Insurance_Linear_Regression

In [ ]:
```
Author ~ Saurabh
Date ~ 04-Dec-21
```

In [1]:
```python
#importing lib
import os
import numpy as np
import pandas as pd

#for ploting
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
sns.set_style("darkgrid")
import ipywidgets as widgets
from IPython.display import display

#to supress warnings
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```python
#pandas profiling
import pandas_profiling as pp
```

In [3]:
```python
pwd
```

Out[3]:
```
'E:\\DataScience\\MachineLearning\\Insurance_Linear_Regression'
```

In [4]:
```python
#listdir
os.listdir("E:\\DataScience\\MachineLearning\\Insurance_Linear_Regression")
```

Out[4]:
```
['.ipynb_checkpoints',
 'Insurance_Data.csv',
 'Insurance_Linear_Regression.ipynb',
 'Insurance_profile_report.html']
```

In [5]:
```python
#read dataset..
df =pd.read_csv("E:\\DataScience\\MachineLearning\\Insurance_Linear_Regression\\Insurance_Data.csv")
```

In [6]:
```python
profile =df.profile_report(title="Insurance Data Profile Report")
```

In [7]:
```python
profile
```

Insurance Data Profile Report

Overview Variables Interactions Correlations Missing values Sample Duplicate rows

Overview

Overview Alerts 7 Reproduction

Dataset statistics

Number of variables	7
Number of observations	1338
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	1
Duplicate rows (%)	0.1%
Total size in memory	73.3 KiB
Average record size in memory	56.1 B

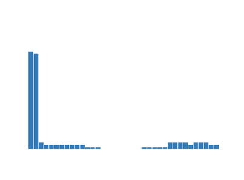
Variable types

Numeric	4
Categorical	2
Boolean	1

Variables

age
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	47	Minimum	18
Distinct	3.5%	Maximum	64



Out[7]:

In [8]:
```python
#profile report
profile.to_file(output_file="Insurance_profile_report.html")
```

In [9]:
```python
#top 5 columns of data
df.head()
```

Out[9]:

	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

In [10]:
```python
#info
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
```

In [11]:
```python
#shape
df.shape
```

Out[11]:
```
(1338, 7)
```

In [12]:
```python
#tail of dataframe
df.tail()
```

Out[12]:

	age	sex	bmi	children	smoker	region	charges
1333	50	male	30.97	3	no	northwest	10600.5483
1334	18	female	31.92	0	no	northeast	2205.9808
1335	18	female	36.85	0	no	southeast	1629.8335
1336	21	female	25.80	0	no	southwest	2007.9450
1337	61	female	29.07	0	yes	northwest	29141.3603

In [13]:
```python
#count of null values
df.isnull().sum()
```

Out[13]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [14]:
```python
#unique value per column
df.nunique()
```

Out[14]:
```
age           47
sex            2
bmi          548
children       6
smoker         2
region         4
charges     1337
dtype: int64
```

In [15]:
```python
# numerical data
df.corr()
```

Out[15]:

	age	bmi	children	charges
age	1.000000	0.109272	0.042469	0.299008
bmi	0.109272	1.000000	0.012759	0.198341
children	0.042469	0.012759	1.000000	0.067998
charges	0.299008	0.198341	0.067998	1.000000

In [33]:
```python
#categorical data
catogorical= []
lst =df.columns
for i in range(7):
    if(df[lst[i]].dtype =='object'):
        catogorical.append(lst[i])

print("catogorical columns :",catogorical)
```

```
catogorical columns : ['sex', 'smoker', 'region']
```

In [36]:
```python
#encoding
from sklearn.preprocessing import LabelEncoder
lb_encoder=LabelEncoder()
```

In [40]:
```python
df['sex'] =lb_encoder.fit_transform(df['sex'])
df['smoker']=lb_encoder.fit_transform(df['smoker'])
df['region'] =lb_encoder.fit_transform(df['region'])
```

In [41]:
```python
df.head()
```

Out[41]:

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

In [52]:
```python
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),cbar=True,square=True,annot=True,annot_kws={'size':10},fmt='.2f',cmap='Dark2_r')
```

Out[52]:
```
<AxesSubplot:>
```

In [44]:
```python
#input and target
X =df.iloc[:,:-1]
y =df.iloc[:,-1]
```

In [45]:
```python
#shape
print("Shape of Input :",X.shape)
print("Shape of Target:",y.shape)
```

```
Shape of Input : (1338, 6)
Shape of Target: (1338,)
```

