practical-4-sneha-1

August 30, 2023

[124]:
```
import pandas as pd
```

[125]:
```
import random
import matplotlib.pyplot as plt
```

[126]:
```
import numpy as np
```

[127]:
```
import seaborn as sns
```

[128]:
```
from sklearn.preprocessing import LabelEncoder
```

[129]:
```
from sklearn.model_selection import train_test_split
```

[130]:
```
from sklearn. linear_model import LogisticRegression
```

[131]:
```
from sklearn. linear_model import LinearRegression
```

[132]:
```
import matplotlib.pyplot as plt
```

[133]:
```
md= pd.read_csv('medical.csv')
md
```

[133]:

	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
…	…	…	…	…	…	…	…
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

[1338 rows x 7 columns]

[134]:
```
md.sample(10)
```

[134]:

	age	sex	bmi	children	smoker	region	charges
833	58	male	34.39	0	no	northwest	11743.9341
351	50	female	25.60	0	no	southwest	8932.0840
766	47	male	32.30	1	no	southwest	8062.7640
837	56	female	28.31	0	no	northeast	11657.7189
399	18	female	38.17	0	no	southeast	1631.6683
1	18	male	33.77	1	no	southeast	1725.5523
709	36	female	27.74	0	no	northeast	5469.0066
260	58	female	25.20	0	no	southwest	11837.1600
630	53	male	36.10	1	no	southwest	10085.8460
970	50	female	28.16	3	no	southeast	10702.6424

[135]:
```
print ("Number Of applicants:" +str(len(md.index)))
```

```
Number Of applicants:1338
```

