Insurance Cost Analysis

September 28, 2024

1 Practice Project : Insurance Cost Analysis

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	Wether 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

1.0.1 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.

1.0.2 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.

Importing required libraries

[3]:
```
import seaborn as sns
import pandas as pd
import numpy as np
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

[13]:
```
filepath = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBMDeveloperSkillsNetwork-DA0101EN-Coursera/medical_insurance_dataset.csv'
df = pd.read_csv(filepath, header=None)
```

1.0.3 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.

[14]:
```
df.head(10)
```

[14]:

	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.

[15]:
```
df.columns = ["age", "gender", "bmi", "no_of_children", "smoker", "region",␣
↪"charges"]
```

[23]:
```
df.head(10)
```

[23]:

	age	gender	bmi	no_of_children	smoker	region	charges
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  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
```

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

[16]:
```
df.isna().sum()
```

[16]:
```
age               0
gender            0
bmi               0
no_of_children    0
smoker            0
region            0
charges           0
dtype: int64
```

[17]:
```
(df == '?').sum()
```

[17]:
```
age               4
gender            0
bmi               0
no_of_children    0
smoker            7
region            0
charges           0
dtype: int64
```

[18]:
```
df.replace('?', np.nan, inplace=True)
```

[19]:
```
df.isna().sum()
```

[19]:
```
age               4
gender            0
bmi               0
no_of_children    0
smoker            7
region            0
charges           0
dtype: int64
```

1.0.4 Task 2: Data Wrangling

[20]:
```
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: 151.7+ KB
```

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()`.

[25]:
```python
#Continious Attribute
mean_age = df['age'].astype(float).mean(axis=0)
df['age'].replace(np.nan, mean_age, inplace=True)
```

```
C:\Users\bendh\AppData\Local\Temp\ipykernel_28596\2234849207.py:3:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['age'].replace(np.nan, mean_age, inplace=True)
```

