Medical Health Insurance

https://www.kaggle.com/datasets/mirichoi0218/insurance?select=insurance.csv

This dataset is downloaded from Kaggle and is used by many textbooks as standard datasets to perform Predictive Analytics

Importing Libraries

In [1]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
import sys
import numpy as np
```

Load the Dataset

In [2]:
```
insurance_data=pd.read_csv('/home/utk.tennessee.edu/nnaraya2/SanDisk/Endicot
print('The shape of the Insurance Data is ', insurance_data.shape)
```

```
The shape of the Insurance Data is  (1338, 7)
```

In [3]:
```
insurance_data.head(15)
```

Out[3]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960
7	37	female	27.740	3	no	northwest	7281.50560
8	37	male	29.830	2	no	northeast	6406.41070
9	60	female	25.840	0	no	northwest	28923.13692
10	25	male	26.220	0	no	northeast	2721.32080
11	62	female	26.290	0	yes	southeast	27808.72510
12	23	male	34.400	0	no	southwest	1826.84300
13	56	female	39.820	0	no	southeast	11090.71780
14	27	male	42.130	0	yes	southeast	39611.75770

Basic Terminology

Depending on the type of field you study in, rows and columns have different names,

Row=Observation=Records=Entries=Sample=Instance

Column=Variable=Features=Attributes=Fields=Dimensions

For now, we will use "observations" for row and "variables for columns". Therefore, the dataset contains 1338 observations on 7 variables.

Here's an overview of the attributes:

1. **age**: Integer - Age of the Primary Beneficiary
2. **sex**: Categorical (object) - insurance contractor gender, female, male
3. **bmi**: Float - Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9
4. **children**: Integer - Number of children/dependents covered by insurace
5. **smoker**: Categorical (object) - Whether the individual is a smoker (yes/no).
6. **region**: Categorical (object) - the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

7. **charges**: Float - Individual Medical cost charged by the insurer

Descriptive Analytics

Fact: Per person health insurance premium in Massachusetts is $8068

As employess of insurance company,if you had access to the historical data, what would you like to know?

In [4]:
```
1.
2.
3.
4.
```

Out[4]:
```
4.0
```

quantitative_variables=['age', 'bmi', 'children', 'charges']

qualitative_varaiables=['sex', 'children', 'smoker', 'region']

Notice: 'children' variable appears in both qualitative and quantitative both type of variable

News headlines

"As per the latest data, the average insurance premium per person for Massachussets is approximately $8800"

In [5]:
```
print('the average insurance premium paid according to the data is ', insura
```

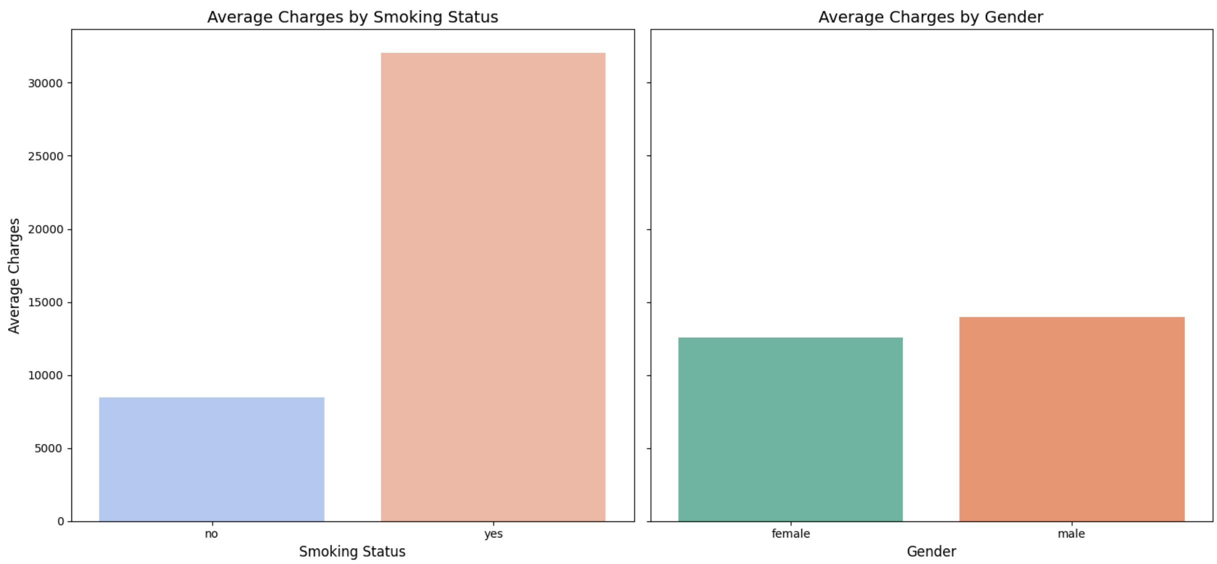
```
the average insurance premium paid according to the data is  13270.422265141
257
```

Bar Plot

"Average Charges vs Gender" & "Average Charges vs Smoking"

In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Group data for descriptive statistics
avg_charges_by_smoker = insurance_data.groupby('smoker')['charges'].mean().r
avg_charges_by_gender = insurance_data.groupby('sex')['charges'].mean().rese
```

```
# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(15, 7), sharey=True)

# Subplot 1: Average charges by smoker status
sns.barplot(data=avg_charges_by_smoker, x='smoker',hue='smoker', y='charges'
axes[0].set_title('Average Charges by Smoking Status', fontsize=14)
axes[0].set_xlabel('Smoking Status', fontsize=12)
axes[0].set_ylabel('Average Charges', fontsize=12)

# Subplot 2: Average charges by gender
sns.barplot(data=avg_charges_by_gender, x='sex',hue='sex', y='charges', pale
axes[1].set_title('Average Charges by Gender', fontsize=14)
axes[1].set_xlabel('Gender', fontsize=12)
axes[1].set_ylabel('')

# Adjust layout
plt.tight_layout()
plt.show()
```



Does smoking affect insurance charges?