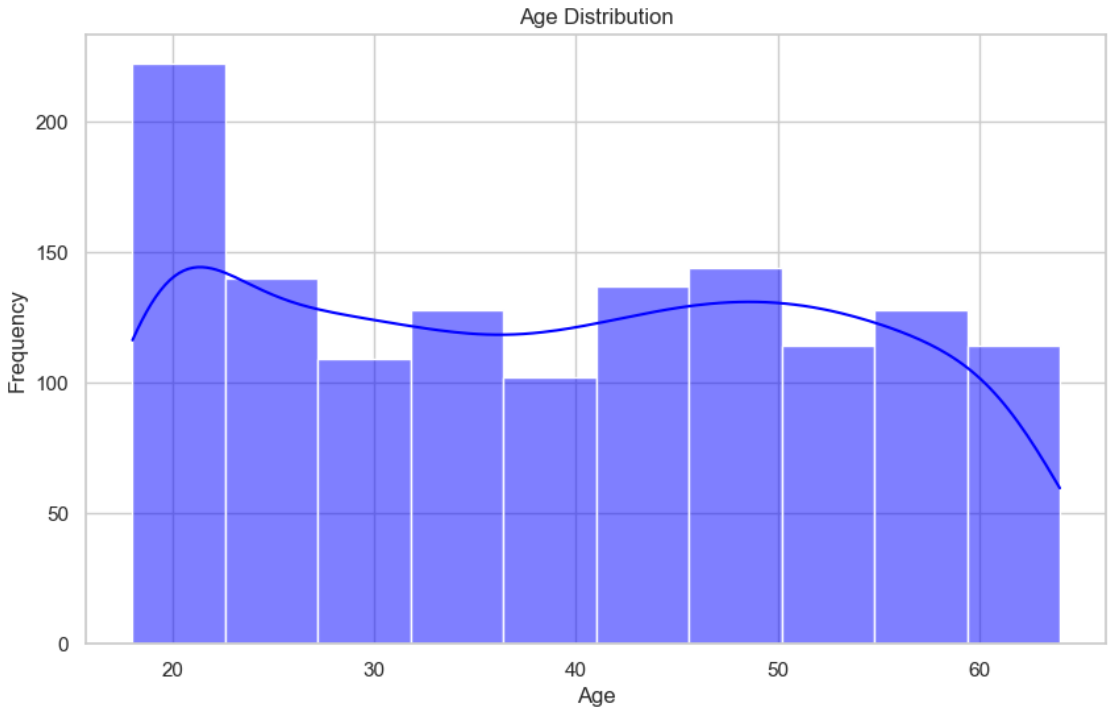
[136]:
```
md.columns
```

[136]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'],
dtype='object')
```

[137]:
```
sns.color_palette("mako", as_cmap=True)
```

[137]:

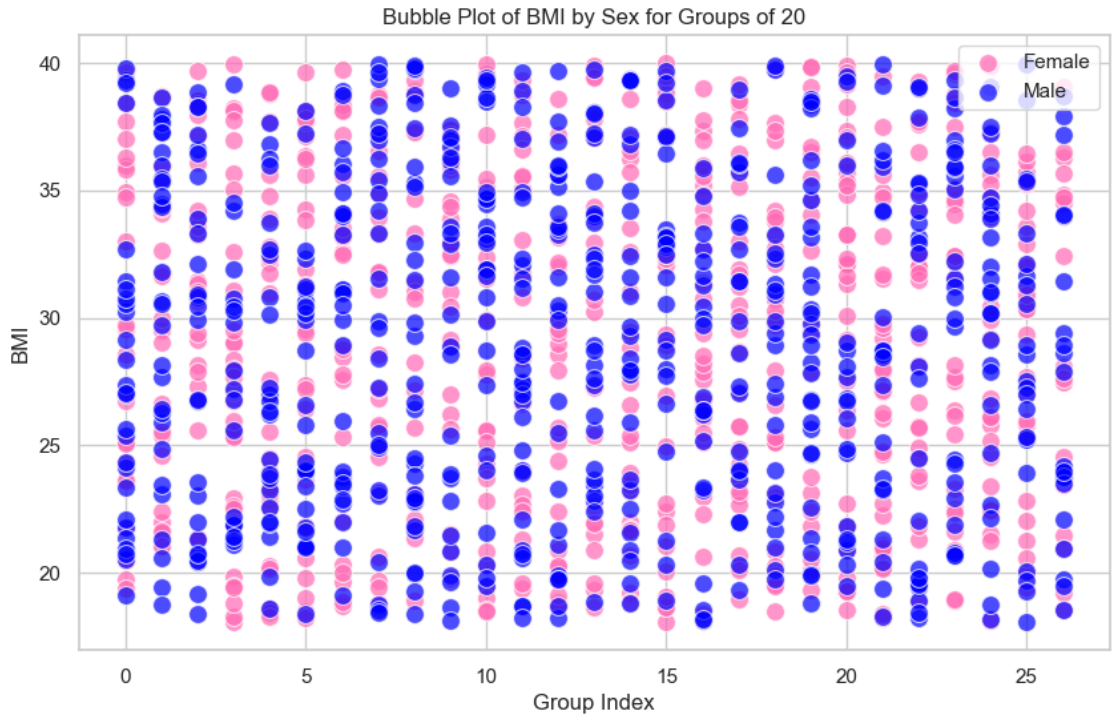
[138]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.histplot(data=md, x='age', bins=10, kde=True, color='blue')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution')
plt.show()
```



The presented histogram illustrates a distinct concentration of individuals in their 20s who have acquired insurance, demonstrating a higher participation rate within this age group compared to subsequent decades. The distribution pattern suggests a noticeable drop in numbers beyond the 20s, resulting in a relatively uniform distribution across age ranges beyond the early twenties.