[22]:
```python
#Categorical Value
is_smoker = df['smoker'].value_counts().idxmax()
df['smoker'].replace(np.nan, is_smoker, inplace=True)
```

```
C:\Users\bendh\AppData\Local\Temp\ipykernel_28596\2974893863.py:3:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
```

```
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


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

[29]:
```python
#updating the datatypes
df[['age', 'smoker']]= df[['age', 'smoker']].astype(int)
```

[30]:
```python
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             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   int32
 5   region          2772 non-null   int64
 6   charges         2772 non-null   float64
dtypes: float64(2), int32(2), int64(3)
memory usage: 130.1 KB
```

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.

[32]:
```python
df[['charges']] = np.round(df[['charges']],2)
df.head(10)
```

[32]:

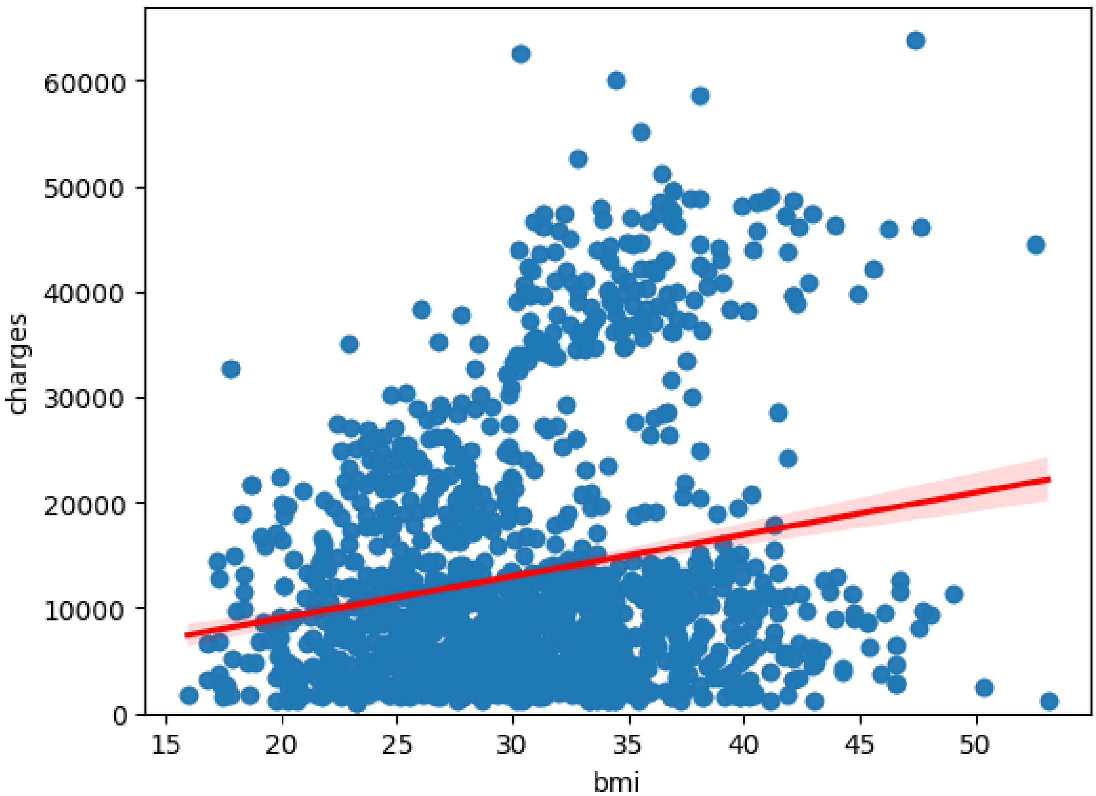
	age	gender	bmi	no_of_children	smoker	region	charges
0	19	1	27.900	0	1	3	16884.92
1	18	2	33.770	1	0	4	1725.55
2	28	2	33.000	3	0	4	4449.46
3	33	2	22.705	0	0	1	21984.47
4	32	2	28.880	0	0	1	3866.86
5	31	1	25.740	0	0	4	3756.62
6	46	1	33.440	1	0	4	8240.59
7	37	1	27.740	3	0	1	7281.51
8	37	2	29.830	2	0	2	6406.41
9	60	1	25.840	0	0	1	28923.14

1.0.5 Task 3 : Exploratory Data Analysis (EDA)

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

[33]:
```
sns.regplot(x='bmi', y='charges', data=df, line_kws ={'color' : 'red'})
plt.ylim(0,)
```

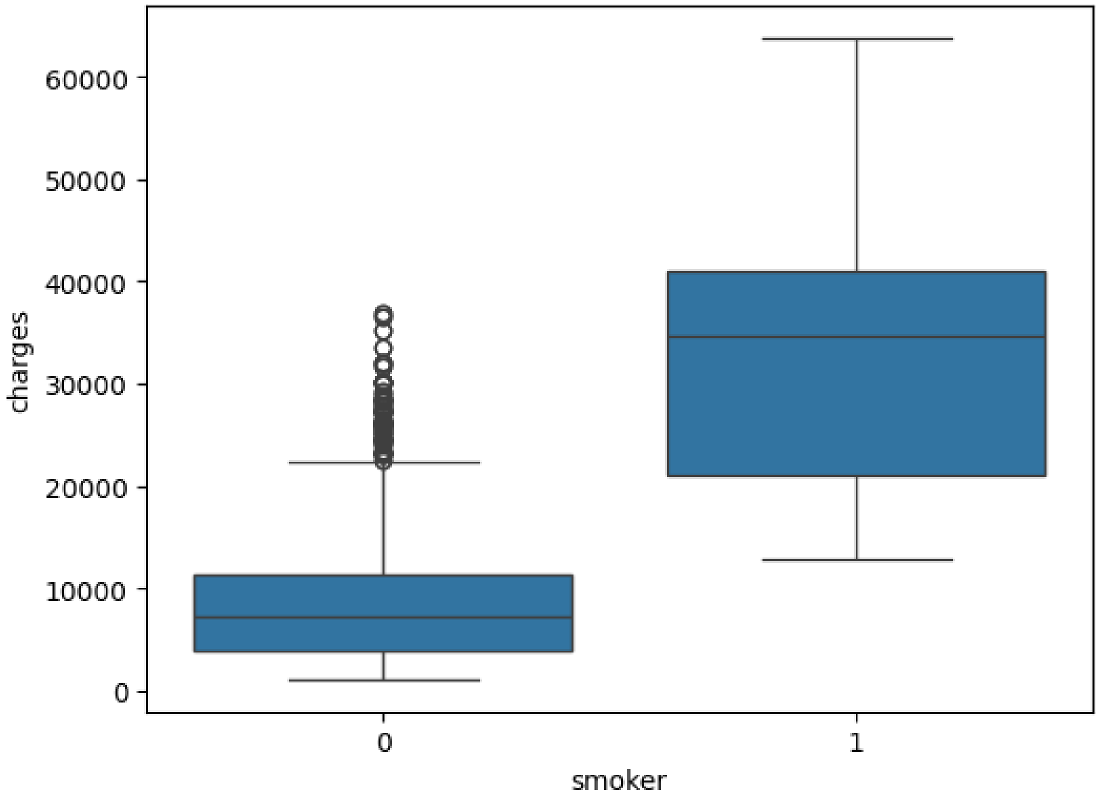
[33]: (0.0, 66902.85800000001)



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

[37]:
```
sns.boxplot(x ='smoker', y='charges', data=df)
```

[37]: <Axes: xlabel='smoker', ylabel='charges'>



Print the correlation matrix for the dataset.

```
[38]: df.corr()
```

```
[38]:                   age    gender       bmi  no_of_children    smoker  \
age             1.000000 -0.026046  0.113048        0.037574 -0.023286
gender         -0.026046  1.000000  0.042924        0.016020  0.082326
bmi             0.113048  0.042924  1.000000       -0.001492  0.011489
no_of_children  0.037574  0.016020 -0.001492        1.000000  0.006362
smoker         -0.023286  0.082326  0.011489        0.006362  1.000000
region         -0.007167  0.022213  0.271119       -0.025717  0.054077
charges         0.298624  0.062837  0.199846        0.066442  0.788783

                  region   charges
age            -0.007167  0.298624
gender          0.022213  0.062837
bmi             0.271119  0.199846
no_of_children -0.025717  0.066442
smoker          0.054077  0.788783
region          1.000000  0.054058
charges         0.054058  1.000000
```

1.0.6 Task 4 : Model Developement

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.

[39]:
```
X = df[['smoker']]
Y = df[['charges']]
lm = LinearRegression()
lm.fit(X,Y)
lm.score(X,Y)
```

[39]: 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.

[41]:
```
Z = df[['age', 'gender', 'bmi', 'no_of_children', 'smoker', 'region']]
lm.fit(Z,Y)
lm.score(Z,Y)
```

[41]: 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.

[47]:
```
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)
r2_score(Y, ypipe) #use r2_score with pipelines : r2_score(y_true, y_pred)
```

[47]: 0.8452516370437424

1.0.7 Task 5 : Model Refinement

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

[48]:
```
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.

[49]:
```
RidgeModel = Ridge(alpha=0.1)
RidgeModel.fit(x_train, y_train)
```

```
yhat = RidgeModel.predict(x_test)
r2_score(y_test, yhat)
```

[49]: 0.6760807731582404

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.

[53]:
```
pr = PolynomialFeatures(degree=2)
x_train_pr = pr.fit_transform (x_train)
x_test_pr = pr.transform (x_test)
RidgeModel.fit(x_train_pr, y_train)
y_hat = RidgeModel.predict(x_test_pr)
r2_score(y_test, y_hat)
```

[53]: 0.7835631107608146