Practice Project: Insurance Cost Analysis

Estimated time needed: **75** minutes

In this project, you have to perform analytics operations on an insurance database that uses the below mentioned parameters.

Parameter	Description	Content type
age	Age in years	integer
gender	Male or Female	integer (1 or 2)
bmi	Body mass index	float
no_of_children	Number of children	integer
smoker	Whether smoker or not	integer (0 or 1)
region	Which US region - NW, NE, SW, SE	integer (1,2,3 or 4 respectively)
charges	Annual Insurance charges in USD	float

Objectives

In this project, you will:

- Load the data as a `pandas` dataframe
- Clean the data, taking care of the blank entries
- Run exploratory data analysis (EDA) and identify the attributes that most affect the `charges`
- Develop single variable and multi variable Linear Regression models for predicting the `charges`
- Use Ridge regression to refine the performance of Linear regression models.

Setup

For this lab, we will be using the following libraries:

- `skillsnetwork` to download the data
- `pandas` for managing the data.
- `numpy` for mathematical operations.
- `sklearn` for machine learning and machine-learning-pipeline related functions.
- `seaborn` for visualizing the data.
- `matplotlib` for additional plotting tools.

The following required libraries are **not** pre-installed in the Skills Network Labs environment. **You will need to run the following cell** to install them:

In [1]:
```
import piplite
await piplite.install('seaborn')
```

Importing Required Libraries

We recommend you import all required libraries in one place (here):

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score, train_test_split
```

▸ Click here for Solution

Download the dataset to this lab environment

Run the cell below to load the dataset to this lab environment.

In [3]:
```
from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

In [4]:
```
filepath = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/medical_insu
```

In [5]:
```
await download(filepath, "insurance.csv")
file_name="insurance.csv"
```

In [20]:
```
df = pd.read_csv(file_name)
```

> Note: This version of the lab is working on JupyterLite, which requires the dataset to be downloaded to the interface.While working on the downloaded version of this notebook on their local machines(Jupyter Anaconda), the learners can simply **skip the steps above,** and simply use the URL directly in the `pandas.read_csv()` function. You can uncomment and run the statements in the cell below.

In [ ]:
```
#filepath = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/medical_ins
#df = pd.read_csv(filepath, header=None)
```

Task 1 : Import the dataset

Import the dataset into a `pandas` dataframe. Note that there are currently no headers in the CSV file.

Print the first 10 rows of the dataframe to confirm successful loading.

In [21]:
```
df.head(10)
```

Out[21]:

	19	1	27.9	0	1.1	3	16884.924
0	18	2	33.770	1	0	4	1725.55230
1	28	2	33.000	3	0	4	4449.46200
2	33	2	22.705	0	0	1	21984.47061
3	32	2	28.880	0	0	1	3866.85520
4	31	1	25.740	0	?	4	3756.62160
5	46	1	33.440	1	0	4	8240.58960
6	37	1	27.740	3	0	1	7281.50560
7	37	2	29.830	2	0	2	6406.41070
8	60	1	25.840	0	0	1	28923.13692
9	25	2	26.220	0	0	2	2721.32080

▸ Click here for Solution

Add the headers to the dataframe, as mentioned in the project scenario.

In [22]:
```
df.columns = ["age","gender","bmi","no_of_children","smoker","region","charges"]
```

▸ Click here for Solution

Now, replace the '?' entries with 'NaN' values.

In [24]:
```
df.replace('?',np.nan,inplace=True)
```

▸ Click here for Solution

Task 2 : Data Wrangling

Use `dataframe.info()` to identify the columns that have some 'Null' (or NaN) information.

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2771 entries, 0 to 2770
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             2767 non-null   object
 1   gender          2771 non-null   int64
 2   bmi             2771 non-null   float64
 3   no_of_children  2771 non-null   int64
 4   smoker          2764 non-null   object
 5   region          2771 non-null   int64
 6   charges         2771 non-null   float64
dtypes: float64(2), int64(3), object(2)
memory usage: 130.0+ KB
```

► Click here for Solution

Handle missing data:

- For continuous attributes (e.g., age), replace missing values with the mean.
- For categorical attributes (e.g., smoker), replace missing values with the most frequent value.
- Update the data types of the respective columns.
- Verify the update using `df.info()` .

In [ ]:
```
mean_age = df['age'].astype('float').mean(axis=0)
df["age"].replace(np.nan, mean_age, inplace=True)

is_smoker = df['smoker'].value_counts().idxmax()
df["smoker"].replace(np.nan, is_smoker, inplace=True)

df[["age","smoker"]] = df[["age","smoker"]].astype("int")
```

► Click here for Solution

Also note, that the `charges` column has values which are more than 2 decimal places long. Update the `charges` column such that all values are rounded to nearest 2 decimal places. Verify conversion by printing the first 5 values of the updated dataframe.

In [32]:
```
df['charges'] = np.round(df['charges'],2)
```

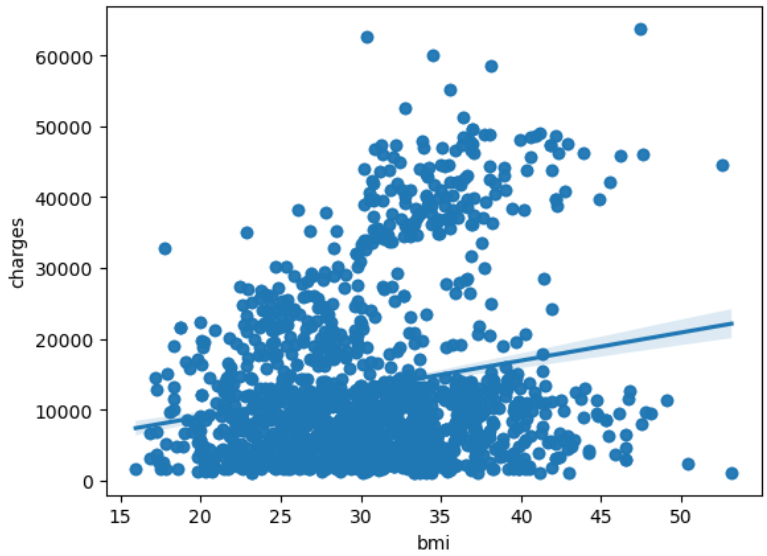
► Click here for Solution

Task 3 : Exploratory Data Analysis (EDA)

Implement the regression plot for `charges` with respect to `bmi` .

In [33]:
```
sns.regplot(x='bmi',y='charges',data=df)
```

Out[33]:
```
<AxesSubplot:xlabel='bmi', ylabel='charges'>
```

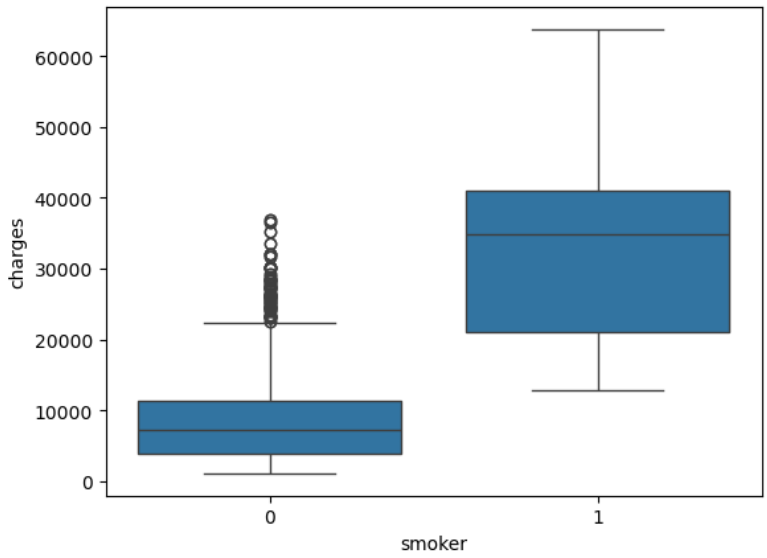


► Click here for Solution

Implement the box plot for `charges` with respect to `smoker`.

In [34]:
```
sns.boxplot(x='smoker',y='charges',data=df)
```

Out[34]:
```
<AxesSubplot:xlabel='smoker', ylabel='charges'>
```



▸ Click here for Solution

Print the correlation matrix for the dataset.

In [35]:
```
df.corr()
```

Out[35]:

	age	gender	bmi	no_of_children	smoker	region	charges
age	1.000000	-0.026584	0.112859	0.037126	-0.022290	-0.006969	0.298892
gender	-0.026584	1.000000	0.042766	0.015693	0.083125	0.022360	0.062959
bmi	0.112859	0.042766	1.000000	-0.001642	0.011824	0.271200	0.199906
no_of_children	0.037126	0.015693	-0.001642	1.000000	0.007016	-0.025594	0.066551
smoker	-0.022290	0.083125	0.011824	0.007016	1.000000	0.053839	0.789141
region	-0.006969	0.022360	0.271200	-0.025594	0.053839	1.000000	0.054018
charges	0.298892	0.062959	0.199906	0.066551	0.789141	0.054018	1.000000

▸ Click here for Solution

Task 4 : Model Development

Fit a linear regression model that may be used to predict the `charges` value, just by using the `smoker` attribute of the dataset. Print the R^2 score of this model.

In [38]:
```
lr = LinearRegression()
lr.fit(df[['smoker']],df[['charges']])
print(lr.score(df[['smoker']],df[['charges']]))
```

```
0.6227430402464125
```

▸ Click here for Solution

Fit a linear regression model that may be used to predict the `charges` value, just by using all other attributes of the dataset. Print the R^2 score of this model. You should see an improvement in the performance.

In [40]:
```
X = df.copy()
X.drop('charges',axis=1,inplace=True)
Y = df[['charges']]
lrm = LinearRegression()
lrm.fit(X,Y)
print(lrm.score(X,Y))
```

```
0.7505888664568174
```

▶ Click here for Solution

Create a training pipeline that uses `StandardScaler()`, `PolynomialFeatures()` and `LinearRegression()` to create a model that can predict the `charges` value using all the other attributes of the dataset. There should be even further improvement in the performance.

In [43]:
```
pipe = Pipeline([('scale',StandardScaler()),('polynomial',PolynomialFeatures()),('model',LinearRegression())])
pipe.fit(X.astype('float'),Y)
ypipe = pipe.predict(X.astype('float'))
print(r2_score(Y,ypipe))
```

```
0.8452536178009997
```

▶ Click here for Solution

Task 5 : Model Refinement

Split the data into training and testing subsets, assuming that 20% of the data will be reserved for testing.

In [44]:
```
x_train,x_test,y_train,y_test = train_test_split(X,Y, test_size=0.2,random_state=1)
```

▶ Click here for Solution

Initialize a Ridge regressor that used hyperparameter $\alpha = 0.1$. Fit the model using training data data subset. Print the R^2 score for the testing data.

In [46]:
```
rm = Ridge(alpha=0.1)
rm.fit(x_train,y_train)
yhat = rm.predict(x_test)
print(r2_score(y_test,yhat))
```

```
0.7254198858412217
```

▶ Click here for Solution

Apply polynomial transformation to the training parameters with degree=2. Use this transformed feature set to fit the same regression model, as above, using the training subset. Print the R^2 score for the testing subset.

In [49]:
```
pr = PolynomialFeatures(degree=2)
x_train_pr = pr.fit_transform(x_train)
x_test_pr = pr.fit_transform(x_test)

rm1 = Ridge(alpha=0.1)
rm1.fit(x_train_pr,y_train)
yhat1 = rm1.predict(x_test_pr)
print(r2_score(y_test,yhat1))
```

```
0.8208413195172275
```

▶ Click here for Solution

Congratulations! You have completed this project

Authors

Abhishek Gagneja

Vicky Kuo