Ozan MÖHÜRCÜ

Data Analyst | Data Scientist

👋 Hello! I am Ozan, a data analyst who is open to learning and who improves myself in analytical thinking and producing data-driven solutions. I have successfully completed my analyst training and am currently focusing on data science and increasing my competencies in this field.

📊 What Do I Know?

I can extract meaningful results from data by working with Python, SQL and data visualization tools. I am constantly improving myself in statistical analysis and reporting. I aim to solve problems and support decision processes with the insights I obtain.

📚 What Am I Doing Right Now? In my data science training, I am gaining knowledge on topics such as machine learning and big data analytics. In addition, I am looking for opportunities to put my theoretical knowledge into practice by gaining experience in real-world projects.

🎯 My Goal: To contribute to the growth goals of companies by using my talents in data analysis and data science in a way that will create value in the business world. I am here to learn new information and to constantly improve by sharing my experiences.

If you would like to discuss projects, collaborate or share experiences, I would be happy to connect!

LinkedIn GitHub

Medical Insurance Cost

About The Dataset

- **Age** : The age of the individual, which can influence health risks and insurance charges. 🎂📊
- **Sex** : The gender of the individual (Male or Female). This may be used to analyze how health insurance charges differ by gender. 🚹🚺
- **BMI (Body Mass Index)** : A measure of body fat based on height and weight. It helps assess an individual's overall health and risk of certain diseases, which can impact insurance costs. ⚖️💪
- **Children**
  The number of children/dependents the individual has. This can be relevant for insurance plans that cover family members and influence the total charges. 👶👨‍👩‍👧‍👦
- **Smoker** : Whether the individual smokes (Yes or No). Smoking can lead to higher health risks, which is reflected in higher insurance premiums. 🚬❌
- **Region** : The geographical location of the individual (e.g., northeast, southwest, etc.). This may affect the cost of health insurance depending on local healthcare costs and policies. 🌍🏙️
- **Charges** : The medical charges billed to the individual by the insurance company. These are influenced by factors such as age, smoking status, BMI, and region, and are the dependent variable of interest in insurance models. 💰💳

1.1 Libraries and Utilities

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
warnings.warn("this will not show")
```

Loading data

In [2]:

```
df = pd.read_csv("/kaggle/input/insurance/insurance.csv")
df.head()
df1 = df.copy() # We took a copy of our original data because we will be work
```

In [3]:

```
df.info()

# The `df.info()` method provides a quick overview of a pandas DataFrame's st
# and memory usage for each column, which is crucial for understanding the da
```

```
<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 [4]:

```
df.describe(exclude = 'object').style.background_gradient(cmap='BuPu')

# This method provides a transposed summary of the descriptive statistics for
# min, 25th percentile, median, 75th percentile, and max for numerical column
```

Out[4]:

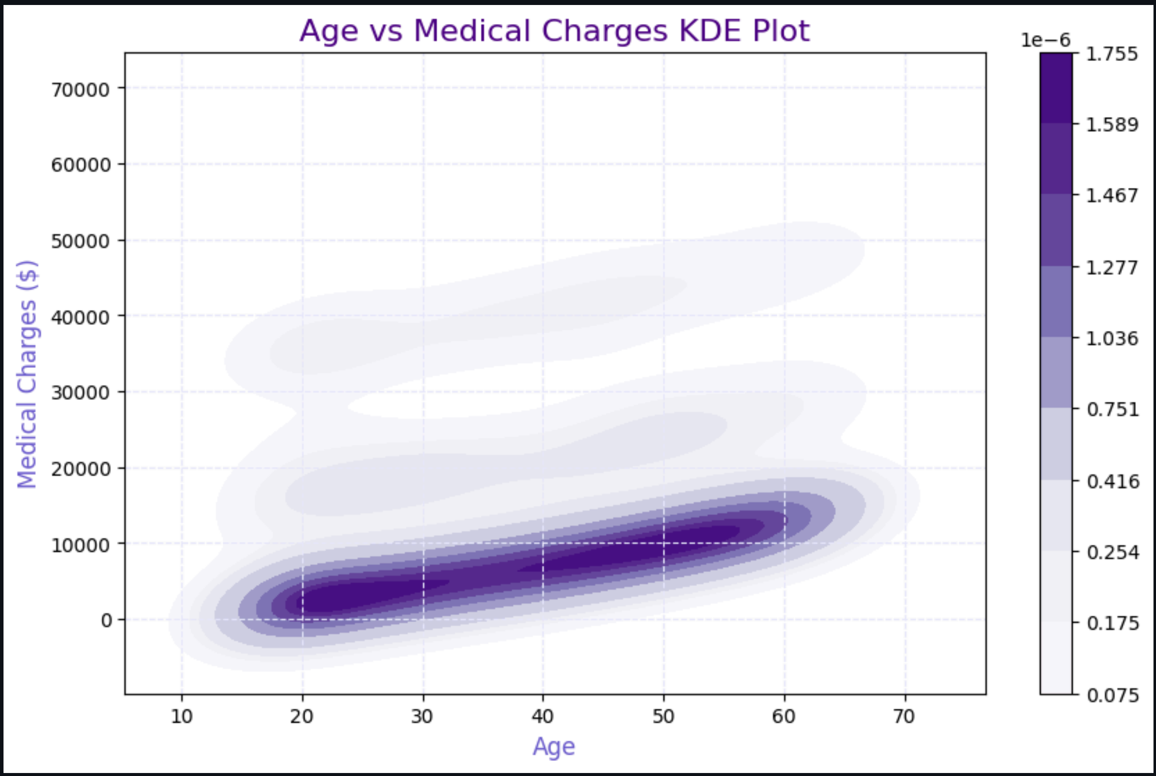
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515

max	64.000000	53.130000	5.000000	63770.428010

Relationship Between Age and Medical Expenses

In [5]:

```
plt.figure(figsize=(10, 6))
sns.kdeplot(
    data=df,
    x="age",
    y="charges",
    cmap="Purples",
    shade=True,
    cbar=True
)

plt.title("Age vs Medical Charges KDE Plot", fontsize=16, color='indigo')
plt.xlabel("Age", fontsize=12, color='slateblue')
plt.ylabel("Medical Charges ($)", fontsize=12, color='slateblue')
plt.grid(True, color='lavender', linestyle='--')
plt.show()
```

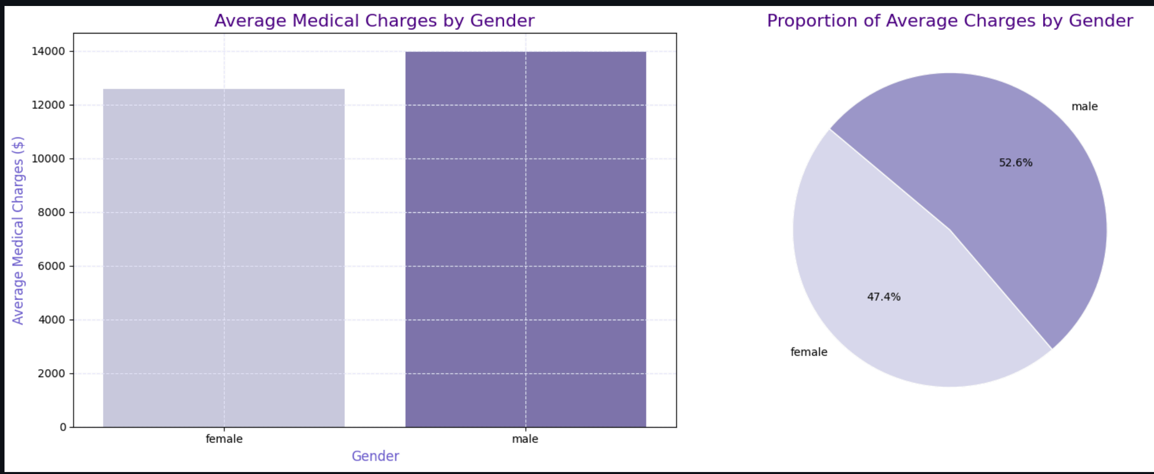


Gender and Average Medical Expenses

In [6]:

```
average_charges_by_sex = df.groupby('sex')['charges'].mean().reset_index()

fig, axes = plt.subplots(1, 2, figsize=(16, 6))
colors = sns.color_palette("Purples", 3)
# Sütun grafiği
sns.barplot(ax=axes[0], data=average_charges_by_sex, x="sex", y="charges", pa
axes[0].set_title("Average Medical Charges by Gender", fontsize=16, color='in
axes[0].set_xlabel("Gender", fontsize=12, color='slateblue')
axes[0].set_ylabel("Average Medical Charges ($)", fontsize=12, color='slatebl
axes[0].grid(True, linestyle='--', color='lavender')


axes[1].pie(
```

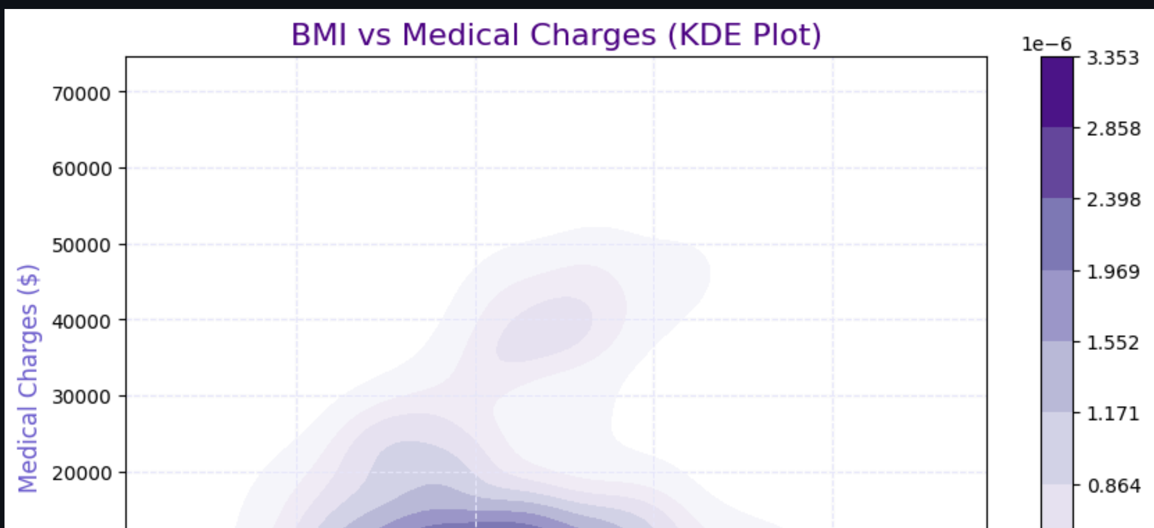
```
axes[1].pie(
    average_charges_by_sex['charges'],
    labels=average_charges_by_sex['sex'],
    autopct='%1.1f%%',
    colors=colors,
    startangle=140,
    wedgeprops={'edgecolor': 'white'}
)
axes[1].set_title("Proportion of Average Charges by Gender", fontsize=16, col

plt.tight_layout()
plt.show()
```

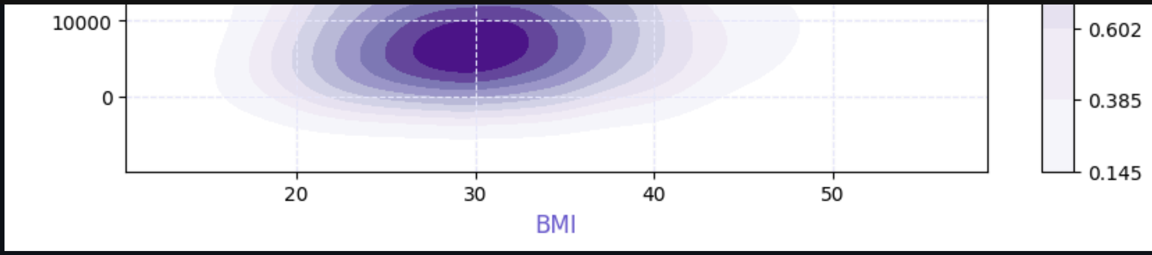


Relationship Between BMI and Medical Expenses

In [7]:

```
plt.figure(figsize=(10, 6))
sns.kdeplot(
    data=df,
    x="bmi",
    y="charges",
    cmap="Purples",
    shade=True,
    cbar=True
)
plt.title("BMI vs Medical Charges (KDE Plot)", fontsize=16, color='indigo')
plt.xlabel("BMI", fontsize=12, color='slateblue')
plt.ylabel("Medical Charges ($)", fontsize=12, color='slateblue')
plt.grid(True, color='lavender', linestyle='--')
plt.show()
```

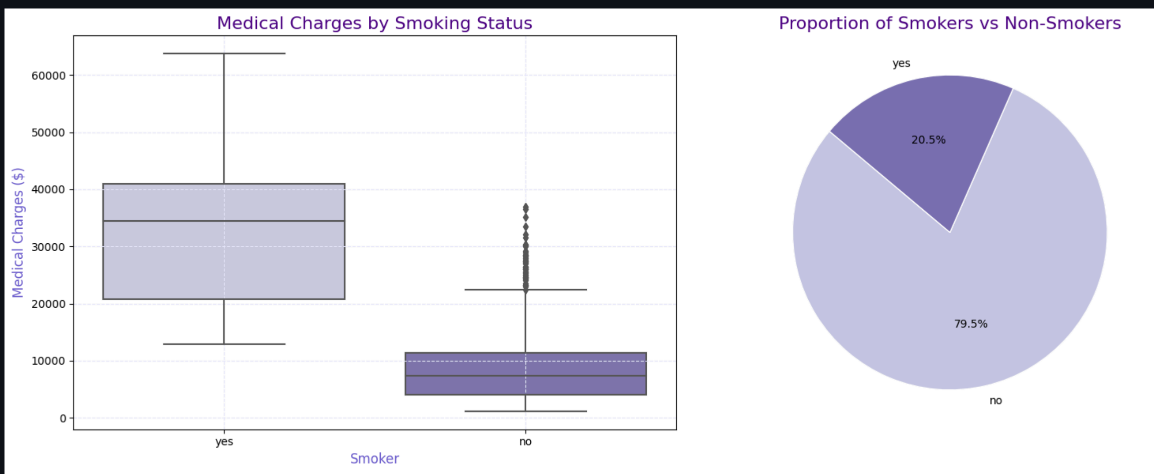




Comparison of Smokers and Non-Smokers

In [8]:

```
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
colors = sns.color_palette("Purples", 2)

sns.boxplot(ax=axes[0], data=df, x="smoker", y="charges", palette="Purples")
axes[0].set_title("Medical Charges by Smoking Status", fontsize=16, color='in
axes[0].set_xlabel("Smoker", fontsize=12, color='slateblue')
axes[0].set_ylabel("Medical Charges ($)", fontsize=12, color='slateblue')
axes[0].grid(True, linestyle='--', color='lavender')

smoker_counts = df['smoker'].value_counts()
axes[1].pie(
    smoker_counts,
    labels=smoker_counts.index,
    autopct='%1.1f%%',
    colors=colors,
    startangle=140,
    wedgeprops={'edgecolor': 'white'}
)
axes[1].set_title("Proportion of Smokers vs Non-Smokers", fontsize=16, color=

plt.tight_layout()
plt.show()
```

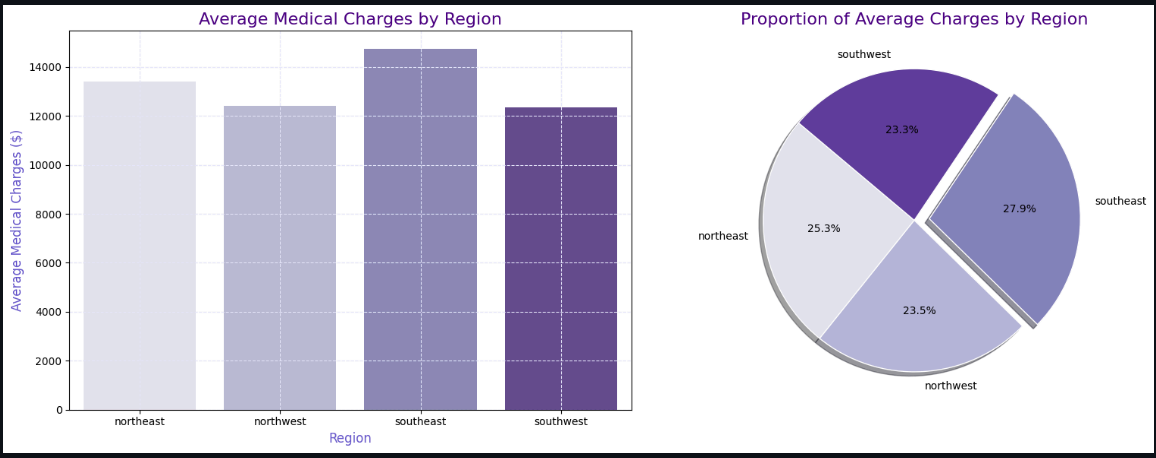


Cost Analysis by Region

In [9]:

```
average_charges_by_region = df.groupby('region')['charges'].mean().reset_inde

fig, axes = plt.subplots(1, 2, figsize=(16, 6))
colors = sns.color_palette("Purples", len(average_charges_by_region))


sns.barplot(ax=axes[0], data=average_charges_by_region, x="region", y="charge
axes[0].set_title("Average Medical Charges by Region", fontsize=16, color='in
```

```
axes[0].set_xlabel("Region", fontsize=12, color='slateblue')
axes[0].set_ylabel("Average Medical Charges ($)", fontsize=12, color='slatebl
axes[0].grid(True, linestyle='--', color='lavender')

charges = average_charges_by_region['charges']
regions = average_charges_by_region['region']
explode = [0.1 if i == charges.idxmax() else 0 for i in range(len(charges))]
axes[1].pie(
    charges,
    labels=regions,
    autopct='%1.1f%%',
    colors=colors,
    explode=explode,
    shadow=True,
    startangle=140,
    wedgeprops={'edgecolor': 'white'}
)
axes[1].set_title("Proportion of Average Charges by Region", fontsize=16, col

plt.tight_layout()
plt.show()
```

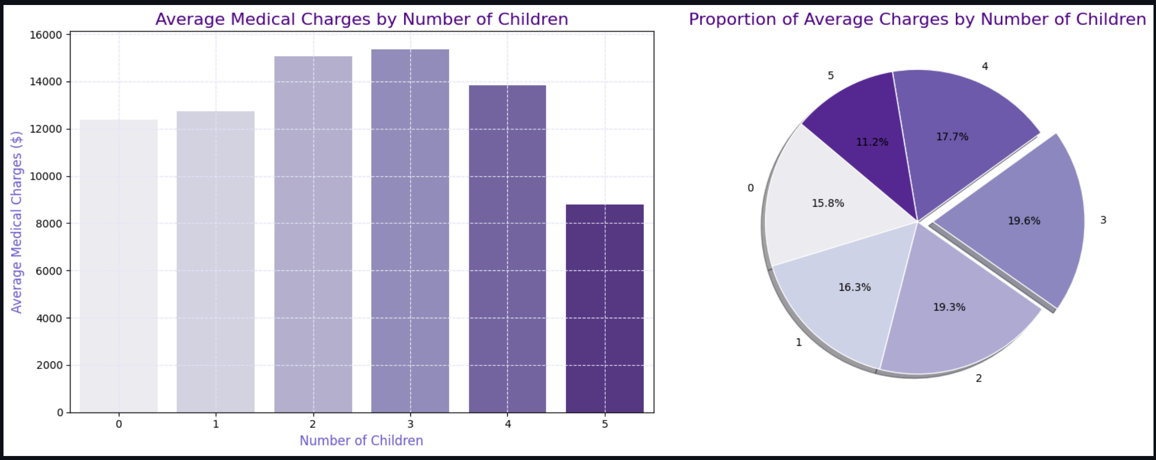


Relationship Between Number of Children and Expenses

In [10]:

```
average_charges_by_children = df.groupby('children')['charges'].mean().reset_

fig, axes = plt.subplots(1, 2, figsize=(16, 6))
colors = sns.color_palette("Purples", len(average_charges_by_children))


sns.barplot(ax=axes[0], data=average_charges_by_children, x="children", y="ch
axes[0].set_title("Average Medical Charges by Number of Children", fontsize=1
axes[0].set_xlabel("Number of Children", fontsize=12, color='slateblue')
axes[0].set_ylabel("Average Medical Charges ($)", fontsize=12, color='slatebl
axes[0].grid(True, linestyle='--', color='lavender')

charges = average_charges_by_children['charges']
children = average_charges_by_children['children']
explode = [0.1 if i == charges.idxmax() else 0 for i in range(len(charges))]
axes[1].pie(
    charges,
    labels=children,
    autopct='%1.1f%%',
    colors=colors,
    explode=explode,
```

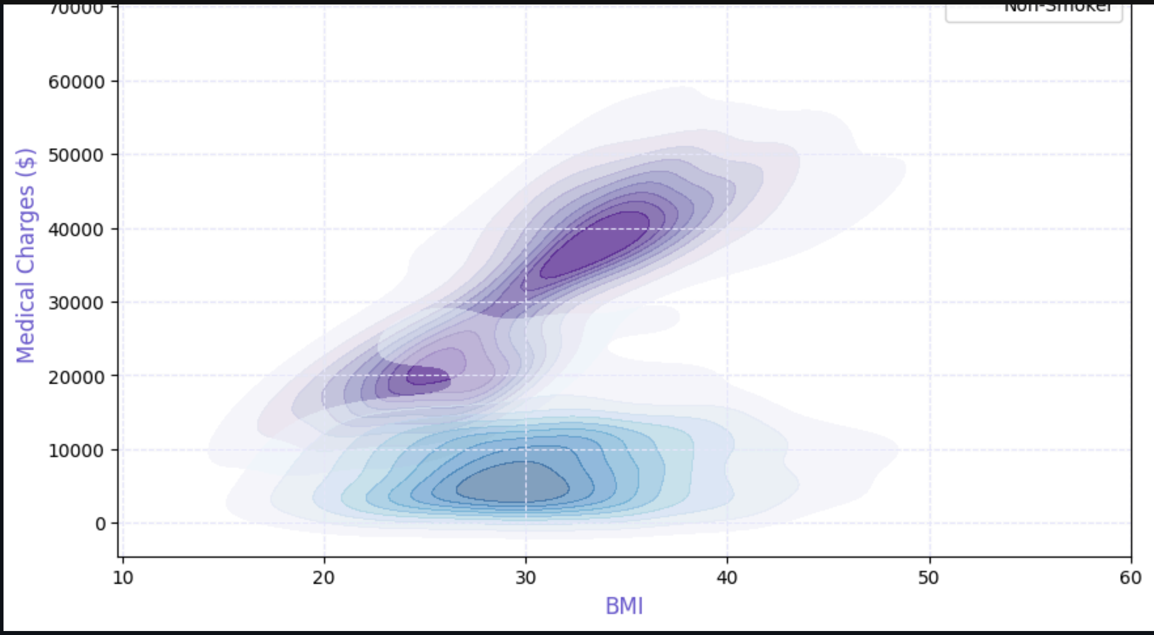
```
    shadow=True,
    startangle=140,
    wedgeprops={'edgecolor': 'white'}
)
axes[1].set_title("Proportion of Average Charges by Number of Children", font

plt.tight_layout()
plt.show()
```



