Linear Regression for Medical Cost

March 14, 2025

[25]:
```
## Import the required libraries
import pandas as pd       # For data manipulation
import numpy as np        # For numerical computations
import matplotlib.pyplot as plt  # For plotting
import seaborn as sns     # For statistical data visualization
from sklearn.model_selection import train_test_split  # For splitting data
from sklearn.linear_model import LinearRegression  # For linear regression
from sklearn.metrics import mean_squared_error, r2_score  # For evaluation
import joblib
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
## Load the dataset
df = pd.read_csv("C:/Users/PC/OneDrive/Desktop/Data Science/Datasets/Datasets/
 ↪medical costs.csv")
```

[3]:
```
## View the first few observations of the dataset
df.head(10)
```

[3]:

	Age	Sex	BMI	Children	Smoker	Region	Medical Cost
0	58	male	15.6	2	yes	northwest	17907.54
1	24	male	29.8	0	yes	northeast	16312.64
2	50	male	29.0	5	no	northwest	6819.21
3	35	male	34.0	1	no	southeast	5247.87
4	31	female	17.6	3	yes	southeast	17525.49
5	56	female	35.2	5	no	northeast	7829.12
6	42	male	28.8	5	no	northwest	6668.57
7	20	male	34.3	0	yes	northeast	16409.13
8	47	female	19.1	2	yes	southeast	18024.82
9	61	male	30.2	1	yes	northwest	18618.26

[4]:
```
## Assess the structure of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
```

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Age           10000 non-null  int64
 1   Sex           10000 non-null  object
 2   BMI           10000 non-null  float64
 3   Children      10000 non-null  int64
 4   Smoker        10000 non-null  object
 5   Region        10000 non-null  object
 6   Medical Cost  10000 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 547.0+ KB
```

[6]:
```
## Check for duplicates
df.duplicated().sum()
```

[6]:
```
0
```

[7]:
```
## Check for missing values
df.isnull().sum()
```

[7]:
```
Age             0
Sex             0
BMI             0
Children        0
Smoker          0
Region          0
Medical Cost    0
dtype: int64
```

[8]:
```
## Summary statistics
df.describe()
```

[8]:

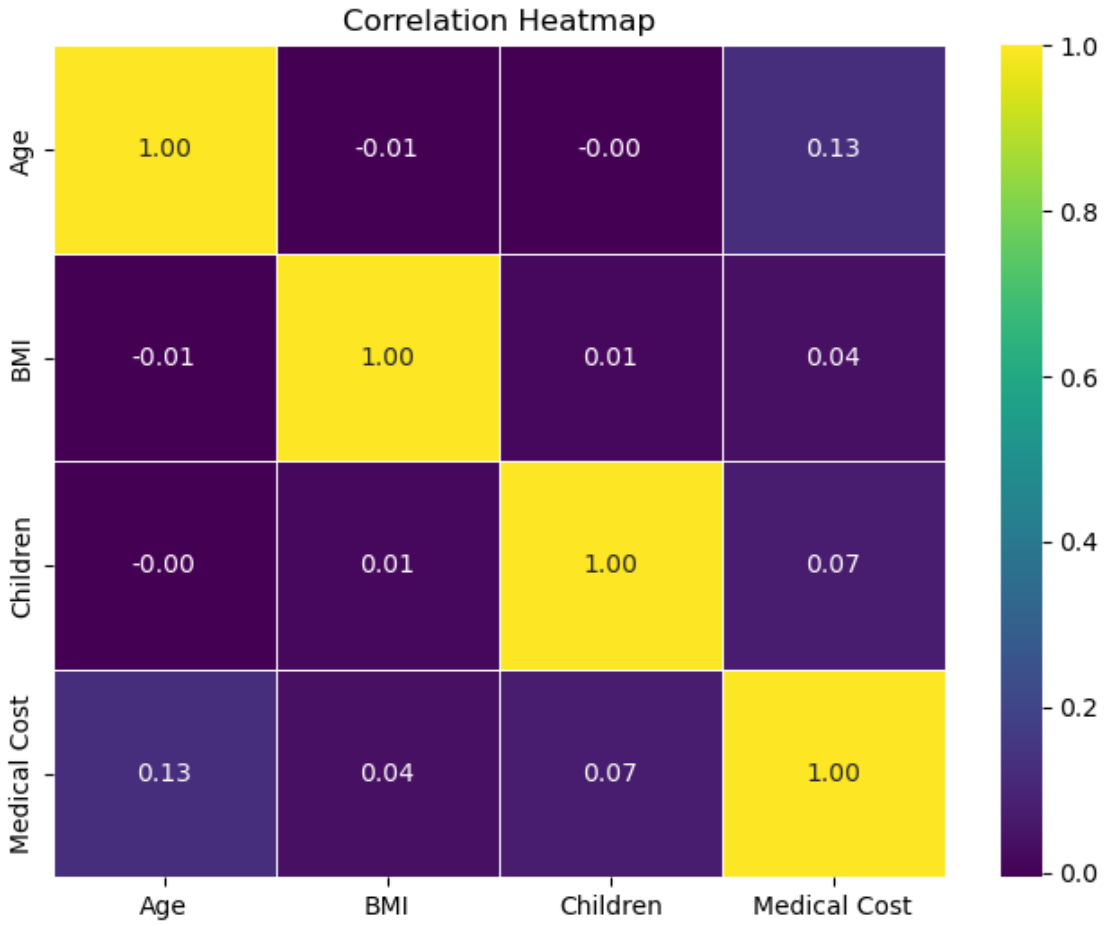
	Age	BMI	Children	Medical Cost
count	10000.000000	10000.00000	10000.000000	10000.000000
mean	41.678400	27.40301	2.501700	11898.932216
std	13.807724	7.22896	1.701672	6073.875834
min	18.000000	15.00000	0.000000	3617.090000
25%	30.000000	21.10000	1.000000	5909.925000
50%	42.000000	27.40000	2.000000	7957.430000
75%	54.000000	33.70000	4.000000	17931.962500
max	65.000000	40.00000	5.000000	20268.210000

[16]:
```
## Perform correlation analysis
numeric_vars = df.select_dtypes(include = ["float64", "int64"])
correlation_matrix = numeric_vars.corr()
print(correlation_matrix)
```

```
                Age       BMI  Children  Medical Cost
Age        1.000000 -0.005848 -0.004944      0.125649
```

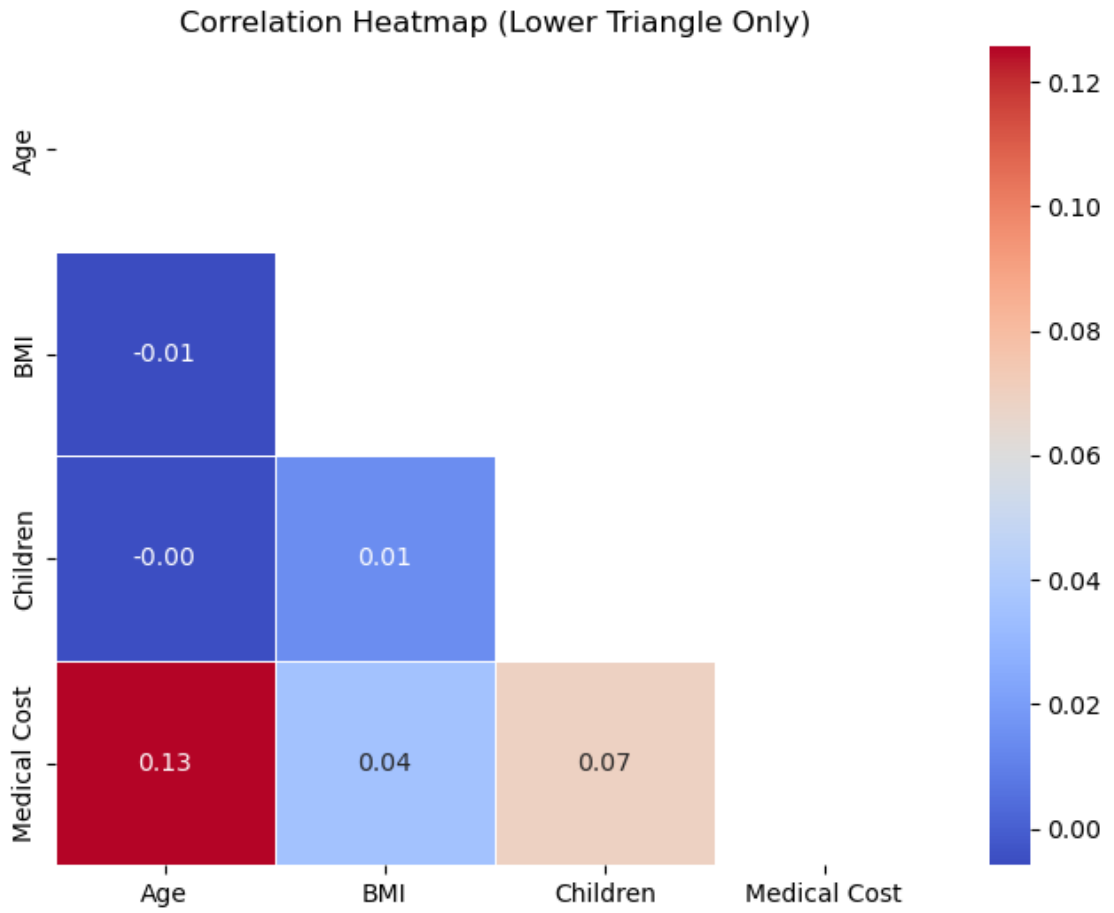
```
BMI           -0.005848  1.000000  0.014562       0.035249
Children      -0.004944  0.014562  1.000000       0.069575
Medical Cost   0.125649  0.035249  0.069575       1.000000
```

[18]:
```
## Visualize your correlation matrix
plt.figure(figsize=(8,6))
sns.heatmap(correlation_matrix, annot=True, cmap="viridis", fmt=".2f",␣
 ↪linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```



[19]:
```
## Self-Correlation
plt.figure(figsize=(8,6))
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))  # Mask upper␣
 ↪triangle
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f",␣
 ↪linewidths=0.5, mask=mask)
plt.title("Correlation Heatmap")
```

```
plt.show()
```



[26]:
```
## Label encoding
## Select categorical columns
categorical_cols = df.select_dtypes(include = ["object"]).columns
## Initialize the label encoder
label_encoder = LabelEncoder()
## Apply label encooding to selected columns
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])
```

[27]:
```
## Define Independent and Dependent Variables
X = df.drop(columns = ["Medical Cost"])
y = df["Medical Cost"]
```

[28]:
```
## Split the Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[29]:
```
## Train the Linear Regression Model
# Create model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
```

[29]:
```
LinearRegression()
```

[30]:
```
## Check Model Coefficients
print(f"Intercept (b0): {model.intercept_}")  # The constant term
print(f"Coefficient (b1): {model.coef_[0]}")  # The slope
```

```
Intercept (b0): 2537.682071174586
Coefficient (b1): 49.94317281914052
```

[31]:
```
## Make Predictions
y_pred = model.predict(X_test)
```

[36]:
```
## Model Evaluation
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
r2 = r2_score(y_test, y_pred)
print(f"R-squared Score: {r2}")
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE:", rmse)
```

```
Mean Squared Error: 84935.35538635665
R-squared Score: 0.9976894506758467
RMSE: 291.43670905765566
```

[37]:
```
# Extract feature importance (coefficients)
feature_importance = pd.DataFrame({'Feature': X.columns, 'Importance': np.
 ↪abs(model.coef_)})
feature_importance = feature_importance.sort_values(by="Importance",␣
 ↪ascending=False)
print(feature_importance)
```

```
    Feature    Importance
4    Smoker  12001.205596
3  Children    201.771058
0       Age     49.943173
2       BMI     28.889135
1       Sex      8.785177
5    Region      1.364502
```

[40]:
```
plt.figure(figsize=(8, 5))
sns.barplot(x=feature_importance['Importance'],␣
 ↪y=feature_importance['Feature'], palette='coolwarm')
```

```
plt.xlabel("Importance (Absolute Coefficients)")
plt.ylabel("Features")
plt.title("Feature Importance in Linear Regression")
plt.show()
```