15.960000	0.000000	0.000000	1121.873900
25%	27.000000	0.000000	26.296250	0.000000	0.000000	4740.287150
50%	39.000000	1.000000	30.400000	1.000000	0.000000	9382.033000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	16639.912515
max	64.000000	1.000000	53.130000	5.000000	1.000000	63770.428010



```
<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
```

“Age” and “children” columns have integer data type; “bmi” and “charges” columns have float data type; “sex”, “smoker” and “region” have object data type.

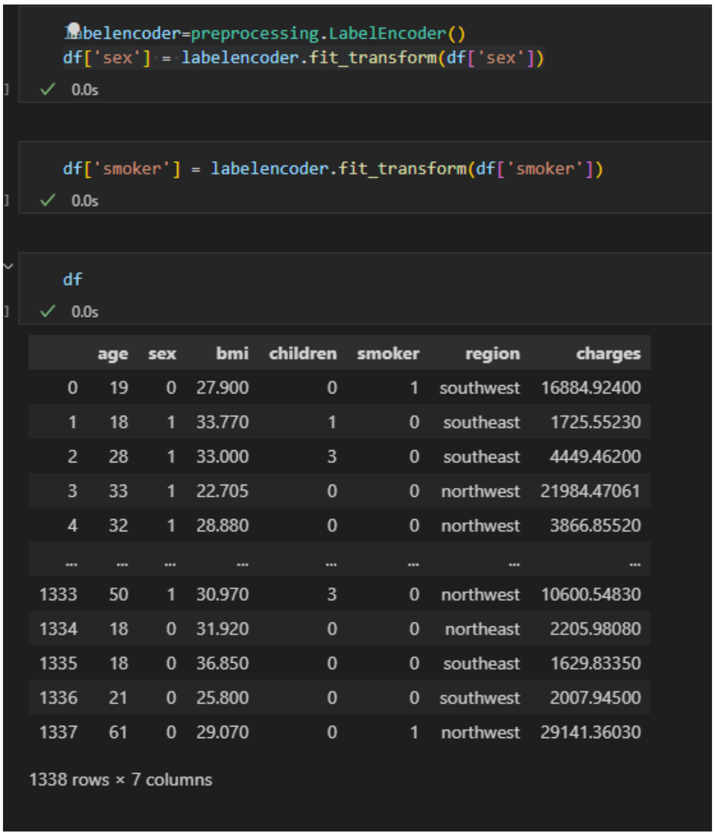
But we will change datatypes of “sex” and “smoker” columns to 1-0 integer type. Lets check the dataset if it has any missing data.



```
import missingno as mso
mso.bar(df)
```

```
<AxesSubplot: >
```

As we can see, there is no missing data.



```
labelencoder=preprocessing.LabelEncoder()
df['sex'] = labelencoder.fit_transform(df['sex'])
```

✓ 0.0s

```
df['smoker'] = labelencoder.fit_transform(df['smoker'])
```

✓ 0.0s

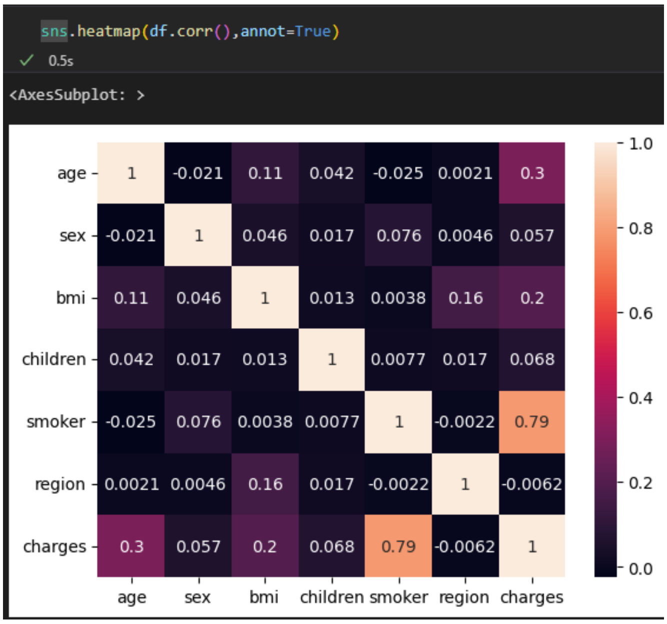
```
df
```

✓ 0.0s

	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
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	0	northwest	10600.54830
1334	18	0	31.920	0	0	northeast	2205.98080
1335	18	0	36.850	0	0	southeast	1629.83350
1336	21	0	25.800	0	0	southwest	2007.94500
1337	61	0	29.070	0	1	northwest	29141.36030

1338 rows × 7 columns

We did the encoding to change datatypes of “sex” and “smoker”. “1” in the “sex” column means “male” and “0” means “female”. “1” in the “smoker column means “yes” and “0” means is “no”. Now, lets see the correlation of dataset.



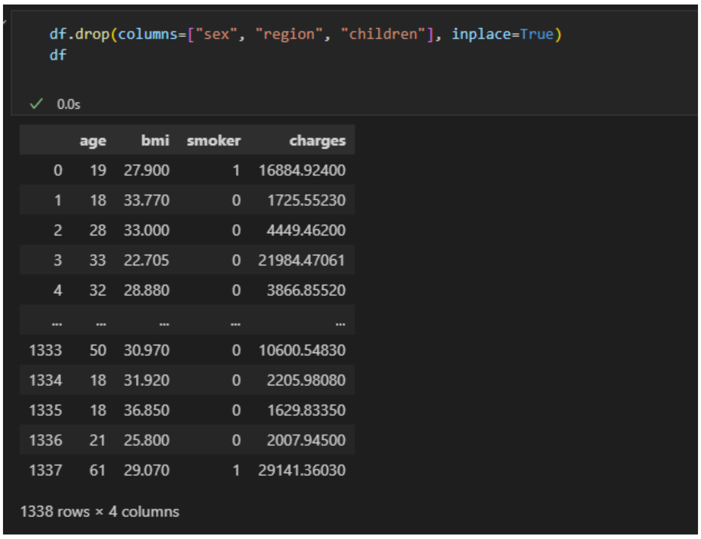
```
sns.heatmap(df.corr(),annot=True)
```

✓ 0.5s

```
<AxesSubplot: >
```

	age	sex	bmi	children	smoker	region	charges
age	1	-0.021	0.11	0.042	-0.025	0.0021	0.3
sex	-0.021	1	0.046	0.017	0.076	0.0046	0.057
bmi	0.11	0.046	1	0.013	0.0038	0.16	0.2
children	0.042	0.017	0.013	1	0.0077	0.017	0.068
smoker	-0.025	0.076	0.0038	0.0077	1	-0.0022	0.79
region	0.0021	0.0046	0.16	0.017	-0.0022	1	-0.0062
charges	0.3	0.057	0.2	0.068	0.79	-0.0062	1

“sex”, “region” and “children” are has really low correlation with chatges. So, we can drop these columns before train the model.



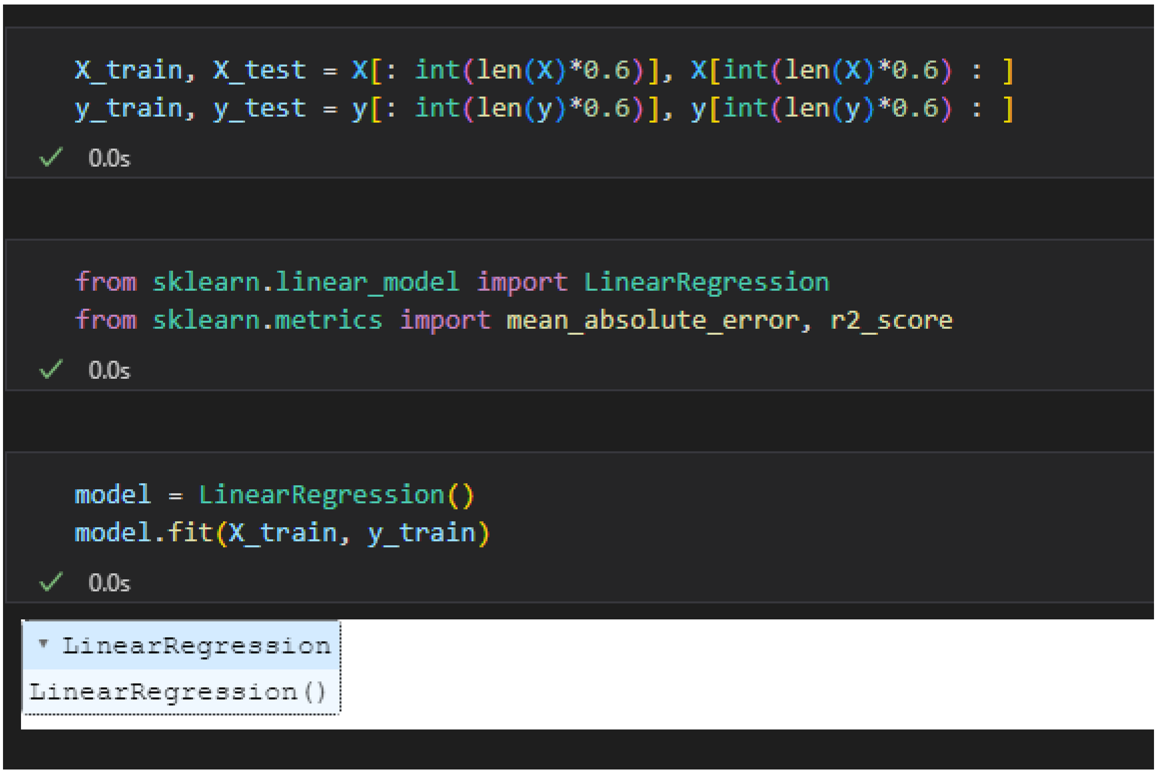
```
df.drop(columns=["sex", "region", "children"], inplace=True)
df
```

✓ 0.0s

	age	bmi	smoker	charges
0	19	27.900	1	16884.92400
1	18	33.770	0	1725.55230
2	28	33.000	0	4449.46200
3	33	22.705	0	21984.47061
4	32	28.880	0	3866.85520
...	...	...	...	...
1333	50	30.970	0	10600.54830
1334	18	31.920	0	2205.98080
1335	18	36.850	0	1629.83350
1336	21	25.800	0	2007.94500
1337	61	29.070	1	29141.36030

1338 rows × 4 columns

Now, we are ready to train our model.



```
X_train, X_test = X[: int(len(X)*0.6)], X[int(len(X)*0.6) : ]
y_train, y_test = y[: int(len(y)*0.6)], y[int(len(y)*0.6) : ]
```

✓ 0.0s

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score
```

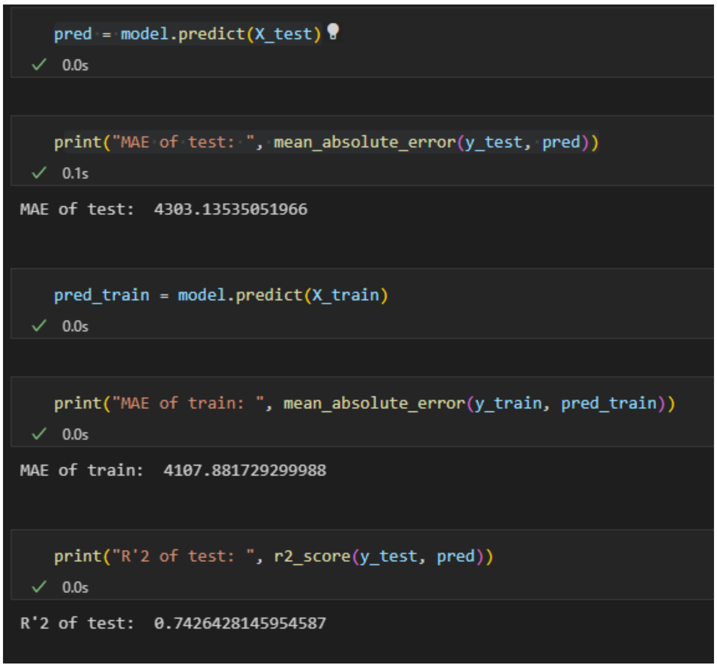
✓ 0.0s

```
model = LinearRegression()
model.fit(X_train, y_train)
```

✓ 0.0s

```
▾ LinearRegression
LinearRegression()
```

We split our dataset %60 for train and %40 test, and fit it with Linear Regression model.



```
pred = model.predict(X_test)
✓ 0.0s
```

```
print("MAE of test: ", mean_absolute_error(y_test, pred))
✓ 0.1s
```

```
MAE of test:  4303.13535051966
```

```
pred_train = model.predict(X_train)
✓ 0.0s
```

```
print("MAE of train: ", mean_absolute_error(y_train, pred_train))
✓ 0.0s
```

```
MAE of train:  4107.881729299988
```

```
print("R'2 of test: ", r2_score(y_test, pred))
✓ 0.0s
```

```
R'2 of test:  0.7426428145954587
```

When we calculate our scores, we can see our model performans is a bit low. We can increase our split with %70 or %80 to train model.