Combined Effect of Smoking, BMI and Expenses

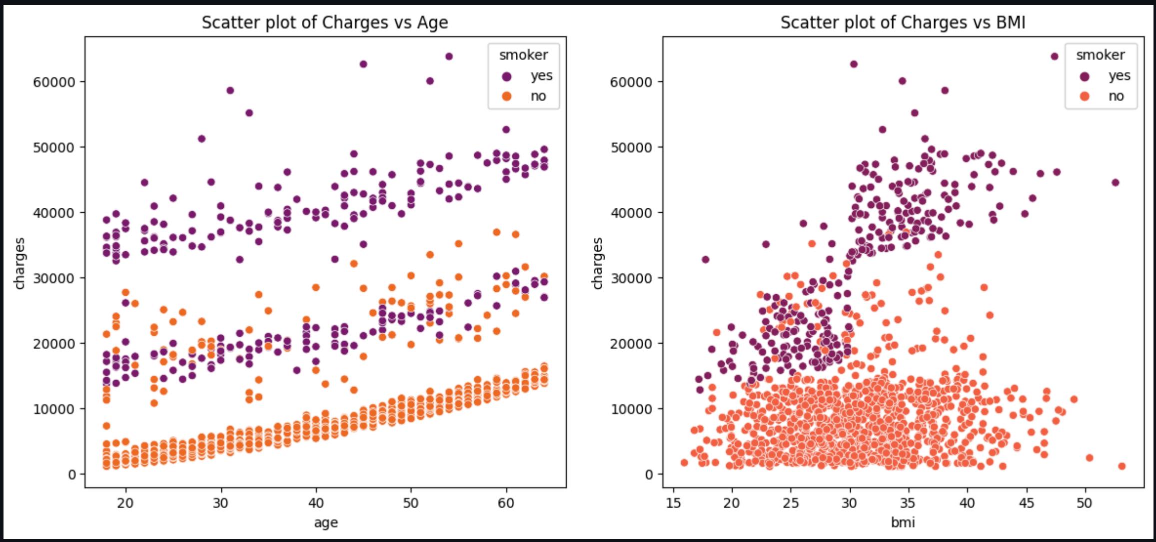
In [11]:

```
plt.figure(figsize=(10, 6))

sns.kdeplot(
    data=df[df['smoker'] == 'yes'],
    x="bmi",
    y="charges",
    cmap="Purples",
    shade=True,
    alpha=0.7,
    label="Smoker",
    linewidth=2
)

sns.kdeplot(
    data=df[df['smoker'] == 'no'],
    x="bmi",
    y="charges",
    cmap="Blues",
    shade=True,
    alpha=0.5,
    label="Non-Smoker",
    linewidth=2
)

plt.title("BMI vs Medical Charges by Smoking Status (KDE Plot)", fontsize=16,
plt.xlabel("BMI", fontsize=12, color='slateblue')
plt.ylabel("Medical Charges ($)", fontsize=12, color='slateblue')
plt.grid(True, linestyle='--', color='lavender')
plt.legend()
plt.show()
```



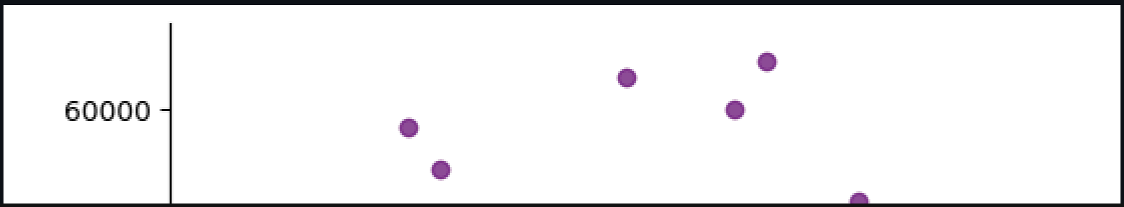
In [12]:

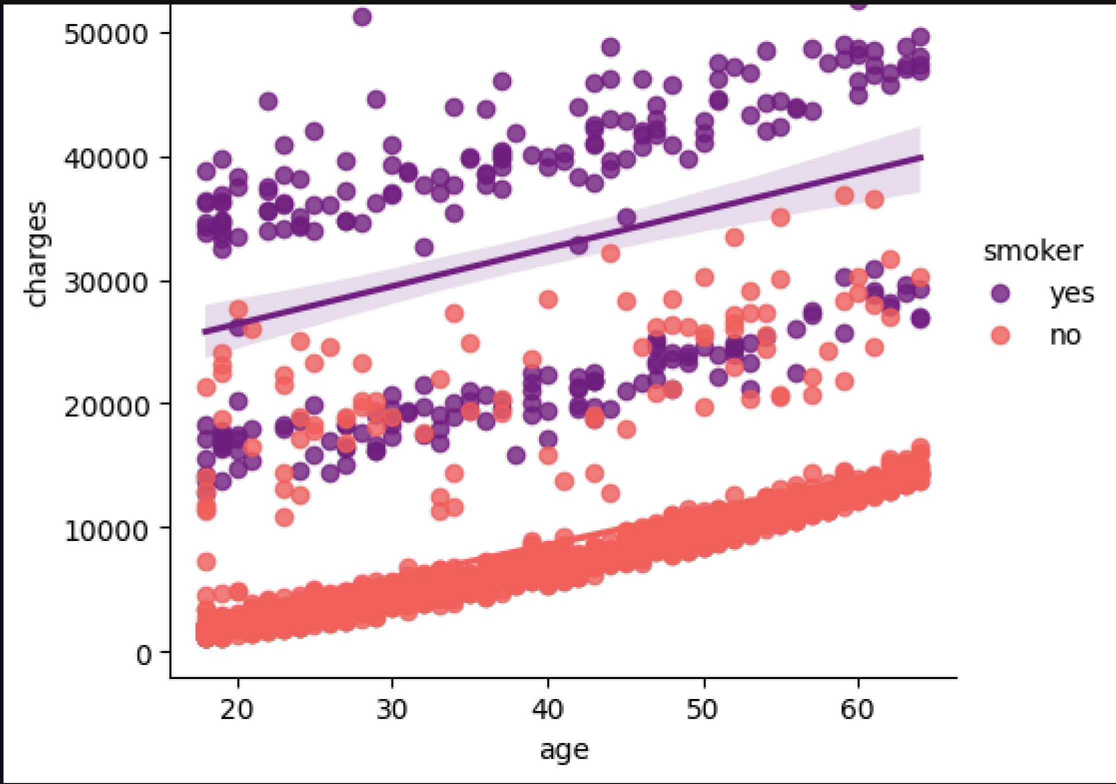
```
f = plt.figure(figsize=(14,6))

ax = f.add_subplot(121)
sns.scatterplot(x='age', y='charges', data=df, palette='inferno', hue='smoker
ax.set_title('Scatter plot of Charges vs Age')

ax = f.add_subplot(122)
sns.scatterplot(x='bmi', y='charges', data=df, palette='rocket', hue='smoker'
ax.set_title('Scatter plot of Charges vs BMI')

plt.savefig('sc.png')
plt.show()
```



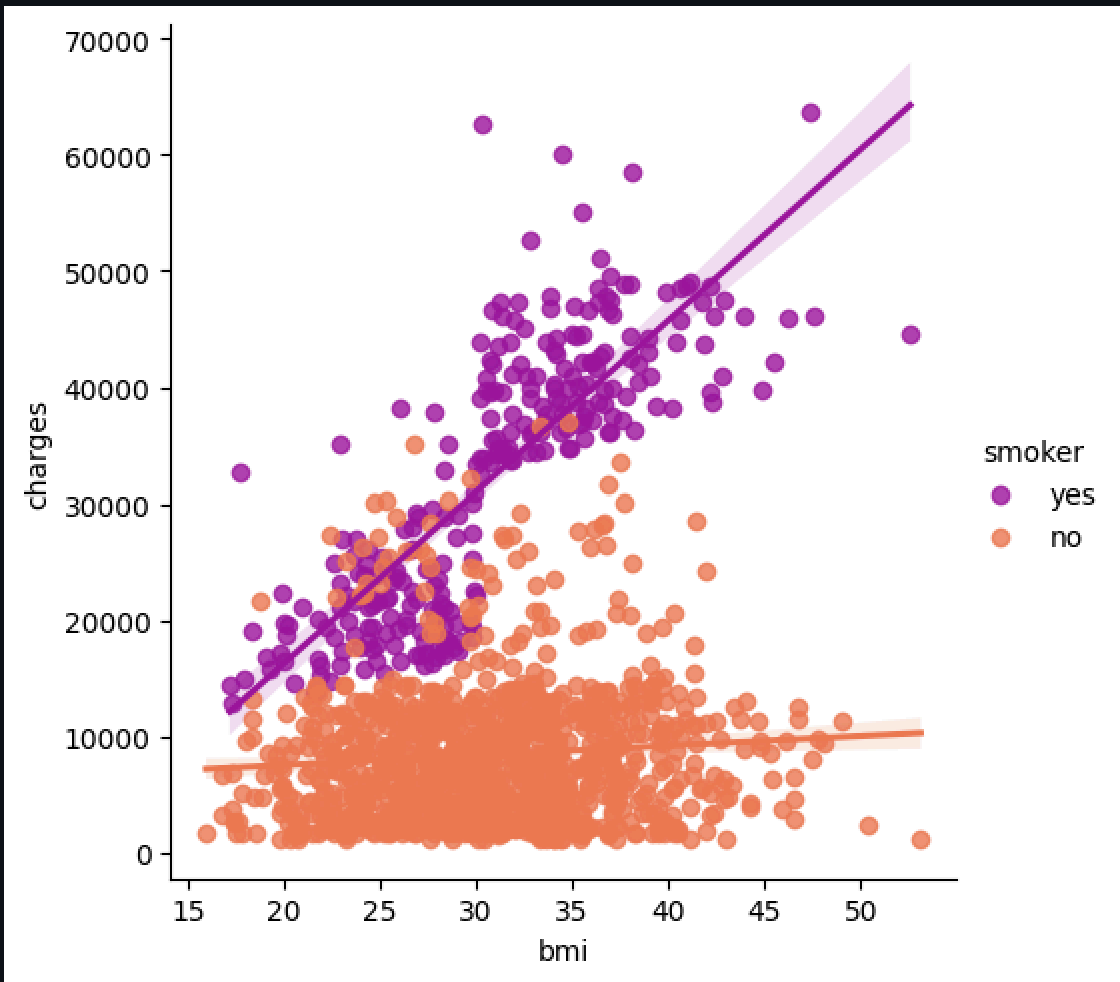
In [13]:

```
ax = sns.lmplot(x='age', y='charges', data=df, hue='smoker', palette='magma')
ax = sns.lmplot(x='bmi', y='charges', data=df, hue='smoker', palette='plasma'
ax = sns.lmplot(x='children', y='charges', data=df, hue='smoker', palette='in
```

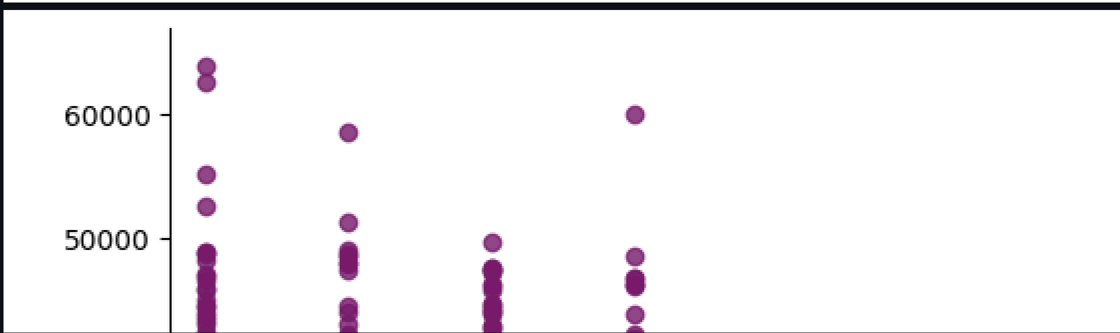


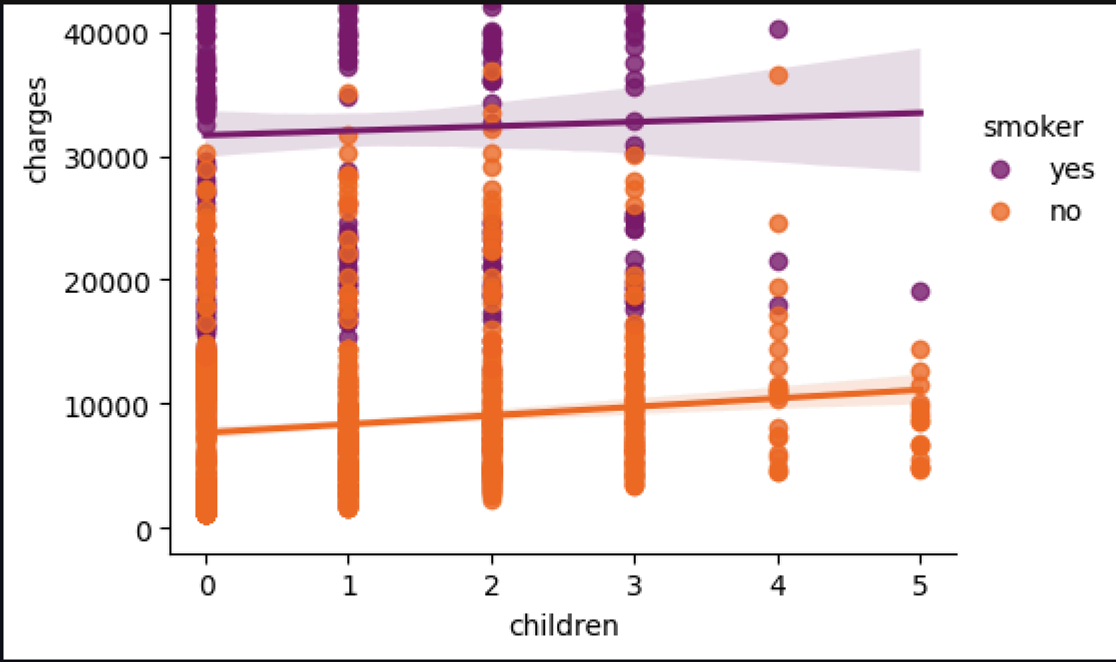


charges
age
smoker
yes
no



charges
bmi
smoker
yes
no





Data Preprocessing - Encoding

1. Label Encoding: Converts categorical values into unique integers.
2. One-Hot Encoding: Creates binary columns for each category.
3. Dummy Variable Trap: Avoid redundancy by dropping one column in One-Hot Encoding to prevent multicollinearity.

In [14]:

```
# Dummy variable
categorical_columns = ['sex','children', 'smoker', 'region']
df_encode = pd.get_dummies(data = df, prefix = 'OHE', prefix_sep='_',
               columns = categorical_columns,
               drop_first =True,
              dtype='int8')
```

In [15]:

```
# Lets verify the dummay variable process
print('Columns in original data frame:\n',df.columns.values)
print('\nNumber of rows and columns in the dataset:',df.shape)
print('\nColumns in data frame after encoding dummy variable:\n',df_encode.co
print('\nNumber of rows and columns in the dataset:',df_encode.shape)
```

```
Columns in original data frame:
 ['age' 'sex' 'bmi' 'children' 'smoker' 'region' 'charges']

Number of rows and columns in the dataset: (1338, 7)

Columns in data frame after encoding dummy variable:
 ['age' 'bmi' 'charges' 'OHE_male' 'OHE_1' 'OHE_2' 'OHE_3' 'OHE_4' 'OHE_5'
 'OHE_yes' 'OHE_northwest' 'OHE_southeast' 'OHE_southwest']

Number of rows and columns in the dataset: (1338, 13)
```

In [16]:

```
from scipy.stats import boxcox
y_bc,lam, ci= boxcox(df_encode['charges'],alpha=0.05)

#df['charges'] = y_bc
```

```
# it did not perform better for this model, so log transform is used
ci,lam
```

Out[16]: ((-0.011402906172966682, 0.09880968597671798), 0.043649061187374535)

In [17]:
```
## Log transform
df_encode['charges'] = np.log(df_encode['charges'])
```

Train Test split

In [18]:
```
from sklearn.model_selection import train_test_split
X = df_encode.drop('charges',axis=1) # Independet variable
y = df_encode['charges'] # dependent variable

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_
```

Model building

In [19]:
```
# Step 1: add x0 =1 to dataset
X_train_0 = np.c_[np.ones((X_train.shape[0],1)),X_train]
X_test_0 = np.c_[np.ones((X_test.shape[0],1)),X_test]

# Step2: build model
theta = np.matmul(np.linalg.inv( np.matmul(X_train_0.T,X_train_0) ), np.matmu
```

In [20]:
```
# The parameters for linear regression model
parameter = ['theta_'+str(i) for i in range(X_train_0.shape[1])]
columns = ['intersect:x_0=1'] + list(X.columns.values)
parameter_df = pd.DataFrame({'Parameter':parameter,'Columns':columns,'theta':
```

In [21]:
```
# Scikit Learn module
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train,y_train) # Note: x_0 =1 is no need to add, sklearn will t

#Parameter
sk_theta = [lin_reg.intercept_]+list(lin_reg.coef_)
parameter_df = parameter_df.join(pd.Series(sk_theta, name='Sklearn_theta'))
parameter_df
```

Out[21]:

	Parameter	Columns	theta	Sklearn_theta
0	theta_0	intersect:x_0=1	7.017931	7.017931
1	theta_1	age	0.033290	0.033290
2	theta_2	bmi	0.014642	0.014642

3	theta_3	OHE_male	-0.040832	-0.040832
4	theta_4	OHE_1	0.100594	0.100594
5	theta_5	OHE_2	0.260231	0.260231
6	theta_6	OHE_3	0.248347	0.248347
7	theta_7	OHE_4	0.504890	0.504890
8	theta_8	OHE_5	0.409276	0.409276
9	theta_9	OHE_yes	1.527625	1.527625
10	theta_10	OHE_northwest	-0.043022	-0.043022
11	theta_11	OHE_southeast	-0.130996	-0.130996
12	theta_12	OHE_southwest	-0.150006	-0.150006

In [22]:

```
# Normal equation
y_pred_norm =  np.matmul(X_test_0,theta)

#Evaluvation: MSE
J_mse = np.sum((y_pred_norm - y_test)**2)/ X_test_0.shape[0]

# R_square
sse = np.sum((y_pred_norm - y_test)**2)
sst = np.sum((y_test - y_test.mean())**2)
R_square = 1 - (sse/sst)
print('The Mean Square Error(MSE) or J(theta) is: ',J_mse)
print('R square obtain for normal equation method is :',R_square)
```

```
The Mean Square Error(MSE) or J(theta) is:  0.17136541675057662
R square obtain for normal equation method is : 0.79346959557574
```

In [23]:

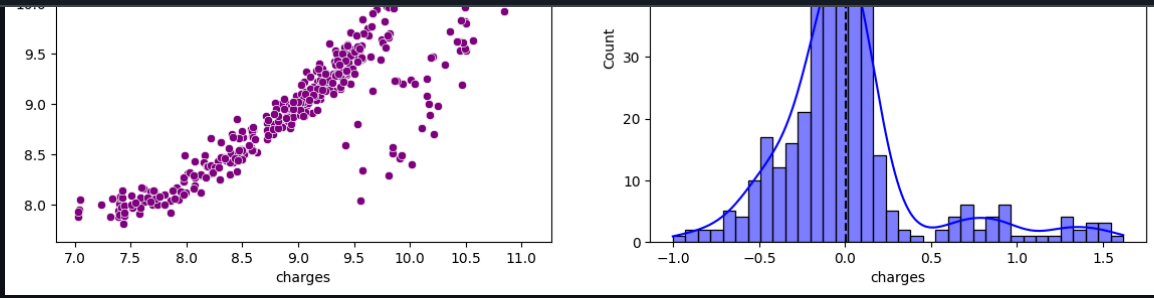
```
# sklearn regression module
y_pred_sk = lin_reg.predict(X_test)

#Evaluvation: MSE
from sklearn.metrics import mean_squared_error
J_mse_sk = mean_squared_error(y_pred_sk, y_test)

# R_square
R_square_sk = lin_reg.score(X_test,y_test)
print('The Mean Square Error(MSE) or J(theta) is: ',J_mse_sk)
print('R square obtain for scikit learn library is :',R_square_sk)
```

```
The Mean Square Error(MSE) or J(theta) is:  0.17136541675057593
R square obtain for scikit learn library is : 0.7934695955757408
```

In [24]:

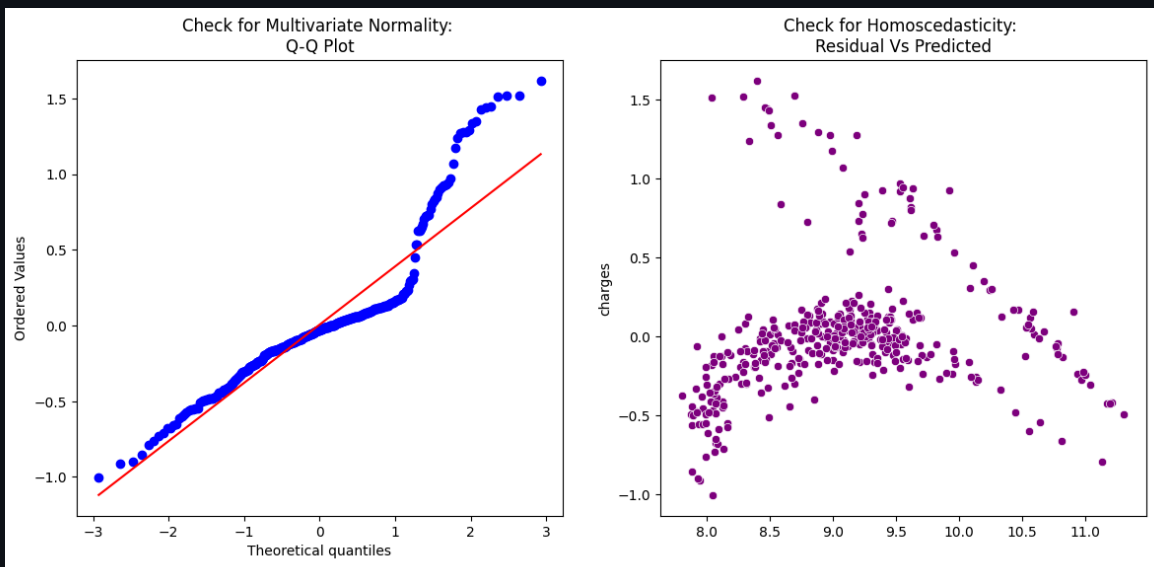
```
f = plt.figure(figsize=(14,5))

# First plot: Check for Linearity
ax = f.add_subplot(121)
sns.scatterplot(x=y_test, y=y_pred_sk, ax=ax, color='purple')
ax.set_title('Check for Linearity:\n Actual Vs Predicted value')

# Second plot: Check for Residual normality & mean
ax = f.add_subplot(122)
sns.histplot((y_test - y_pred_sk), ax=ax, color='b', kde=True)
ax.axvline((y_test - y_pred_sk).mean(), color='k', linestyle='--')
ax.set_title('Check for Residual normality & mean: \n Residual error')
```

```python
ax.set_title('Check for Residual normality & mean: \n Residual error')

plt.show()
```

In [25]:

```python
import scipy as sp

# Check for Multivariate Normality and Homoscedasticity
f, ax = plt.subplots(1, 2, figsize=(14, 6))

# Q-Q Plot
_, (_, _, r) = sp.stats.probplot((y_test - y_pred_sk), fit=True, plot=ax[0])
ax[0].set_title('Check for Multivariate Normality: \nQ-Q Plot')

# Scatterplot for Homoscedasticity
sns.scatterplot(y=(y_test - y_pred_sk), x=y_pred_sk, ax=ax[1], color='purple'
ax[1].set_title('Check for Homoscedasticity: \nResidual Vs Predicted')

plt.show()
```



In [26]:

```python
# Check for Multicollinearity
#Variance Inflation Factor
VIF = 1/(1- R_square_sk)
VIF
```

Out[26]: 4.841902105347058

Linear Regression Model Assumptions:

- **Linearity:**

  The assumption of linearity: The relationship between the actual and predicted values should be linear according to the assumptions of linear regression. However, the Actual vs Predicted plot shows a curve, indicating that the linearity assumption does not hold. This suggests that there is no linear relationship in this model, and a more complex relationship may exist.

- **Residual Normality:**
  Normality of residuals: It is assumed that the residual errors (the difference between observed and predicted values) are normally distributed. The residual error plot is right-skewed, and the residual mean is zero. While the mean of the residuals is zero (which is expected), the right skew indicates that the model fails to fully capture the asymmetric distribution of the data. This could mean that the model is not correctly accounting for some of the data's variability, especially at higher values.

- **Multivariate Normality (Q-Q Plot):**
  Multivariate normality assumption: The Q-Q plot checks the normality of the residuals. It shows that values with a log value greater than 1.5 tend to increase. This indicates a deviation from normality and suggests that extreme values (outliers) are affecting the model, violating the assumption of normal distribution for the residuals.

- **Homoscedasticity:**
  Homoscedasticity (constant variance of residuals): The Residual vs Predicted plot exhibits heteroscedasticity, meaning that the variance of the residuals is not constant. The residuals increase in magnitude as the predicted values increase, suggesting that the model makes larger errors at higher values. This violates the assumption that the variance of the residuals should remain constant across all levels of the independent variable.

- **Multicollinearity:**
  Multicollinearity: The Variance Inflation Factor (VIF) value is less than 5, indicating that there is no significant multicollinearity in the model. This suggests that the independent variables are not highly correlated with each other and each variable has an independent effect on the outcome.

Analysis and Results

- **1. Relationship between Age and Medical Expenses;**
  We examined the relationship between age and medical expenses and visualized it with a scatter plot. 📊

- **2. Gender and Average Medical Expenses;**