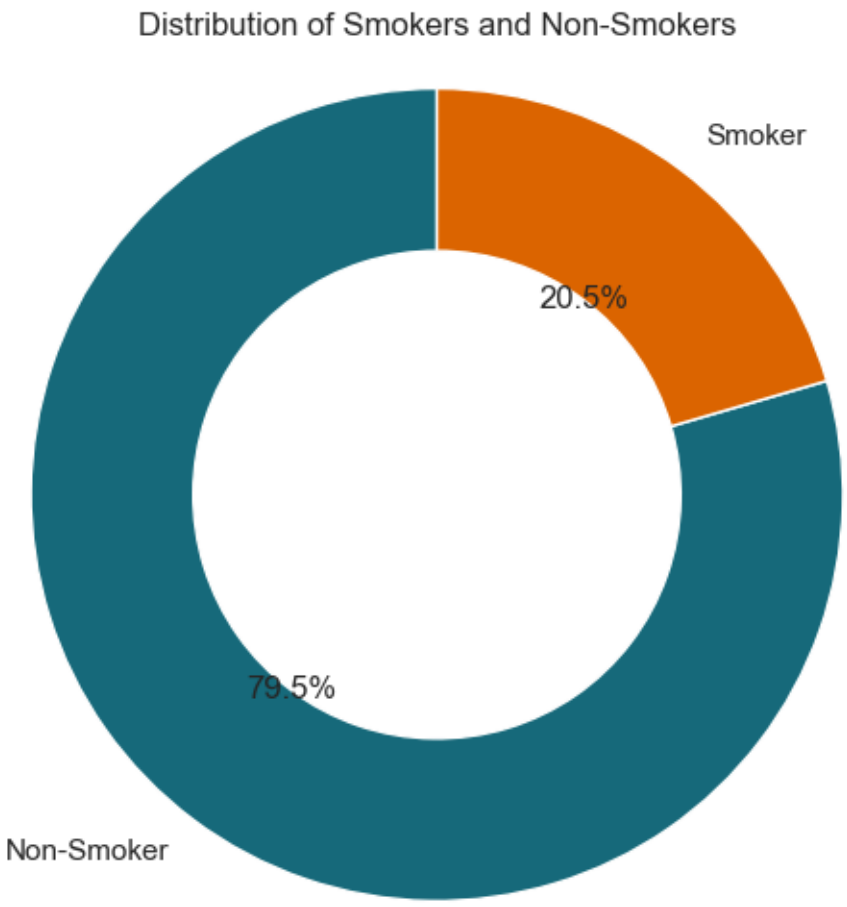
[139]:
```
sns.color_palette("mako", as_cmap=True)
# Create sample data
np.random.seed(42)
data = {
    'sex': np.random.choice(['female', 'male'], size=1338),
    'bmi': np.random.uniform(18, 40, size=1338)
}

# Create a DataFrame
df = pd.DataFrame(data)

# Calculate group indices
group_size = 50
df['group_index'] = df.index // group_size

# Set the plotting style and size
sns.set(style='whitegrid')
plt.figure(figsize=(10, 6))
```

```
# Create separate scatter plots for females and males
female_df = df[df['sex'] == 'female']
male_df = df[df['sex'] == 'male']

sns.scatterplot(x='group_index', y='bmi', data=female_df, color='hotpink',␣
 ↪s=100, alpha=0.7, label='Female')
sns.scatterplot(x='group_index', y='bmi', data=male_df, color='blue', s=100,␣
 ↪alpha=0.7, label='Male')

# Adding labels and title
plt.xlabel('Group Index')
plt.ylabel('BMI')
plt.title('Bubble Plot of BMI by Sex for Groups of 20')

# Display the legend
plt.legend()

# Display the graph
plt.show()
```

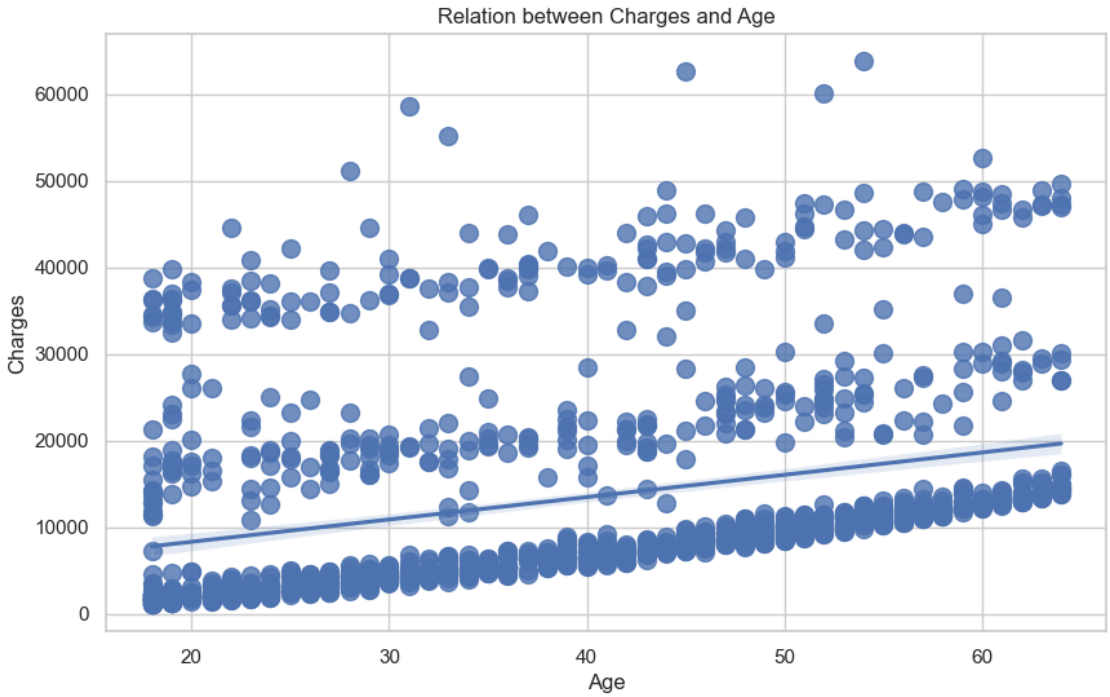


[140]:
```
# Count the occurrences of smokers and non-smokers
smoker_counts = md['smoker'].value_counts()
```