Does gender affect insurance charges?

In [ ]:

```

```

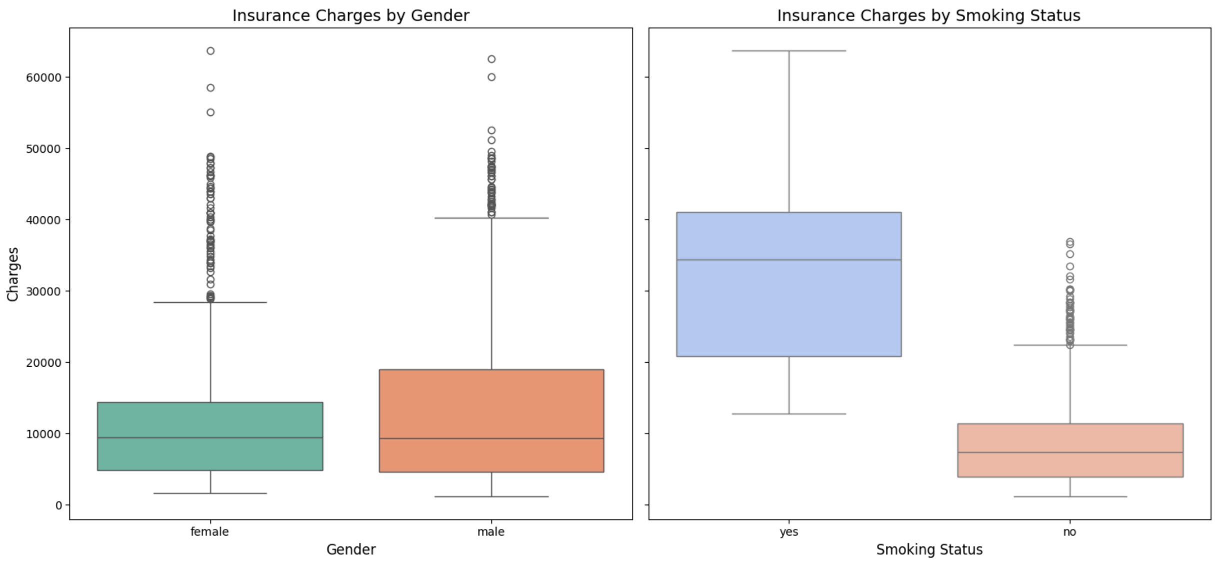
Box Plots

"Average Charges vs Gender" & "Average Charges vs Smoking"

In [7]:

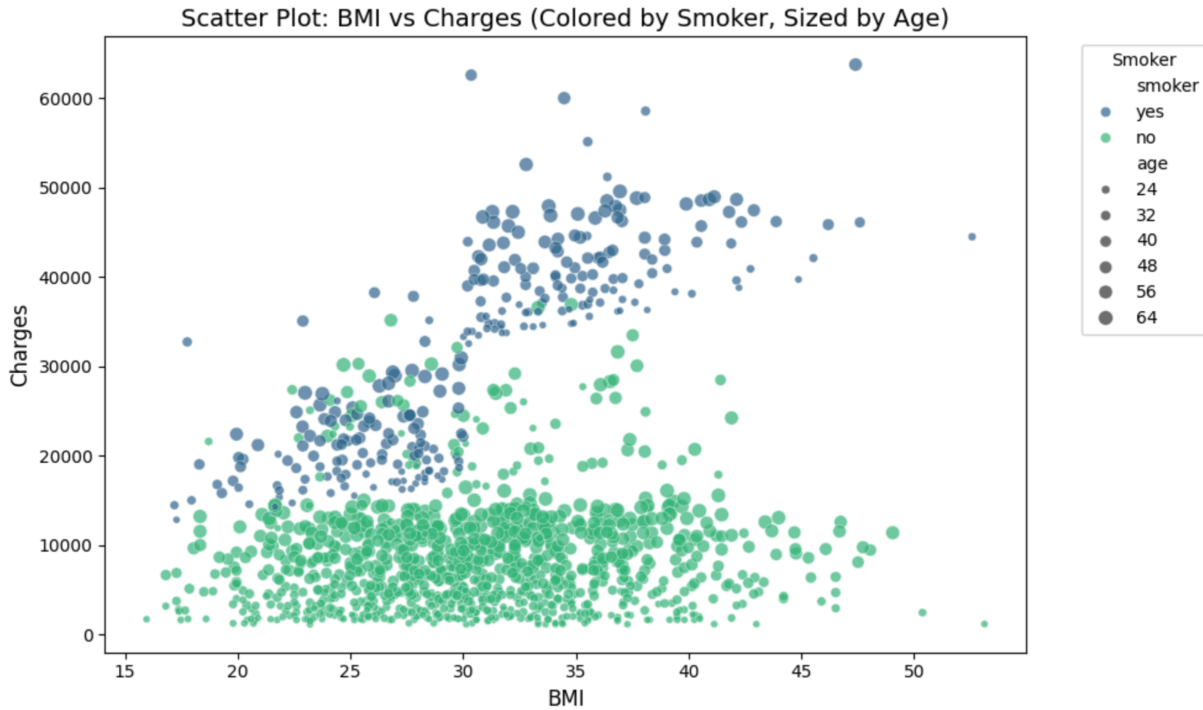
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(15, 7), sharey=True)
```

```
# Subplot 1: Gender vs. Insurance Charges
sns.boxplot(data=insurance_data, x='sex',hue='sex', y='charges', palette='Se
axes[0].set_title("Insurance Charges by Gender", fontsize=14)
axes[0].set_xlabel("Gender", fontsize=12)
axes[0].set_ylabel("Charges", fontsize=12)

# Subplot 2: Smoking vs. Insurance Charges
sns.boxplot(data=insurance_data, x='smoker',hue='smoker', y='charges', palet
axes[1].set_title("Insurance Charges by Smoking Status", fontsize=14)
axes[1].set_xlabel("Smoking Status", fontsize=12)
axes[1].set_ylabel("")  # Remove duplicate y-axis label

# Adjust layout
plt.tight_layout()
plt.show()
```

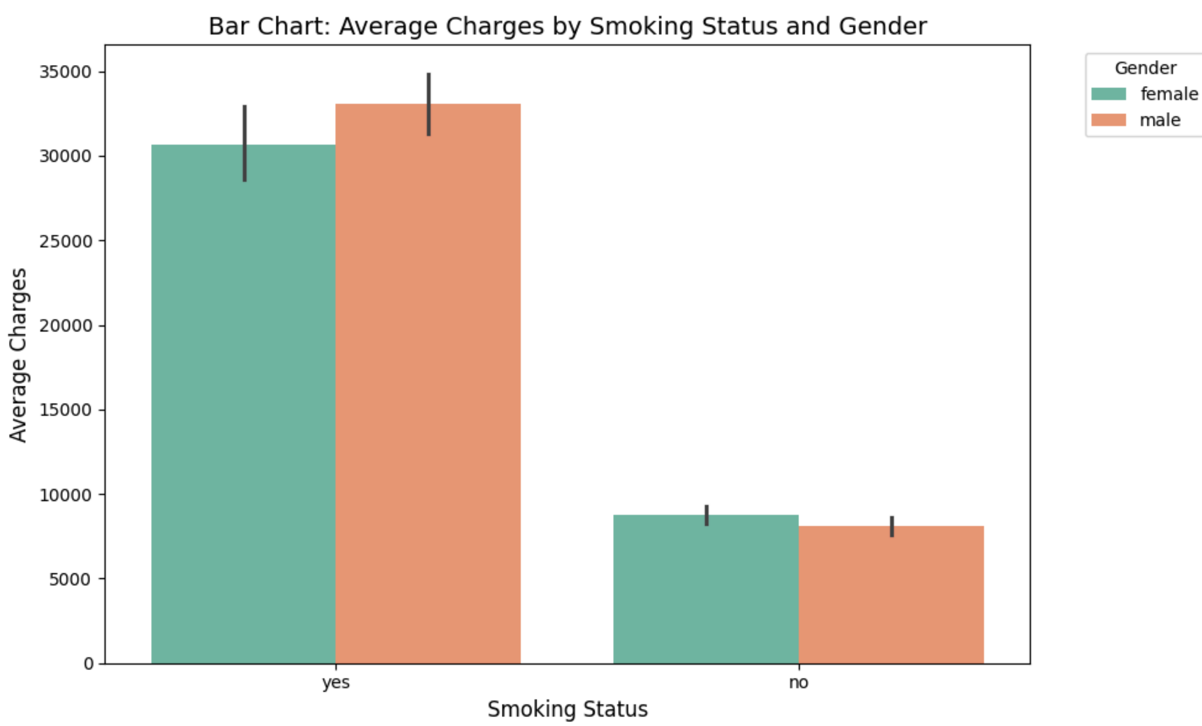


In [8]:

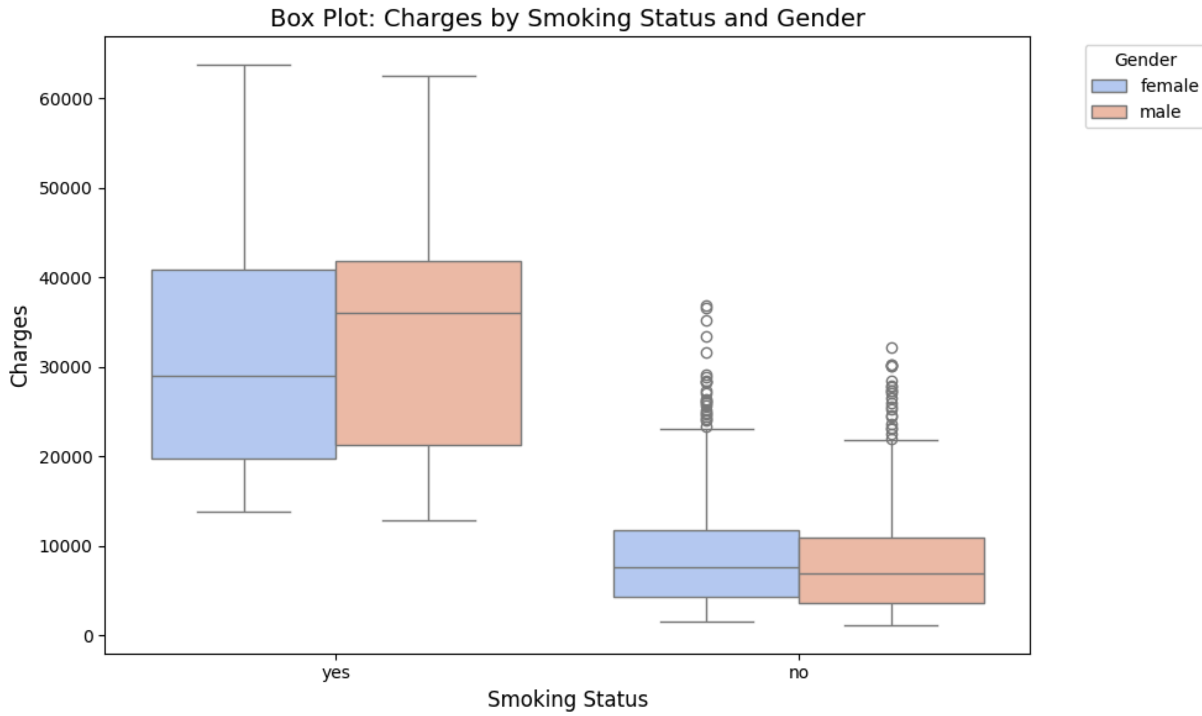
```
# 1. Scatter Plot with Color and Size
plt.figure(figsize=(10, 6))
sns.scatterplot(data=insurance_data, x='bmi', y='charges', hue='smoker', siz
plt.title('Scatter Plot: BMI vs Charges (Colored by Smoker, Sized by Age)',
plt.xlabel('BMI', fontsize=12)
plt.ylabel('Charges', fontsize=12)
plt.legend(title='Smoker', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



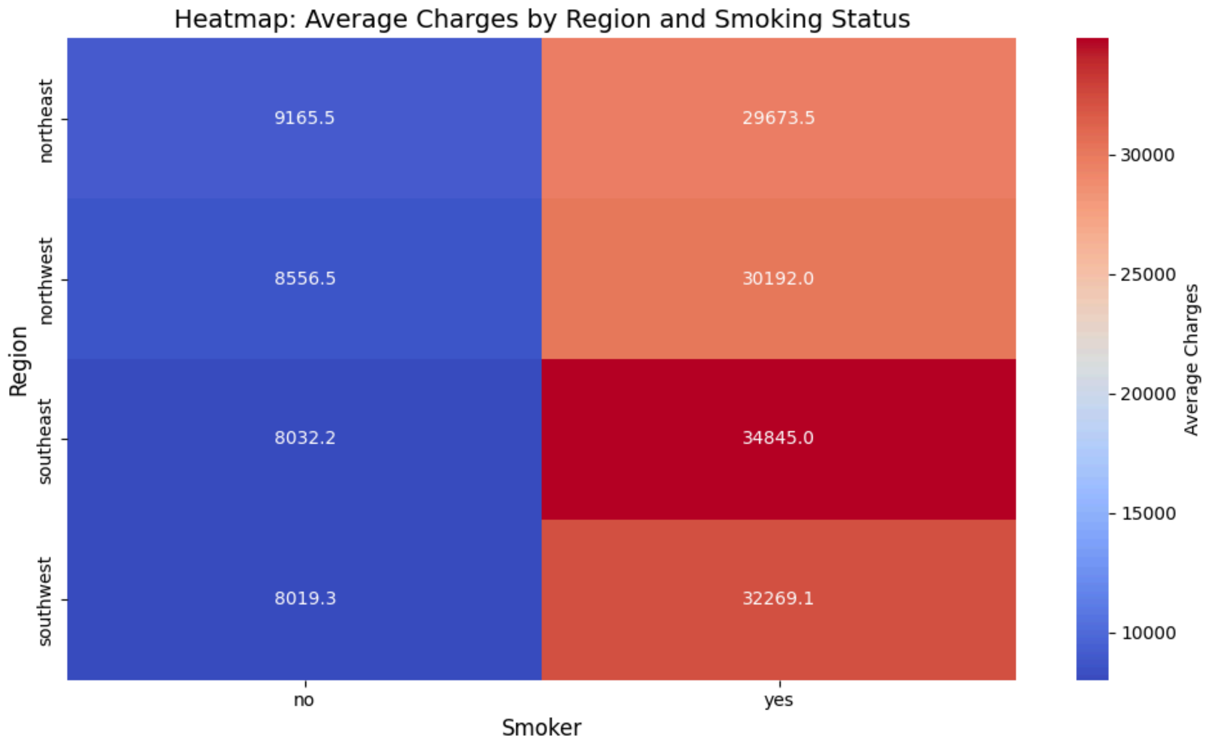
In [9]:
```
# 2. Grouped Bar Chart
plt.figure(figsize=(10, 6))
sns.barplot(data=insurance_data, x='smoker', y='charges', hue='sex', palette
plt.title('Bar Chart: Average Charges by Smoking Status and Gender', fontsiz
plt.xlabel('Smoking Status', fontsize=12)
plt.ylabel('Average Charges', fontsize=12)
plt.legend(title='Gender', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



In [10]:
```
# 3. Box Plot with Hue
plt.figure(figsize=(10, 6))
sns.boxplot(data=insurance_data, x='smoker', y='charges', hue='sex', palette
plt.title('Box Plot: Charges by Smoking Status and Gender', fontsize=14)
plt.xlabel('Smoking Status', fontsize=12)
plt.ylabel('Charges', fontsize=12)
plt.legend(title='Gender', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



In [11]:
```
# 4. Heatmap
plt.figure(figsize=(10, 6))
heatmap_data = insurance_data.pivot_table(values='charges', index='region',
sns.heatmap(heatmap_data, annot=True, fmt='.1f', cmap='coolwarm', cbar_kws={
plt.title('Heatmap: Average Charges by Region and Smoking Status', fontsize=
plt.xlabel('Smoker', fontsize=12)
plt.ylabel('Region', fontsize=12)
plt.tight_layout()
plt.show()
```



Predictive Modeling

Simple Linear Regression

Formula:

Simple linear regression models the relationship between a dependent variable (y) and a single independent variable (X): $$ y = \beta_0 + \beta_1 X + \epsilon $$

Where:

- $$\text{y: Dependent variable (the outcome we want to predict) }$$
- $$\text{X : Independent variable (the predictor) }$$
- $$\beta_0: \text{Intercept (value of y when X = 0)}$$
- $$ \beta_1: \text{Slope (change in y for a one-unit change in X) }$$
- $$ \epsilon: \text{Error term (captures the variation not explained by X) }$$

For the Insurance Dataset:

- X: A single independent variable, such as `bmi` (Body Mass Index).
- y : The dependent variable is `charges` (Medical insurance charges).

Why is it Simple?

- Simple linear regression uses **only one independent variable** X to predict the dependent variable y.
- In this case, we are trying to predict `charges` y using `bmi` X.

Example:

$$ \text{charges} = \beta_0 + \beta_1 \cdot \text{bmi} + \epsilon $$

In the next cell, we will expand this to **multiple linear regression**, which involves using multiple predictors.

Simple Linear Regression: Insurance Charges vs BMI

In [12]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Example: Load your insurance dataset (replace this with actual dataset pat
insurance_data=pd.read_csv('/home/utk.tennessee.edu/nnaraya2/SanDisk/Endicot

# Ensure 'smoker' is encoded numerically
# insurance_data['smoker'] = insurance_data['smoker'].map({'yes': 1, 'no': 0

# Selecting 'bmi' and 'smoker' as predictors and 'charges' as the target var
X = insurance_data[['bmi']]  # Features must be in a DataFrame
y = insurance_data['charges']  # Target variable

# Handle missing values
X = X.dropna()  # Drop rows with missing values in features
y = y[X.index]  # Align target variable with cleaned features

# Ensure all data is numeric
X = X.apply(pd.to_numeric, errors='coerce')
y = pd.to_numeric(y, errors='coerce')

# 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, rar

# Add a constant (intercept) to the training data
X_train_with_const = sm.add_constant(X_train)

# Fit the Multiple Linear Regression model
model = sm.OLS(y_train, X_train_with_const).fit()

# Print the summary of the regression results
print(model.summary())

# Predict charges on the test data
X_test_with_const = sm.add_constant(X_test)  # Add constant to test set
y_pred = model.predict(X_test_with_const)
```

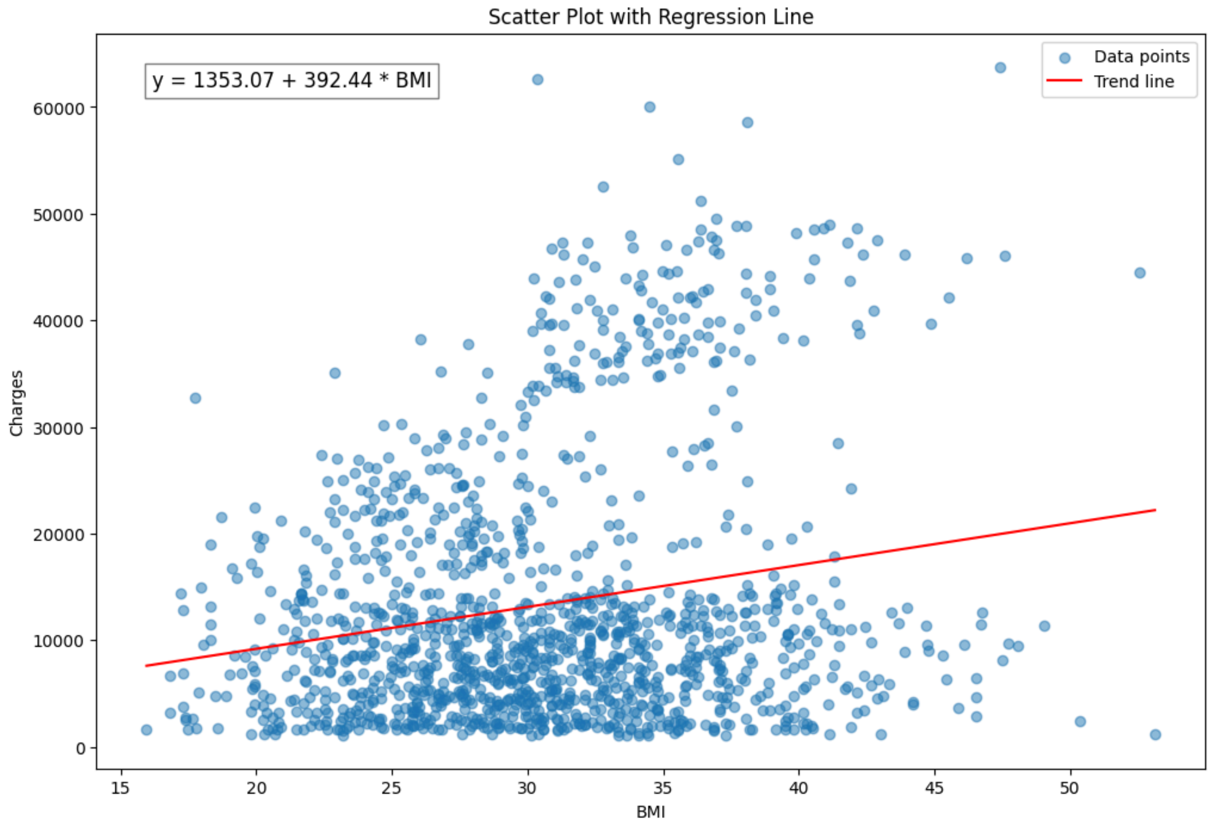
```
# Print model coefficients and intercept
print("\nModel Coefficients:")
print(f"Intercept: {model.params['const']}")
print(f"Slope for BMI: {model.params['bmi']}")

# Interpretation of the results
print("\nInterpretation:")
print(f"For every one-unit increase in BMI, the predicted charges increase b
# print(f"For smokers, the predicted charges increase by approximately {mode
print(f"The intercept {model.params['const']:.2f} represents the predicted c

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nModel Evaluation:")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                charges   R-squared:                       0.0
39
Model:                            OLS   Adj. R-squared:                  0.0
38
Method:                 Least Squares   F-statistic:                     43.
27
Date:                Tue, 10 Dec 2024   Prob (F-statistic):           7.47e-
11
Time:                        06:17:03   Log-Likelihood:                -1154
8.
No. Observations:                1070   AIC:                         2.310e+
04
Df Residuals:                    1068   BIC:                         2.311e+
04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const       1353.0731   1858.569      0.728      0.467   -2293.789    4999.9
35
bmi          392.4365     59.662      6.578      0.000     275.369     509.5
05
==============================================================================
==
Omnibus:                      214.743   Durbin-Watson:                   1.9
30
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              358.0
04
Skew:                           1.308   Prob(JB):                      1.82e-
78
Kurtosis:                       4.087   Cond. No.                         16
1.
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.

Model Coefficients:
Intercept: 1353.0730722046726
Slope for BMI: 392.43654416987937

Interpretation:
For every one-unit increase in BMI, the predicted charges increase by approx
imately 392.44 units.
The intercept 1353.07 represents the predicted charges when BMI is 0 and the
individual is a non-smoker (not realistic in context).
```

```
Model Evaluation:
Mean Squared Error (MSE): 149085057.04
R-squared: 0.04
```

In [13]:

```python
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm

# Assuming the model is already fitted and the dataset is available
# `model` is the trained statsmodels OLS model
# `X` is the DataFrame containing the features, including `bmi`
# `y` is the target variable

# Get the regression coefficients
intercept = model.params['const']
slope = model.params['bmi']

# Create the regression line values
bmi_values = np.linspace(X['bmi'].min(), X['bmi'].max(), 100)  # Generate 10
bmi_values_with_const = sm.add_constant(bmi_values)  # Add constant term
predicted_charges = model.predict(bmi_values_with_const)  # Predict charges

# Plot the scatter plot and regression line
plt.figure(figsize=(12, 8))
plt.scatter(X['bmi'], y, alpha=0.5, label="Data points")  # Scatter plot of
plt.plot(bmi_values, predicted_charges, color='red', label="Trend line")  #

# Add the equation of the line as a text box
equation_text = f"y = {intercept:.2f} + {slope:.2f} * BMI"
plt.text(0.05, 0.95, equation_text, transform=plt.gca().transAxes,
         fontsize=12, verticalalignment='top', bbox=dict(facecolor='white',

# Add titles and labels
plt.title("Scatter Plot with Regression Line")
plt.xlabel("BMI")
plt.ylabel("Charges")
plt.legend()
plt.show()
```



Multiple Linear Regression using BMI and Smoker

In [14]:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Example: Load your insurance dataset (replace this with actual dataset pat
insurance_data=pd.read_csv('/home/utk.tennessee.edu/nnaraya2/SanDisk/Endicot

# Ensure 'smoker' is encoded numerically
insurance_data['smoker'] = insurance_data['smoker'].map({'yes': 1, 'no': 0})

# Selecting 'bmi' and 'smoker' as predictors and 'charges' as the target var
X = insurance_data[['bmi', 'smoker']]  # Features must be in a DataFrame
y = insurance_data['charges']  # Target variable

# Handle missing values
X = X.dropna()  # Drop rows with missing values in features
y = y[X.index]  # Align target variable with cleaned features

# Ensure all data is numeric
X = X.apply(pd.to_numeric, errors='coerce')
y = pd.to_numeric(y, errors='coerce')

# 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, rar

# Add a constant (intercept) to the training data
```

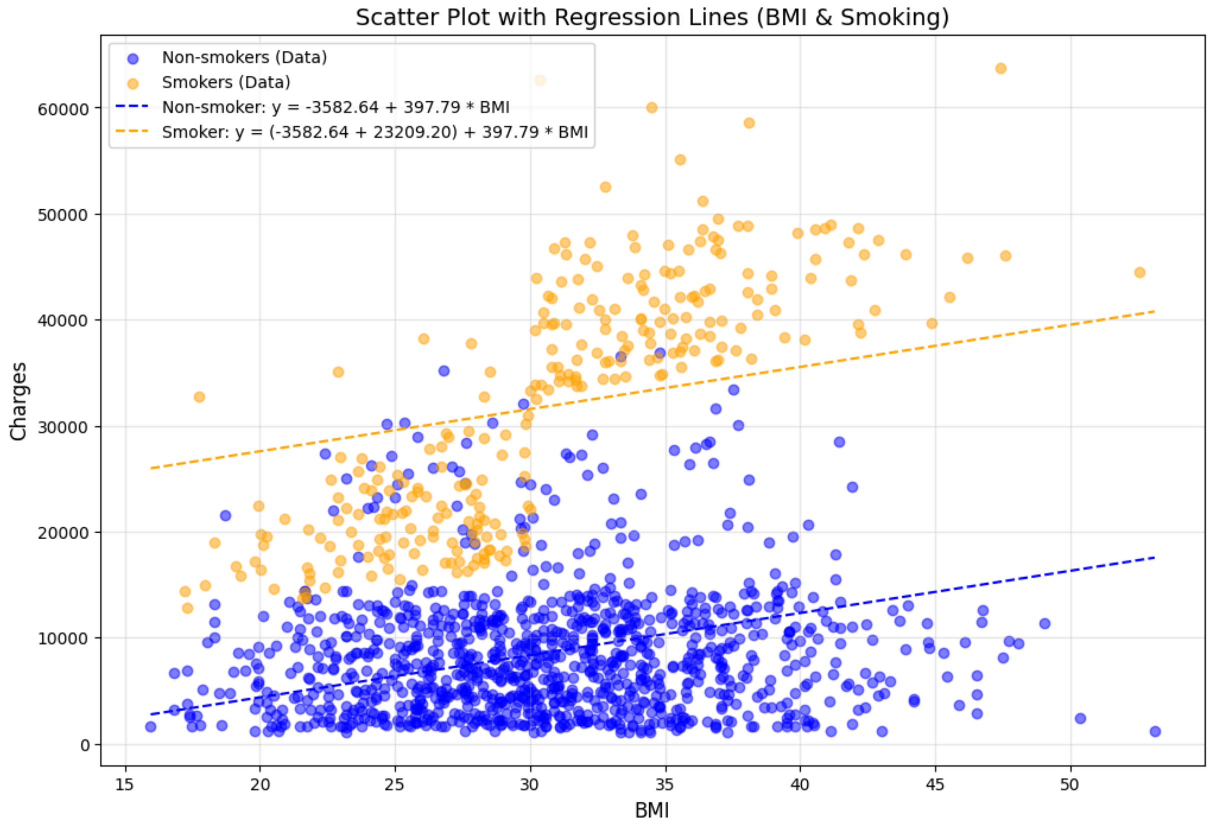
```
X_train_with_const = sm.add_constant(X_train)

# Fit the Multiple Linear Regression model
model = sm.OLS(y_train, X_train_with_const).fit()

# Print the summary of the regression results
print(model.summary())

# Predict charges on the test data
X_test_with_const = sm.add_constant(X_test)  # Add constant to test set
y_pred = model.predict(X_test_with_const)

# Print model coefficients and intercept
print("\nModel Coefficients:")
print(f"Intercept: {model.params['const']}")
print(f"Slope for BMI: {model.params['bmi']}")
print(f"Slope for Smoker: {model.params['smoker']}")

# Interpretation of the results
print("\nInterpretation:")
print(f"For every one-unit increase in BMI, the predicted charges increase b
print(f"For smokers, the predicted charges increase by approximately {model.
print(f"The intercept {model.params['const']:.2f} represents the predicted c

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nModel Evaluation:")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                charges   R-squared:                       0.6
49
Model:                            OLS   Adj. R-squared:                  0.6
48
Method:                 Least Squares   F-statistic:                     98
4.3
Date:                Tue, 10 Dec 2024   Prob (F-statistic):          5.63e-2
43
Time:                        06:19:10   Log-Likelihood:                -1101
0.
No. Observations:                1070   AIC:                         2.203e+
04
Df Residuals:                    1067   BIC:                         2.204e+
04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const      -3582.6389   1130.360     -3.169      0.002   -5800.619   -1364.6
59
bmi          397.7940     36.098     11.020      0.000     326.962     468.6
26
smoker      2.321e+04    539.547     43.016      0.000    2.22e+04    2.43e+
04
==============================================================================
==
Omnibus:                      129.938   Durbin-Watson:                   2.0
14
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              205.7
81
Skew:                           0.828   Prob(JB):                     2.07e-
45
Kurtosis:                       4.369   Cond. No.                         16
2.
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.

Model Coefficients:
Intercept: -3582.6389129022255
Slope for BMI: 397.79403555337575
Slope for Smoker: 23209.19938583341

Interpretation:
For every one-unit increase in BMI, the predicted charges increase by approx
imately 397.79 units.
```

```
For smokers, the predicted charges increase by approximately 23209.20 units
compared to non-smokers.
The intercept -3582.64 represents the predicted charges when BMI is 0 and th
e individual is a non-smoker (not realistic in context).

Model Evaluation:
Mean Squared Error (MSE): 47887940.86
R-squared: 0.69
```

In [15]:

```python
import matplotlib.pyplot as plt
import numpy as np

# Regression coefficients
intercept = model.params['const']
slope_bmi = model.params['bmi']
slope_smoker = model.params['smoker']

# Create BMI range for plotting
bmi_values = np.linspace(X['bmi'].min(), X['bmi'].max(), 100)

# Regression lines for smokers and non-smokers
charges_non_smoker = intercept + slope_bmi * bmi_values  # Non-smokers: smok
charges_smoker = intercept + slope_bmi * bmi_values + slope_smoker  # Smoker

# Plot scatter points
plt.figure(figsize=(12, 8))
plt.scatter(X[X['smoker'] == 0]['bmi'], y[X['smoker'] == 0], alpha=0.5, labe
plt.scatter(X[X['smoker'] == 1]['bmi'], y[X['smoker'] == 1], alpha=0.5, labe

# Plot regression lines
plt.plot(bmi_values, charges_non_smoker, color='blue', label=f"Non-smoker: y
plt.plot(bmi_values, charges_smoker, color='orange', label=f"Smoker: y = ({i

# Add plot details
plt.title("Scatter Plot with Regression Lines (BMI & Smoking)", fontsize=14)
plt.xlabel("BMI", fontsize=12)
plt.ylabel("Charges", fontsize=12)
plt.legend(fontsize=10)
plt.grid(alpha=0.3)
plt.show()
```



Decision Tree

In [ ]:

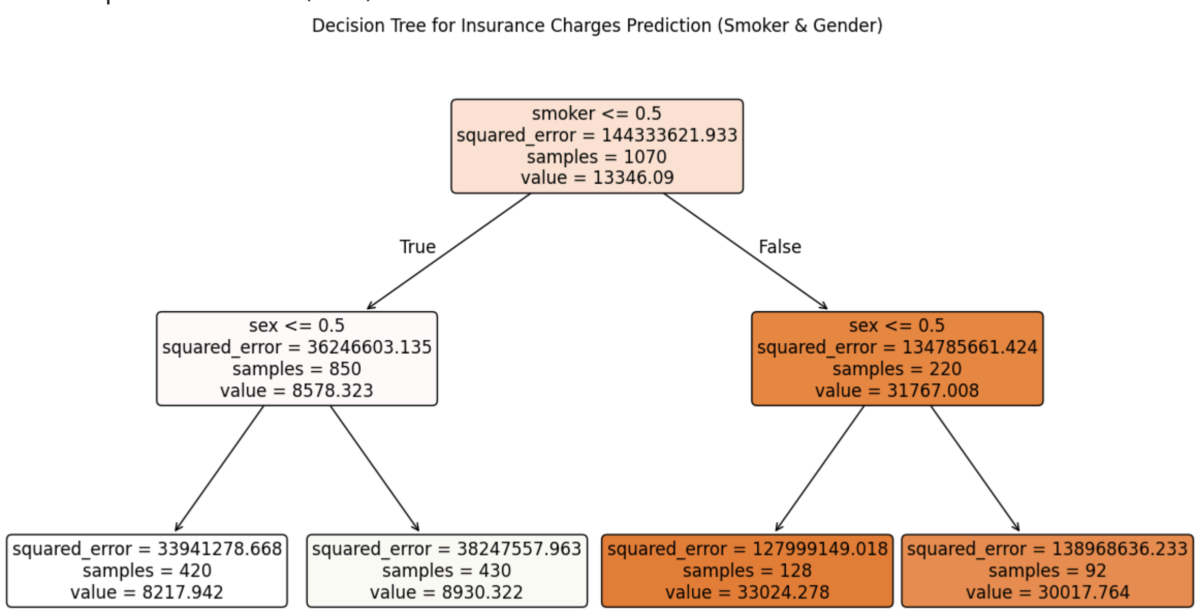
```
insurance_data
```

In [16]:

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor, export_text
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import tree
import matplotlib.pyplot as plt

# Example: Load your insurance dataset (replace this with actual dataset pat
insurance_data=pd.read_csv('/home/utk.tennessee.edu/nnaraya2/SanDisk/Endicot

# Encode categorical variables (sex, smoker)
insurance_data['sex'] = insurance_data['sex'].map({'male': 0, 'female': 1})
insurance_data['smoker'] = insurance_data['smoker'].map({'no': 0, 'yes': 1})

# Define features and target variable
X = insurance_data[['smoker', 'sex']]  # Features: smoker and sex
y = insurance_data['charges']  # Target variable

# Handle missing values
X = X.dropna()  # Drop rows with missing values in features
y = y[X.index]  # Align target variable with cleaned features

# Split the data into training and testing sets
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

# Train a Decision Tree Regressor
dt_model = DecisionTreeRegressor(max_depth=3, random_state=42)  # Adjust max
dt_model.fit(X_train, y_train)

# Predict charges on the test set
y_pred = dt_model.predict(X_test)

# Compute MSE
mse = mean_squared_error(y_test, y_pred)

# Print MSE
print(f"Mean Squared Error (MSE): {mse:.2f}")

# Visualize the Decision Tree
plt.figure(figsize=(15, 8))
tree.plot_tree(
    dt_model,
    feature_names=X.columns,
    filled=True,
    rounded=True,
    fontsize=12
)
plt.title("Decision Tree for Insurance Charges Prediction (Smoker & Gender)"
plt.show()

# Print the textual representation of the tree
tree_text = export_text(dt_model, feature_names=list(X.columns))
print("\nDecision Tree Structure:")
print(tree_text)
```

```
Mean Squared Error (MSE): 53225281.16
```



```
Decision Tree Structure:
|--- smoker <= 0.50
|   |--- sex <= 0.50
|   |   |--- value: [8217.94]
|   |--- sex >  0.50
|   |   |--- value: [8930.32]
|--- smoker >  0.50
|   |--- sex <= 0.50
|   |   |--- value: [33024.28]
|   |--- sex >  0.50
|   |   |--- value: [30017.76]
```

In [ ]:

In [ ]: