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:

```
import piplite
await piplite.install('seaborn')
await piplite.install('skillsnetwork')
import skillsnetwork
```

Importing Required Libraries

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

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
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
```

Download the dataset to this lab environment

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

```
filepath = 'https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/
medical_insurance_dataset.csv'
await skillsnetwork.download(filepath,'./insurance.csv')
path = './insurance.csv'
```

```
Downloading medical_insurance_dataset.csv: 100%|██████████|
78536/78536 [00:00<00:00, 78504256.18it/s]

Saved as 'insurance.csv'
```

> Note: In case you are using the lab offline on your local machines, you may simply use the URL in `filepath` in the `pd.read_csv()` function to access the data.

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.

```
df = pd.read_csv(path, header=None)
print(df.head(10))
```

```
    0  1       2  3  4  5            6
0  19  1  27.900  0  1  3  16884.92400
1  18  2  33.770  1  0  4   1725.55230
2  28  2  33.000  3  0  4   4449.46200
3  33  2  22.705  0  0  1  21984.47061
4  32  2  28.880  0  0  1   3866.85520
5  31  1  25.740  0  ?  4   3756.62160
6  46  1  33.440  1  0  4   8240.58960
7  37  1  27.740  3  0  1   7281.50560
8  37  2  29.830  2  0  2   6406.41070
9  60  1  25.840  0  0  1  28923.13692
```

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

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

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

```
df.replace("?", np.nan, inplace = True)
```

Task 2 : Data Wrangling

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

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2772 entries, 0 to 2771
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             2768 non-null   object
 1   gender          2772 non-null   int64
 2   bmi             2772 non-null   float64
 3   no_of_children  2772 non-null   int64
 4   smoker          2765 non-null   object
 5   region          2772 non-null   int64
 6   charges         2772 non-null   float64
dtypes: float64(2), int64(3), object(2)
memory usage: 130.0+ KB
None
```

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()`.

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

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

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

df['charges'] = df["charges"].round(2)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2772 entries, 0 to 2771
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             2772 non-null   int32
 1   gender          2772 non-null   int64
 2   bmi             2772 non-null   float64
 3   no_of_children  2772 non-null   int64
 4   smoker          2772 non-null   object
 5   region          2772 non-null   int64
 6   charges         2772 non-null   float64
 7   smoke           2772 non-null   int32
dtypes: float64(2), int32(2), int64(3), object(1)
memory usage: 140.8+ KB
None
```

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.

```
df[["charges"]]= np.round(df[["charges"]],2)
print(df.head())
```

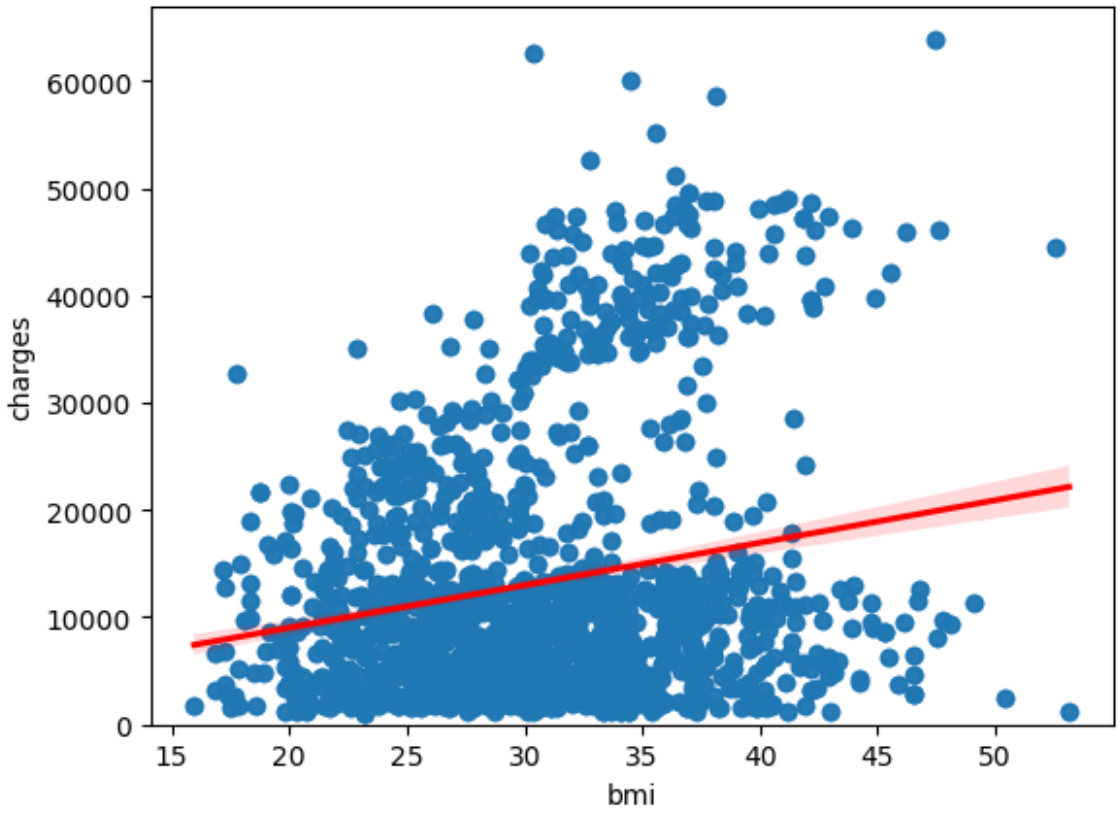
```
   age  gender     bmi  no_of_children smoker  region   charges  smoke
0   19       1  27.900               0      1       3  16884.92      1
1   18       2  33.770               1      0       4   1725.55      0
2   28       2  33.000               3      0       4   4449.46      0
3   33       2  22.705               0      0       1  21984.47      0
4   32       2  28.880               0      0       1   3866.86      0
```

Task 3 : Exploratory Data Analysis (EDA)

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

```
sns.regplot(x="bmi", y="charges", data=df, line_kws={"color":"red"})
plt.ylim(0,)
```

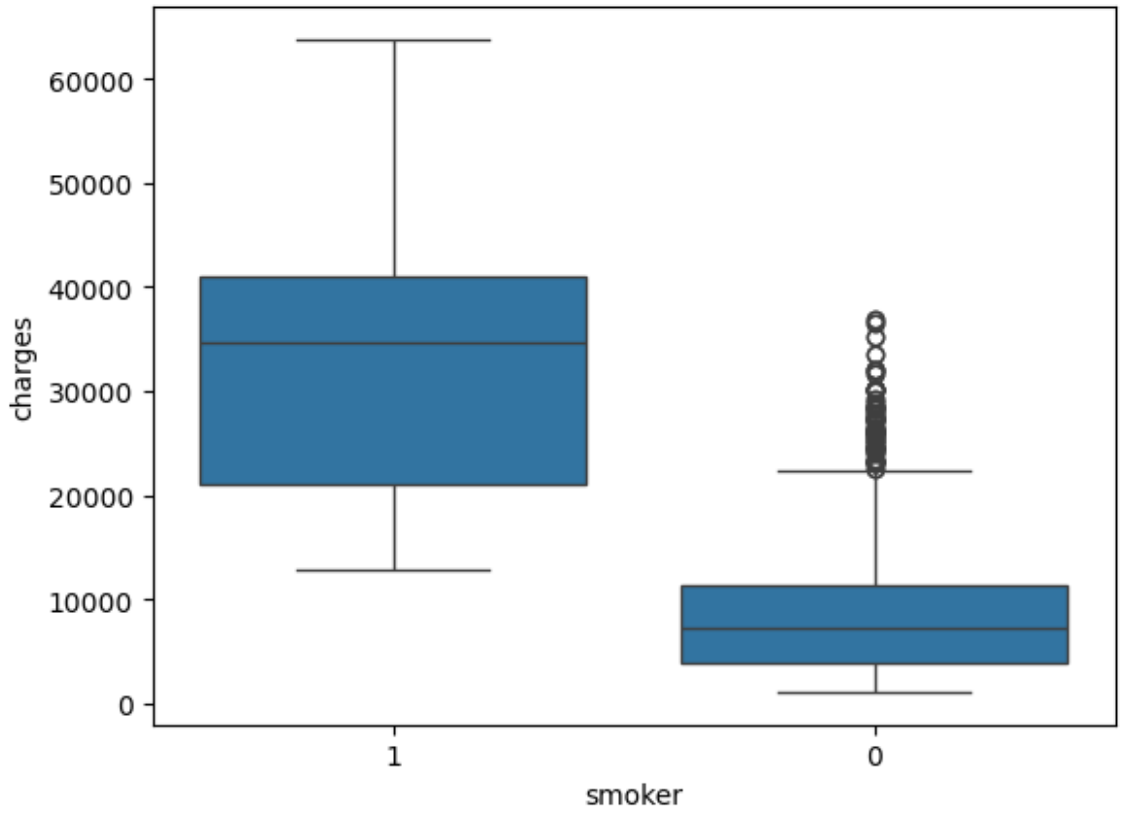
```
(0.0, 66902.85800000001)
```



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

```
sns.boxplot(x="smoker", y="charges", data=df)
```

```
<AxesSubplot:xlabel='smoker', ylabel='charges'>
```



Print the correlation matrix for the dataset.

```
print(df.corr())
```

```
<ipython-input-12-23236a4e6045>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  print(df.corr())
```

```
                     age    gender       bmi  no_of_children    region
\
age             1.000000 -0.026046  0.113048        0.037574 -0.007167

gender         -0.026046  1.000000  0.042924        0.016020  0.022213

bmi             0.113048  0.042924  1.000000       -0.001492  0.271119

no_of_children  0.037574  0.016020 -0.001492        1.000000 -0.025717

region         -0.007167  0.022213  0.271119       -0.025717  1.000000

charges         0.298624  0.062837  0.199846        0.066442  0.054058

smoke          -0.023286  0.082326  0.011489        0.006362  0.054077
```

```
                  charges     smoke
age              0.298624 -0.023286
gender           0.062837  0.082326
bmi              0.199846  0.011489
no_of_children   0.066442  0.006362
region           0.054058  0.054077
charges          1.000000  0.788783
smoke            0.788783  1.000000
```

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.

```
X=df[["smoker"]]
Y=df[["charges"]]
lm = LinearRegression()
lm.fit(X,Y)
print(lm.score(X,Y))
```

```
0.6221791733924185
```

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.

```
Z=df[["age", "gender", "bmi", "no_of_children", "smoker", "region"]]
lm.fit(Z,Y)
print(lm.score(Z, Y))
```

```
0.7504083820289634
```

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.

```
Input = [('scale', StandardScaler()), ('polynomial',
PolynomialFeatures(include_bias=False)), ('model',
LinearRegression())]
pipe=Pipeline(Input)
Z=Z.astype(float)
pipe.fit(Z,Y)
ypipe=pipe.predict(Z)
print(r2_score(Y, ypipe))
```

```
0.845256277259561
```

Task 5 : Model Refinement

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

```
x_train, x_test, y_train, y_test = train_test_split(Z, Y,
test_size=0.2, random_state=1)
```

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.

```
RidgeModel= Ridge(alpha=0.1)
RidgeModel.fit(x_train, y_train)
yhat= RidgeModel.predict(x_test)
print(r2_score(y_test,yhat))
```

```
0.6760807731582406
```

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.

```
pr= PolynomialFeatures(degree=2)
x_train_pr = pr.fit_transform(x_train)
x_test_pr = pr.fit_transform(x_test)
RidgeModel.fit(x_train_pr, y_train)
y_hat = RidgeModel.predict(x_test_pr)
print(r2_score(y_test, y_hat))
```

```
0.7835631107608061
```

Congratulations! You have completed this project

Authors

Abhishek Gagneja

Vicky Kuo

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-09-16	0.1	Abhishek Gagneja	Initial Version Created
2023-09-19	0.2	Vicky Kuo	Reviewed and Revised