ST1 Capstone Programming Project

This project will be analysing the attributes of a medical insurance dataset to create a model for prediction of incurred charges. GITHUB LINK

My approach to this task will be incorporating the Python libraries **tkinter**, **pandas**, **seaborn**, and **matplotlib**.

Therefore, before we move on we shall import these libraries:

```
import tkinter as tk
from tkinter import filedialog, messagebox
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Step 1.

Reading the Dataset

```
# Load dataset
data = pd.read_csv('medical_insurance.csv')

# Display first few rows of the dataset in consoel
print(data.head())

# Display general information and basic statistics in console
print(data.info())
print(data.describe())
```

```
>>> print(data.head())
   age     bmi  children     charges  sex_male  smoker_yes  region_northwest  region_southeast  region_southwest
0   19  27.900         0  16884.92400     False        True             False             False              True
1   18  33.770         1   1725.55230      True       False             False              True             False
2   28  33.000         3   4449.46200      True       False             False              True             False
3   33  22.705         0  21984.47061      True       False              True             False             False
4   32  28.880         0   3866.85520      True       False              True             False             False
```

```
>>> print(data.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   int64
 1   sex       2772 non-null   object
 2   bmi       2772 non-null   float64
 3   children  2772 non-null   int64
 4   smoker    2772 non-null   object
 5   region    2772 non-null   object
 6   charges   2772 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 151.7+ KB
None
```

```
>>> print(data.describe())
               age          bmi     children       charges
count  2772.000000  2772.000000  2772.000000   2772.000000
mean     39.109668    30.701349     1.101732  13261.369959
std      14.081459     6.129449     1.214806  12151.768945
min      18.000000    15.960000     0.000000   1121.873900
25%      26.000000    26.220000     0.000000   4687.797000
50%      39.000000    30.447500     1.000000   9333.014350
75%      51.000000    34.770000     2.000000  16577.779500
max      64.000000    53.130000     5.000000  63770.428010
```

Key observations from Step 1

- The dataset is a sample of 2772 people with medical insurance
- The attributes include age, bmi, number of children, charges, gender, smoker, and region.
- Gender, smoker and region attributes are functionally Booleans, being True or False. For example, sex_male = False would indicate a female.

Step 2.

Problem Statement Definition

- Creating a prediction model in order to predict the price of medical insurance based on given attributes.

Step 3.

Target Variable Identification

- Target Variable: Charges (Affected by: Age, Sex, Smoker, Region, etc)

Step 4.

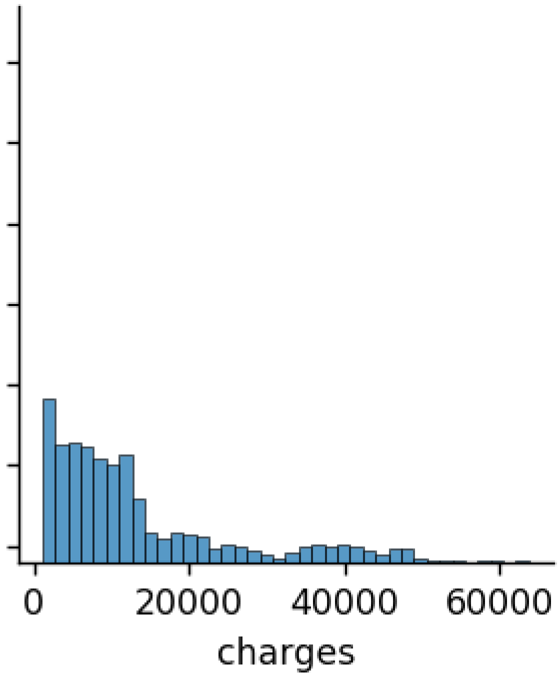
Choosing appropriate Algorithm for Data Analysis

- The target variable is Continuous, and we shall be proceeding with a Ridge regression model as we expect a linear relationship.

Visualising the distribution of Target Variable

```
# Pair plot to examine relationships among numerical features
sns.pairplot(data)
plt.show()
```

- We can see that the data has a relatively linear relationship, with most charges being weighted towards the lower end.
- We shall proceed and investigate how the other variables affect this.



Step 5.

Data Exploration at the Basic Level

- As per Step 1.

```
# Display first few rows of the dataset in consoel
print(data.head())

# Display general information and basic statistics in console
print(data.info())
print(data.describe())
```

```
>>> print(data.head())
   age     bmi  children      charges  sex_male  smoker_yes  region_northwest  region_southeast  region_southwest
0   19  27.900         0  16884.92400     False        True             False             False              True
1   18  33.770         1   1725.55230      True       False             False              True             False
2   28  33.000         3   4449.46200      True       False             False              True             False
3   33  22.705         0  21984.47061      True       False              True             False             False
4   32  28.880         0   3866.85520      True       False              True             False             False
```

```
>>> print(data.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   int64
 1   sex       2772 non-null   object
 2   bmi       2772 non-null   float64
 3   children  2772 non-null   int64
 4   smoker    2772 non-null   object
 5   region    2772 non-null   object
 6   charges   2772 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 151.7+ KB
None
```

```
>>> print(data.describe())
               age          bmi     children       charges
count  2772.000000  2772.000000  2772.000000   2772.000000
mean     39.109668    30.701349     1.101732  13261.369959
std      14.081459     6.129449     1.214806  12151.768945
min      18.000000    15.960000     0.000000   1121.873900
25%      26.000000    26.220000     0.000000   4687.797000
50%      39.000000    30.447500     1.000000   9333.014350
75%      51.000000    34.770000     2.000000  16577.779500
max      64.000000    53.130000     5.000000  63770.428010
```

- We expect that the target variable will be affected by number of *children*, *sex*, whether they are a *smoker*, and their *region*. BMI could be a potential factor.
- Age – Continuous
- BMI – Continuous
- Children – Continuous
- Charges – Continuous
- Sex – Categorical
- Smoker – Categorical
- Region – Categorical

Step 6.

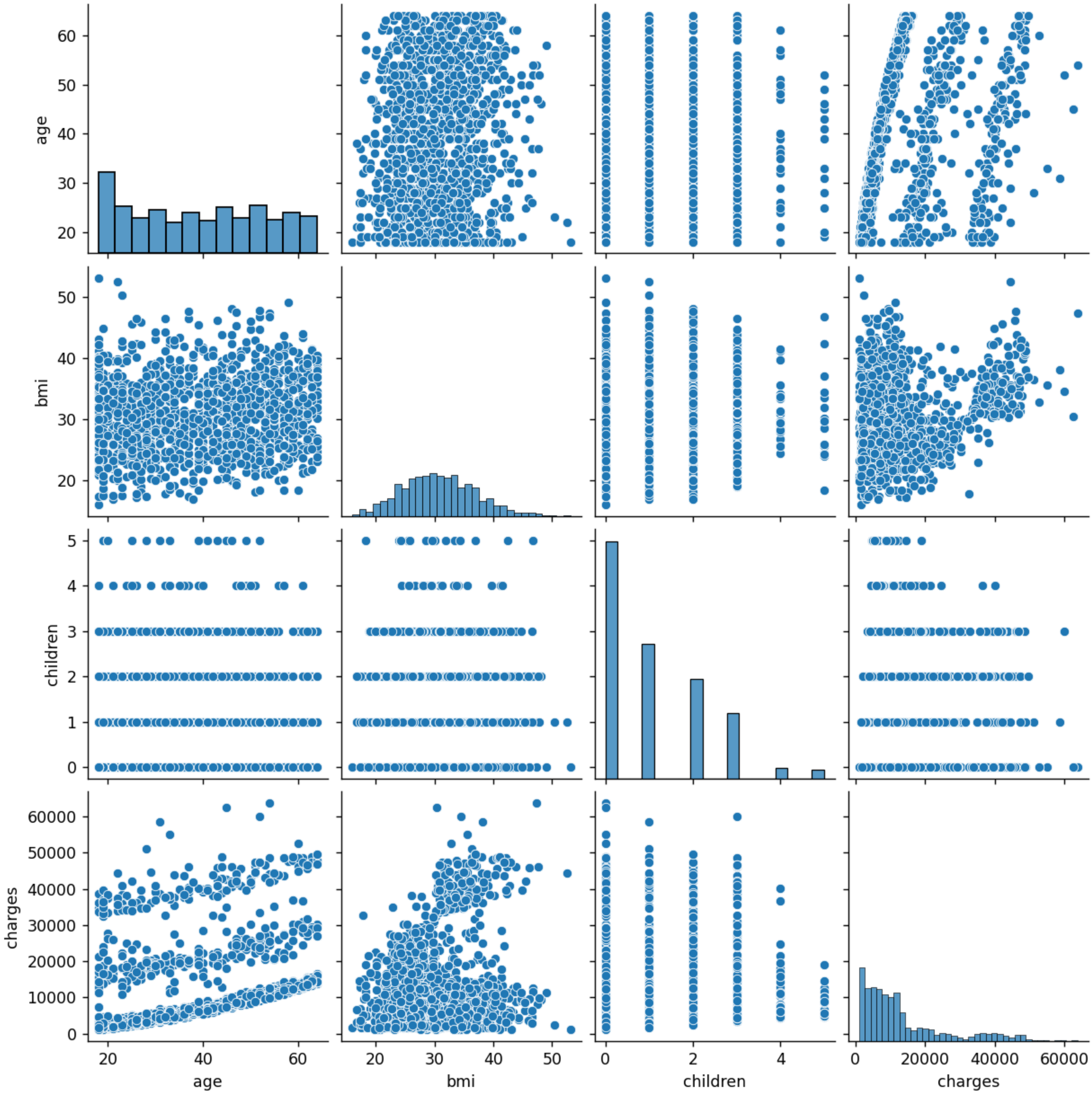
Identifying and Rejecting unwanted columns

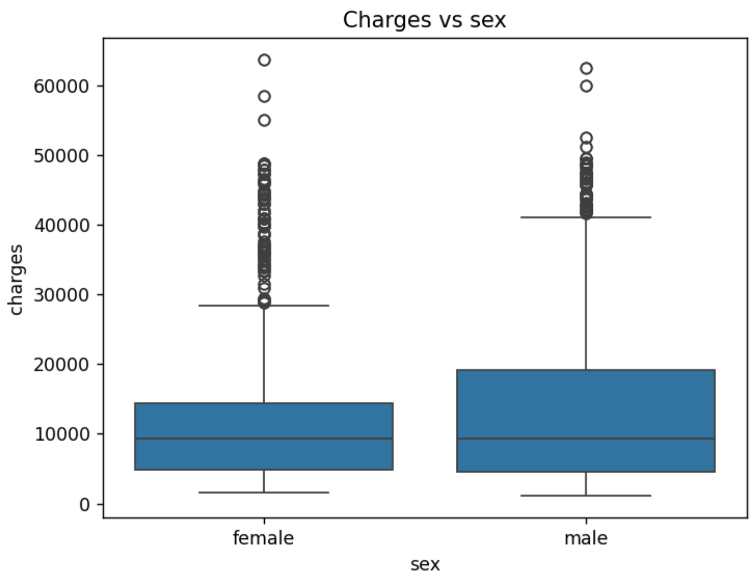
- Given our variables, there should be no need to remove any columns
- I suspect they shall all have some effect

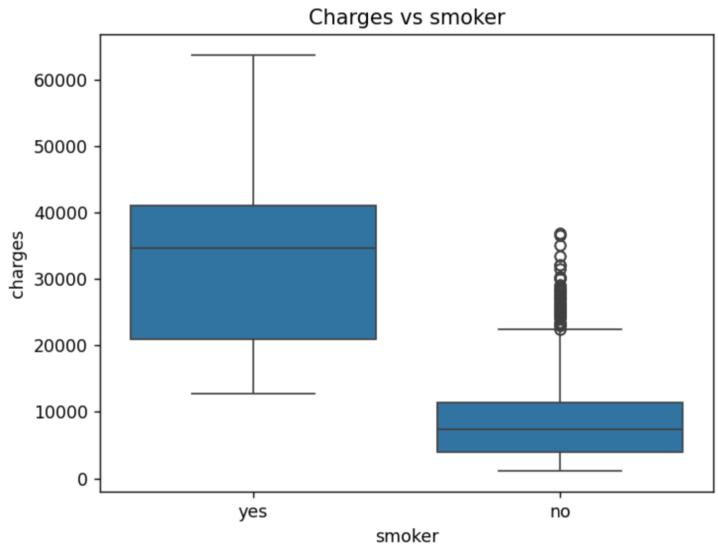
Step 7.

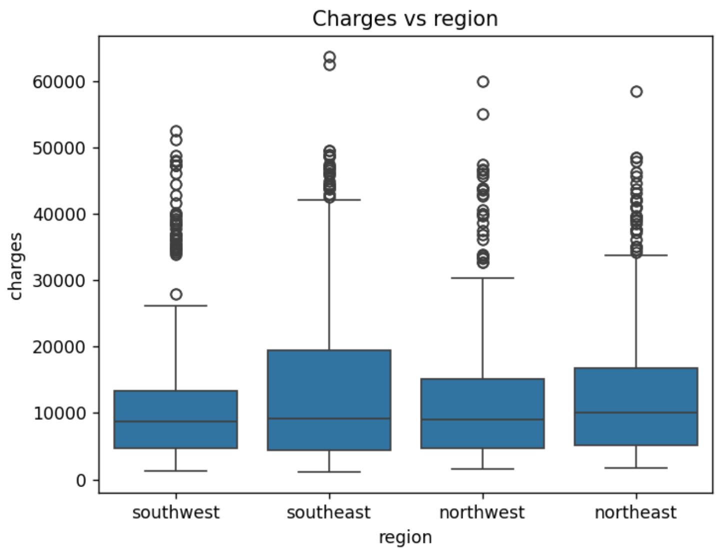
Visual Exploratory Data Analysis

```
# Pair plot to examine relationships among numerical features
sns.pairplot(data)
plt.show()
```









Observations from Step 7.

- The pairplot shows the non-categorical data from the dataset.
- The variables are represented on both X and Y axis
- We can see how some variables such as BMI form a nice bell curve indicating a sufficient and diverse sample of people
- Some charts are less relevant although still do have features that indicate a relationship.
- As for the bar charts, I noticed by far the most difference between whether a person smoked or not. If they did, they had to pay a substantial amount more than a non-smoker, which makes sense.
- Males cost more on average along with greater standard deviation
- Some regions have larger standard deviations and extremes although all had a similar median

Step 8.

Feature Selection based on data distribution

- Despite some graphs being rather dense due to the dataset density, all show signs of being relevant to the target variable.
- Therefore we shall move on and analyse further

Step 9.

Outliers and Missing Values

```
# Calculates the Interquartile Range for charges
Q1 = data['charges'].quantile(0.25)
Q3 = data['charges'].quantile(0.75)
IQR = Q3 - Q1

# Define bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the dataset to remove outliers
filtered_data = data[(data['charges'] >= lower_bound) & (data['charges'] <=
upper_bound)]

# Check the size of the filtered dataset compared to the original dataset
print("Original dataset size:", len(data))
print("Filtered dataset size:", len(filtered_data))
```

- After running this code, we have trimmed down the dataset from 2772 to 2476, which is a moderate amount of "outliers" given the total and suggests that the data is relatively consistent.

Step 10.

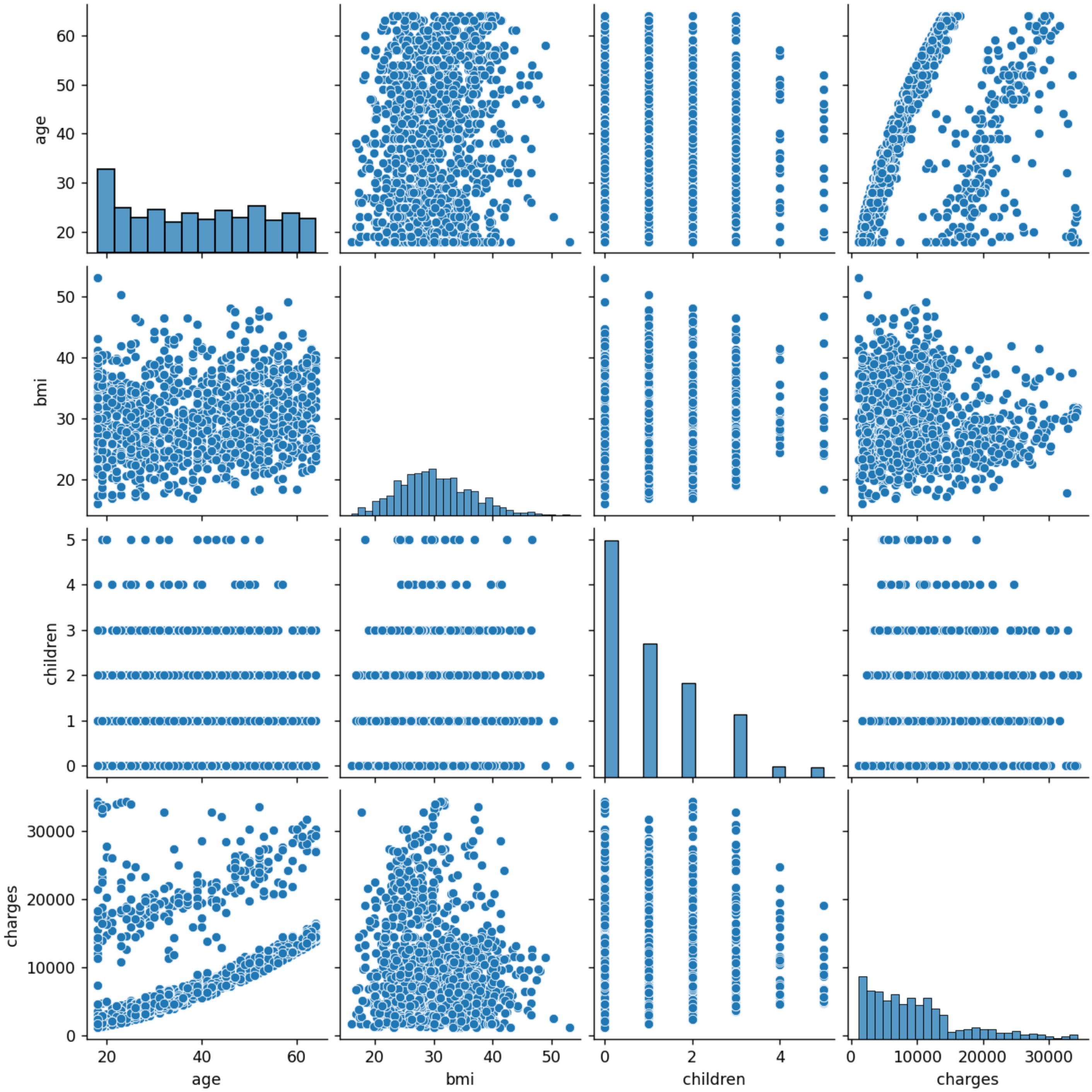
Visualising data after outlier removal

```
# Pair plot to examine relationships among numerical features
sns.pairplot(filtered_data)
plt.show()

# Analyse categorical variables using boxplots
for column in ['sex', 'smoker', 'region']:
```

```python
sns.boxplot(x=column, y='charges', data=filtered_data)
plt.title(f"Charges vs {column}")
plt.show()
```

- Data has been changed to filtered data for analysis



Observations from Step 10.

- There is little visible difference in the pairplot although certainly present
- Greatest change can be seen in the target variable 'charges'
- The Interquartile Range filtering may have adverse effects on the overall data as now it potentially may be too consistent
- We no longer have charges approaching 20K
- Tail is still fairly linear

Step 11.

Data Conversion to numeric values for machine learning

```
# Convert categorical variables to one-hot encoded format
data = pd.get_dummies(data, columns=['sex', 'smoker', 'region'],
drop_first=True)
```

- Convert categorical data into pandas dummy variables
- drop_first=True to filter out 'Unknown' gender
- Picked these variables as they seemed to be the best features

Step 12.

Training/Testing

```
from sklearn.model_selection import train_test_split

# Define features and target variable
X = data.drop('charges', axis=1)
y = data['charges']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score

# Create and train the Ridge regression model
model = Ridge(alpha=1.0) # Default alpha 1.0 as per Ridge documentation
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)
```

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

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

- Uses fairly default parameters based on documentation
- Trains science kit Ridge regression model

Step 13.

Selection of the best model

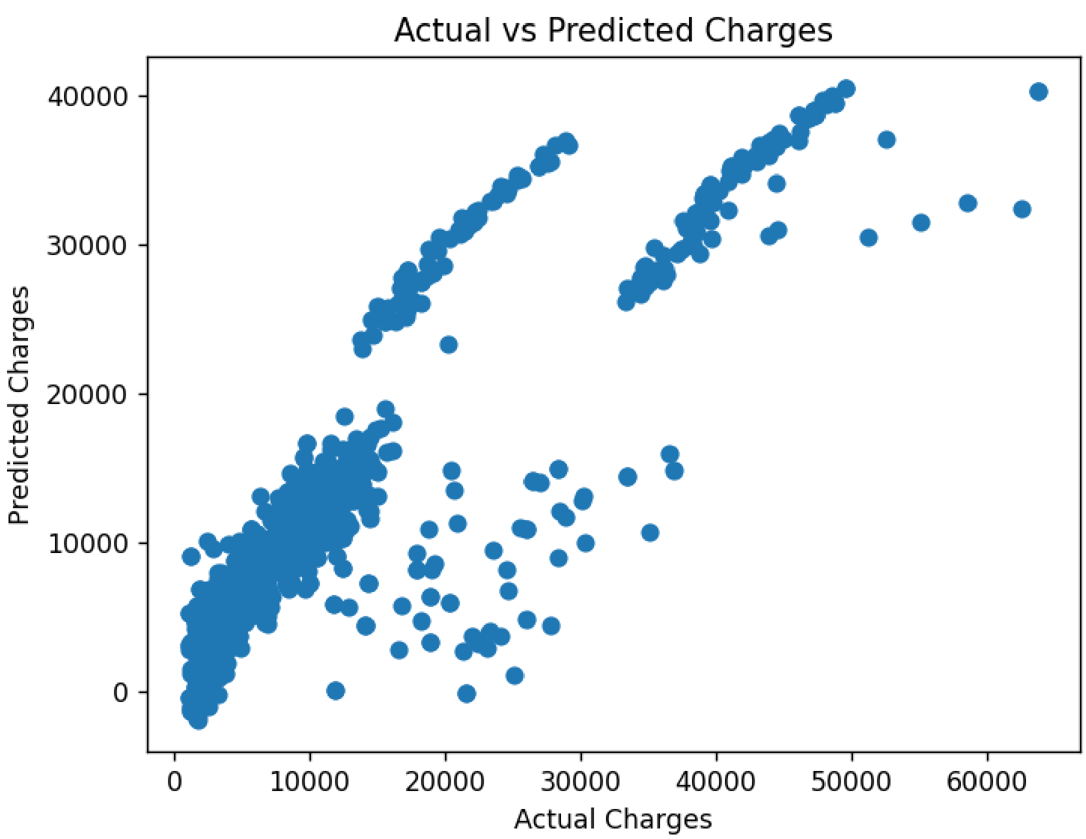
- Ridge regression was the only one that worked during my scope of the project
- However, it's a good general use model that still allows us to analyse the data.

Step 14.

Deployment of the Model

```python
# Scatter plot of actual vs predicted charges
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Charges")
plt.ylabel("Predicted Charges")
plt.title("Actual vs Predicted Charges")
plt.show()
```

- Together with the previous code, this presents us with a scatter plot of the actual vs predicted charges.



Observations from Step 14.

- Given our chosen categorical variables of Sex, Smoker and Region, we can see that there is a moderate relationship between the variables.
- As expected of a defined linear relationship, the path is dense
- Actual charges align fairly closely to the predicted charges, although with more variation.

Step 16.

GUI Deployment

```
# Root window for the Tkinter GUI
root = tk.Tk()
root.title("Insurance Analysis")

# Generic file loading for flexibility
def load_data():
    # Open a file dialog to select the CSV file
    file_path = filedialog.askopenfilename(title="Select CSV File",
filetypes=[("CSV Files", "*.csv")])
    if not file_path:
        messagebox.showwarning("No File Selected", "Please select a CSV
file.")
        return None

    # Loads data
    data = pd.read_csv(file_path)
    return data

# Displays data information
def display_data_info():
    data = load_data()
    if data is not None:
        # Display basic info
        info = data.info()
        print(info)  # For console output
        # Show basic stats in a new window
        new_window = tk.Toplevel(root)
        new_window.title("Data Information")
        info_label = tk.Label(new_window, text=str(data.describe()),
justify=tk.LEFT)
        info_label.pack()

# Create scatter plots
def create_scatter_plot():
    data = load_data()
    if data is not None:
        plt.scatter(data['age'], data['charges'])
```

```python
        plt.xlabel("Age")
        plt.ylabel("Charges")
        plt.title("Age vs Charges")
        plt.show()

# Define a function to create box plots for categorical data
def create_box_plots():
    data = load_data()
    if data is not None:
        for column in ['sex', 'smoker', 'region']:
            sns.boxplot(x=column, y='charges', data=data)
            plt.title(f"Charges vs {column}")
            plt.show()

# Create buttons
info_button = tk.Button(root, text="Display Data Information",
command=display_data_info)
scatter_button = tk.Button(root, text="Create Scatter Plot",
command=create_scatter_plot)
box_plot_button = tk.Button(root, text="Create Box Plots",
command=create_box_plots)

# Place the buttons in the GUI
info_button.pack(pady=10, padx=10)
scatter_button.pack(pady=10, padx=10)
box_plot_button.pack(pady=10, padx=10)

# Start the Tkinter event loop
root.mainloop()
```

- Creates a Tkinter GUI interface
- Made generic to use on any CSV file
- Is able to display basic data from the file
- Can present the scatter plot from earlier
- Can go through the sequence of box plots from earlier