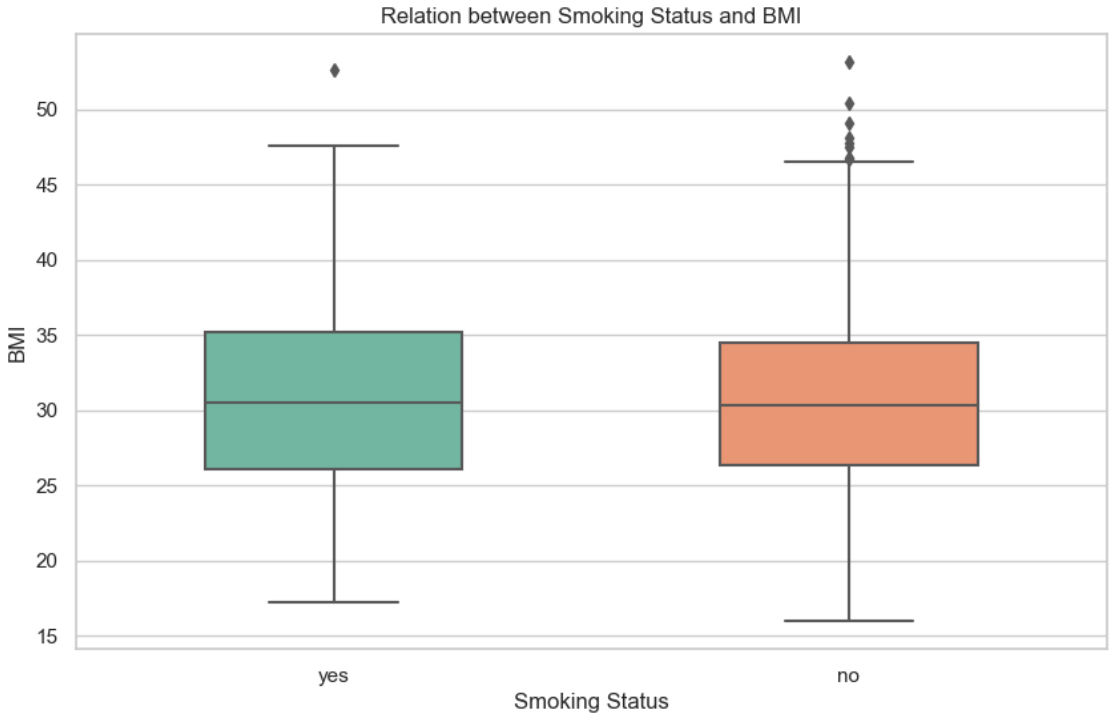
```
# Create a figure and axis
fig, ax = plt.subplots(figsize=(6, 6))

# Colors for the sections
colors = ['#16697A', '#DB6400']

# Create a donut plot
wedges, texts, autotexts = ax.pie(smoker_counts, labels=['Non-Smoker',␣
 ↪'Smoker'], autopct='%1.1f%%',
                                   startangle=90, colors=colors,␣
 ↪wedgeprops=dict(width=0.4, edgecolor='w'))

# Adding a circle in the center to create a donut appearance
centre_circle = plt.Circle((0,0),0.30,fc='white')
fig.gca().add_artist(centre_circle)

# Equal aspect ratio ensures that pie is drawn as a circle.
ax.axis('equal')

# Adding title
plt.title('Distribution of Smokers and Non-Smokers')

# Display the plot
plt.show()
```



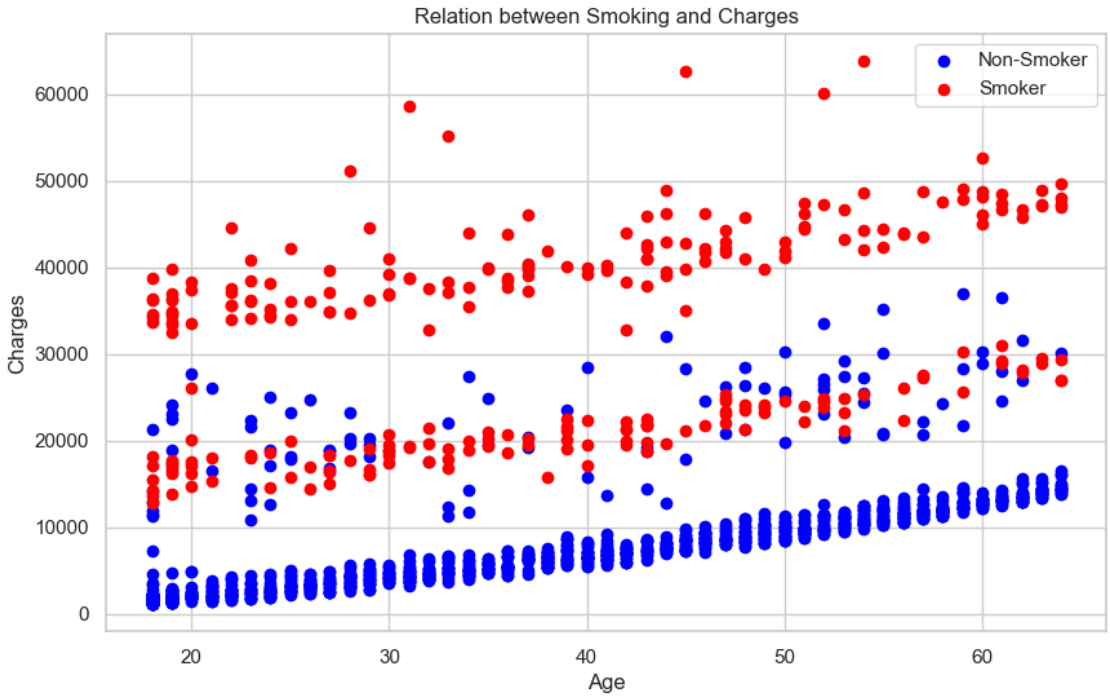
[141]:
```
plt.figure(figsize=(10, 6))
sns.regplot(x='age', y='charges', data=md, scatter_kws={'s': 100})
plt.xlabel('Age')
plt.ylabel('Charges')
plt.title('Relation between Charges and Age')
plt.grid(True)
plt.show()
```



[142]:
```
# Create a box plot or violin plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='smoker', y='bmi', data=md, palette='Set2', width=0.5)  # You can␣
 ↪use sns.violinplot() for violin plot

# Add labels and title
plt.xlabel('Smoking Status')
plt.ylabel('BMI')
plt.title('Relation between Smoking Status and BMI')

plt.show()
```

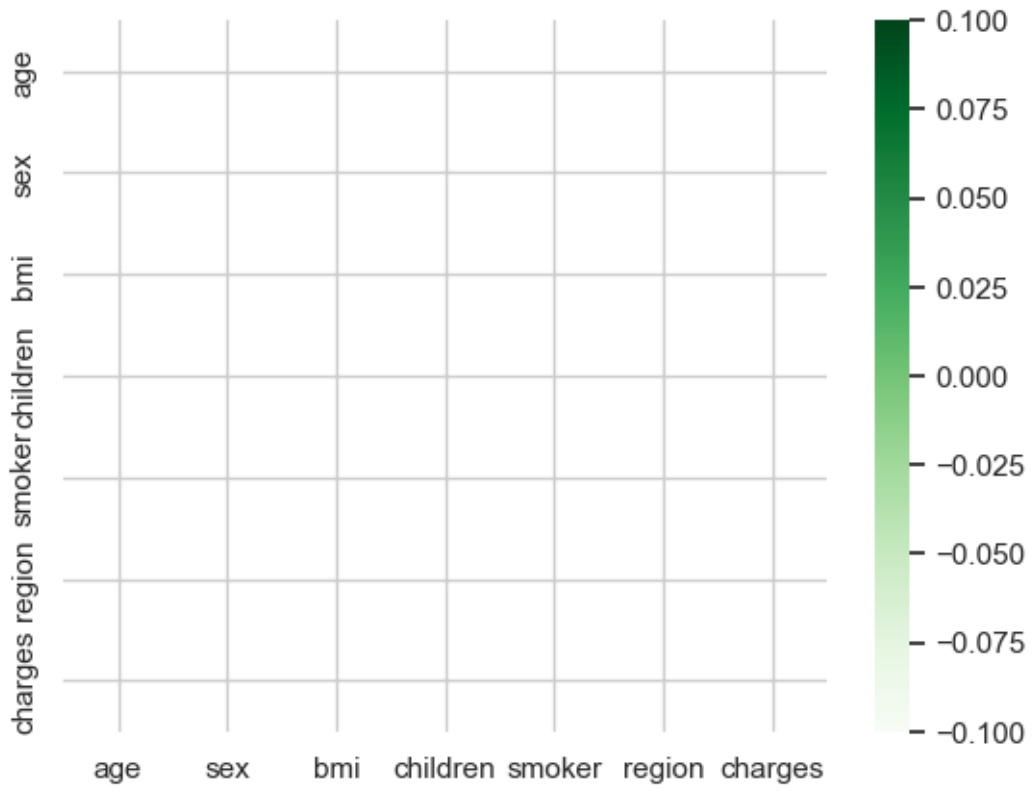


[143]:
```
# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(md[md['smoker'] == 'no']['age'], md[md['smoker'] ==␣
 ↪'no']['charges'], label='Non-Smoker', color='blue')
plt.scatter(md[md['smoker'] == 'yes']['age'], md[md['smoker'] ==␣
 ↪'yes']['charges'], label='Smoker', color='red')

# Add labels and title
plt.xlabel('Age')
plt.ylabel('Charges')
plt.title('Relation between Smoking and Charges')
plt.legend()

plt.show()
```



[144]:
```
matrix = md[md.smoker==0].corr()
matrix.charges[:-1]
sns.heatmap(matrix, annot=True, cmap='Greens');
```

```
C:\Users\lenovo\AppData\Roaming\Python\Python311\site-
packages\seaborn\matrix.py:202: RuntimeWarning: All-NaN slice encountered
  vmin = np.nanmin(calc_data)
C:\Users\lenovo\AppData\Roaming\Python\Python311\site-
packages\seaborn\matrix.py:207: RuntimeWarning: All-NaN slice encountered
  vmax = np.nanmax(calc_data)
```



[145]:
```
md.isnull().sum()
```

[145]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

[146]:
```
md.info()
```

```
<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
```

[147]:
```
from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder()
```

[148]:
```
md['sex'] = LE.fit_transform(md.sex)
md.info()
```

```
<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   int32
 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), int32(1), int64(2), object(2)
memory usage: 68.1+ KB
```

[149]:
```
md['smoker'] = LE.fit_transform(md.smoker)
md['region'] = LE.fit_transform(md.region)
```

[150]:
```
md.info()
```

```
<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   int32
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   int32
 5   region    1338 non-null   int32
 6   charges   1338 non-null   float64
dtypes: float64(2), int32(3), int64(2)
memory usage: 57.6 KB
```

[174]:
```
# Display unique values in the 'region' column
print(md['region'].unique())

# Map integers to new integer labels
int_labels = {0: 0, 1: 1, 2: 2, 3: 3}
md['region_int'] = md['region'].map(int_labels)

# Display the updated DataFrame
print(md.head())
```

```
[3, 2, 1, 0]
Categories (4, int32): [0, 1, 2, 3]
   age  sex     bmi  children  smoker region      charges region_int
0   19    0  27.900         0       1      3  16884.92400          3
1   18    1  33.770         1       0      2   1725.55230          2
2   28    1  33.000         3       0      2   4449.46200          2
3   33    1  22.705         0       0      1  21984.47061          1
4   32    1  28.880         0       0      1   3866.85520          1
```

[175]:
```
print(md['region'].head())
print(md['region'].unique())
```

```
0    3
1    2
2    2
3    1
4    1
Name: region, dtype: category
Categories (4, int32): [0, 1, 2, 3]
[3, 2, 1, 0]
Categories (4, int32): [0, 1, 2, 3]
```

[176]:
```
md.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         1338 non-null   int64
 1   sex         1338 non-null   int32
 2   bmi         1338 non-null   float64
 3   children    1338 non-null   int64
 4   smoker      1338 non-null   int32
 5   region      1338 non-null   category
 6   charges     1338 non-null   float64
 7   region_int  1338 non-null   category
dtypes: category(2), float64(2), int32(2), int64(2)
```

```
memory usage: 55.3 KB
```

[177]:
```
print(md['region'].unique())

# Convert the 'region' column to integers
md['region_int'] = md['region'].cat.codes

# Display the updated DataFrame
print(md.head())
```

```
[3, 2, 1, 0]
Categories (4, int32): [0, 1, 2, 3]
   age  sex     bmi  children  smoker region      charges  region_int
0   19    0  27.900         0       1      3  16884.92400           3
1   18    1  33.770         1       0      2   1725.55230           2
2   28    1  33.000         3       0      2   4449.46200           2
3   33    1  22.705         0       0      1  21984.47061           1
4   32    1  28.880         0       0      1   3866.85520           1
```

[178]:
```
md.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         1338 non-null   int64
 1   sex         1338 non-null   int32
 2   bmi         1338 non-null   float64
 3   children    1338 non-null   int64
 4   smoker      1338 non-null   int32
 5   region      1338 non-null   category
 6   charges     1338 non-null   float64
 7   region_int  1338 non-null   int8
dtypes: category(1), float64(2), int32(2), int64(2), int8(1)
memory usage: 55.2 KB
```

[186]:
```
print(md.columns)

# Drop the 'region' column
column_to_drop = 'region'
md = md.drop(column_to_drop, axis=1)

# Display the updated DataFrame
print(md.head())
```

```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges',
       'region_int'],
```

```
      dtype='object')
```

	age	sex	bmi	children	smoker	charges	region_int
0	19	0	27.900	0	1	16884.92400	3
1	18	1	33.770	1	0	1725.55230	2
2	28	1	33.000	3	0	4449.46200	2
3	33	1	22.705	0	0	21984.47061	1
4	32	1	28.880	0	0	3866.85520	1

[187]:
```
md.info()
```

```
<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   int32
 2   bmi         1338 non-null   float64
 3   children    1338 non-null   int64
 4   smoker      1338 non-null   int32
 5   charges     1338 non-null   float64
 6   region_int  1338 non-null   int8
dtypes: float64(2), int32(2), int64(2), int8(1)
memory usage: 53.7 KB
```

[188]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[194]:
```
from sklearn.metrics import mean_squared_error

# Train a linear regression model
model1 = LinearRegression()
model1.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model1.predict(X_test)

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 33635210.431178406
```

[196]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import accuracy_score, confusion_matrix

# Assuming you have performed EDA and prepared X and y
X = md.drop('smoker', axis=1)
y = md['smoker']

# Split 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)

# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

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

# Calculate accuracy and confusion matrix
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.9402985074626866
Confusion Matrix:
[[207   7]
 [  9  45]]
```

[197]:
```
from sklearn.linear_model import LinearRegression
model_regression = LinearRegression()
model_regression.fit(X_train, y_train)
```

[197]:
```
LinearRegression()
```

[198]:
```
from sklearn.metrics import mean_squared_error, r2_score
y_pred_regression = model_regression.predict(X_test)
mse = mean_squared_error(y_test, y_pred_regression)
r2 = r2_score(y_test, y_pred_regression)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 0.04153118534805607
R-squared: 0.7418712481447924
```

[199]:
```
from sklearn.linear_model import LogisticRegression
model_classification = LogisticRegression()
model_classification.fit(X_train, y_train)
```

[199]:
```
LogisticRegression()
```

[203]:
```
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
y_pred_classification = model_classification.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_classification)
classification_rep = classification_report(y_test, y_pred_classification)
conf_matrix = confusion_matrix(y_test, y_pred_classification)
print(f'Accuracy: {accuracy*100:.2f}' + ' %')
# print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(classification_rep)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 94.03 %
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       214
           1       0.87      0.83      0.85        54

    accuracy                           0.94       268
   macro avg       0.91      0.90      0.91       268
weighted avg       0.94      0.94      0.94       268

Confusion Matrix:
[[207   7]
 [  9  45]]
```

[202]:
```
from sklearn.linear_model import LogisticRegression

# Assuming you have prepared X_train and y_train for the multi-class␣
 ↪classification scenario
model_multiclass = LogisticRegression(multi_class='ovr')  # 'ovr' or␣
 ↪'multinomial'
model_multiclass.fit(X_train, y_train)  # Use y_train for multi-class, not␣
 ↪y_train_multiclass
```

[202]:
```
LogisticRegression(multi_class='ovr')
```

[ ]:
```

```