In [47]:
```python
#train_test_split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.33,random_state=11)
```

In [49]:
```python
#shape
print("Shape X_train :",X_train.shape)
print("Shape y_train :",y_train.shape)
print("Shape X_test :",X_test.shape)
print("Shape y_test :",y_test.shape)
```

```
Shape X_train : (896, 6)
Shape y_train : (896,)
Shape X_test : (442, 6)
Shape y_test : (442,)
```

In [53]:
```python
#scaling the given data
from sklearn.preprocessing import StandardScaler
Scaler =StandardScaler()
X_train_scale =Scaler.fit_transform(X_train)
X_test_scale =Scaler.fit_transform(X_test)
```

Linear Regression

In [54]:
```python
#importing linear regession
from sklearn.linear_model import LinearRegression
```

In [56]:
```python
#fiiting the data to model
model =LinearRegression()
model.fit(X_train,y_train)
```

Out[56]:
```
LinearRegression()
```

In [57]:
```python
#y_pred : predicted values
y_pred=model.predict(X_test)
```

In [58]:
```python
#Report of model
print('Coefficients: \n', model.coef_)
print("Mean squared error: %.2f" % np.mean((model.predict(X_test) - y_test) ** 2))
print('Variance score: %.2f' % model.score(X_test, y_test))
```

```
Coefficients:
 [  262.1433078   -279.12041025   350.7461981    631.75967131
 23728.48775086  -398.06450479]
Mean squared error: 31843626.47
Variance score: 0.77
```

In [59]:
```python
def accuracy(X_test,y_test, y_pred):
    print('accuracy (R^2):\n', model.score(X_test, y_test)*100, '%')
```

In [60]:
```python
accuracy(X_test, y_test, y_pred)
```

```
accuracy (R^2):
 76.7984186524659 %
```

Xgboost

In [63]:
```python
#training with Xgboost regressor
import xgboost as xgb
modelX = xgb.XGBRegressor(objective ='reg:squarederror', learning_rate = 0.2, max_depth = 10, n_estimators = 1
modelX.fit(X_train, y_train)
```

Out[63]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.2, max_delta_step=0,
             max_depth=10, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [64]:
```python
y_hat =modelX.predict(X_test)
```

In [67]:
```python
result = modelX.score(X_test, y_test)
print("Accuracy : {}".format(result))
```

```
Accuracy : 0.8128600393963679
```

RandomForestRegressor

In [68]:
```python
from sklearn.ensemble import RandomForestRegressor
modelR=RandomForestRegressor()
modelR.fit(X_train,y_train)
```

Out[68]:
```
RandomForestRegressor()
```

In [69]:
```python
y1_hat =modelR.predict(X_test)
```

In [70]:
```python
result = modelR.score(X_test, y_test)
print("Accuracy : {}".format(result))
```

```
Accuracy : 0.8403300137546962
```

Interactive Display Widgets

In [75]:
```python
 age_widget = widgets.IntSlider(
    value=38,
    min=18,
    max=64,
    step=1,
    description="Age:"
)

bmi_widget = widgets.FloatSlider(
    value=30,
    min=15,
    max=54,
    step=0.01,
    description="BMI:"
)

children_widget = widgets.IntSlider(
    value=1,
    min=0,
    max=5,
    step=1,
    description="Children:"
)

sex_widget = widgets.ToggleButtons(
    options=[('female',0),('Male',1)],
    description="Sex:"
)

smoker_widget = widgets.ToggleButtons(
    options=[('no',0),('yes',1)],
    description="Smoker:"
)

region_widget = widgets.Dropdown(
    options=[('northeast',4),('Southeast',2),('Northwest',3),('Southwest',1)],
    description="Region:"
)

predict_btn = widgets.Button(
    description="Predict"
)

prediction_out = widgets.Output()


def make_prediction(btn):
    x = pd.DataFrame({
        'age':      age_widget.value,
        'sex':      sex_widget.value,
        'bmi':      bmi_widget.value,
        'children': children_widget.value,
        'smoker':   smoker_widget.value,
        'region':   region_widget.value
    }, index=[0])

    prediction = modelR.predict(x)

    with prediction_out:
        prediction_out.clear_output()
        print("Prediction: {:.4f}".format(prediction[0]))


predict_btn.on_click(make_prediction)


display(age_widget, bmi_widget, children_widget, sex_widget, smoker_widget, region_widget, predict_btn, predict
```

In [ ]: