STAT603-Fall2023-FinalExam

December 11, 2023

1 Final Exam Fall 2023

Question 1.An airline runs a small commuter flight that has 10 seats. The probability that a passenger turns up for the flight is 0.92. What is the smallest number of seats the airline should sell to ensure that the probability the flight will be full is greater than 0.93?

```
[26]: import math
      from scipy.stats import norm
      import numpy as np
      from scipy.stats import expon
      import matplotlib.pyplot as plt
      import seaborn as sns
      import pandas as pd
      import warnings
      warnings.filterwarnings('ignore')
```

```
[27]: # Given probabilities
      probability_turn_up = 0.92
      desired_probability_full = 0.93

      # Calculate the smallest number of seats
      required_seats = math.ceil(math.log(desired_probability_full) / math.
       ↪log(probability_turn_up))

      print(f"The smallest number of seats to ensure a probability greater than␣
       ↪{desired_probability_full} is: {required_seats}")
```

The smallest number of seats to ensure a probability greater than 0.93 is: 1

Question 2.The typical computer random number generator yields numbers in a uniform distribution between 0 and 1. If 45 random numbers are generated, find approximately the probability that their mean is below 0.465, using the central limit theorem.

```
[28]: # Given values
      sample_size = 45
      mean_uniform = 1/2
      std_dev_uniform = 1/math.sqrt(12)

      # Calculate standard error of the mean
```

```
se = std_dev_uniform / math.sqrt(sample_size)

# Value for which we want to find the probability
X = 0.465

# Calculate Z-score
Z = (X - mean_uniform) / se

# Find the probability using the cumulative distribution function (CDF) of the
 ↪standard normal distribution
probability_below_X = norm.cdf(Z)

print(f"The probability that the mean is below {X} is approximately:␣
 ↪{probability_below_X:.4f}")
```

The probability that the mean is below 0.465 is approximately: 0.2080

Question 3.A farmer is interested in knowing the mean weight of his chickens when they leave the farm. Suppose that the standard deviation of the chicken's weight is 500 grams.

(a) What is the minimum number of chickens needed to ensure that the standard deviation of the sample mean is no more than 90 grams?

(b) Suppose the farm has three coops. The mean weights in each coop are 1.75, 1.85and 2.1 kg, and standard deviations are 450, 520, and 380 grams, respectively. Calculate the probability that a random sample of 30 chickens from the first coop will have a mean weight larger than 1.925 kg. Calculate the same probability for the second and third coops.

(c) Suppose the proportion of the three coops are 0.60, 0.25, 0.15. Given that a random sample of 30 chickens from some coop has a mean weight larger than 1.925 kg, find the posterior probability the sample is from the (i) first coop, (ii) second coop, (iii) third coop. Which coop did the sample of chickens most likely have come from?

```
[29]: #Part(a)

# Given values
population_std_dev = 500
desired_SE = 90

# Calculate the minimum sample size
min_sample_size = math.ceil((population_std_dev / desired_SE)**2)

print(f"The minimum number of chickens needed is: {min_sample_size}")
```

The minimum number of chickens needed is: 31

```
[30]: #Part(b)

# Given values for the first coop
```

```
mean_coop1 = 1.75
std_dev_coop1 = 450
sample_size_coop1 = 30
mean_weight_threshold = 1.925

# Calculate the z-score
z_score_coop1 = (mean_weight_threshold - mean_coop1) / (std_dev_coop1 / math.
  ↪sqrt(sample_size_coop1))

# Calculate the probability using the complementary cumulative distribution␣
  ↪function (1 - CDF)
probability_above_threshold_coop1 = 1 - norm.cdf(z_score_coop1)

print(f"Probability for the first coop: {probability_above_threshold_coop1:.
  ↪4f}")
```

Probability for the first coop: 0.4992

```
[31]: # Part(c)
#P(Coop Mean weight > 1.925 kg)=P(Mean weight > 1.925 kg)/P(Mean weight > 1.925␣
  ↪kg Coop)×P(Coop)


# Given values for the coops
mean_coop1 = 1.75
std_dev_coop1 = 450
sample_size_coop1 = 30
mean_weight_threshold = 1.925

# Calculate the z-score and probability for the first coop
z_score_coop1 = (mean_weight_threshold - mean_coop1) / (std_dev_coop1 / math.
  ↪sqrt(sample_size_coop1))
probability_above_threshold_coop1 = 1 - norm.cdf(z_score_coop1)

# Repeat for the second coop
mean_coop2 = 1.85
std_dev_coop2 = 520
sample_size_coop2 = 30

z_score_coop2 = (mean_weight_threshold - mean_coop2) / (std_dev_coop2 / math.
  ↪sqrt(sample_size_coop2))
probability_above_threshold_coop2 = 1 - norm.cdf(z_score_coop2)

# Repeat for the third coop
mean_coop3 = 2.1
std_dev_coop3 = 380
sample_size_coop3 = 30
```

```
z_score_coop3 = (mean_weight_threshold - mean_coop3) / (std_dev_coop3 / math.
  ↪sqrt(sample_size_coop3))
probability_above_threshold_coop3 = 1 - norm.cdf(z_score_coop3)

print(f"Probability for the first coop: {probability_above_threshold_coop1:.
  ↪4f}")
print(f"Probability for the second coop: {probability_above_threshold_coop2:.
  ↪4f}")
print(f"Probability for the third coop: {probability_above_threshold_coop3:.
  ↪4f}")
```

```
Probability for the first coop: 0.4992
Probability for the second coop: 0.4997
Probability for the third coop: 0.5010
```

Question 4.An engineering team has designed a lamp with two light bulbs. Let X be the lifetime for bulb 1 and Y be the lifetime for bulb 2, both in thousands of hours.Suppose that X and Y are independent and they follow an exponential distribution with mean $\mu = 2$.

(a) What is the probability a bulb lasts more than 2000 hours?

(b) If the lamp works when at least one bulb is lit, what is the probability that the lamp works for more than 2000 hours?

(c) What is the probability that the lamp works no more than 1000 hours?

[32]:
```
# Given mean lifetime
beta = 2

# (a) Probability a bulb lasts more than 2000 hours
prob_a = 1 - expon.cdf(2, scale=beta)

# (b) Probability that the lamp works for more than 2000 hours
prob_b = 1 - expon.cdf(2, scale=beta)**2

# (c) Probability that the lamp works no more than 1000 hours
prob_c = expon.cdf(1, scale=beta)**2

print(f"(a) Probability a bulb lasts more than 2000 hours: {prob_a:.4f}")
print(f"(b) Probability that the lamp works for more than 2000 hours: {prob_b:.
  ↪4f}")
print(f"(c) Probability that the lamp works no more than 1000 hours:{prob_c:.
  ↪4f}")
```

```
(a) Probability a bulb lasts more than 2000 hours: 0.3679
(b) Probability that the lamp works for more than 2000 hours: 0.6004
(c) Probability that the lamp works no more than 1000 hours:0.1548
```

2 Health Insurance

```
[33]: #Google Drive
      from google.colab import drive
      drive.mount('/content/drive',force_remount=True)
```

Mounted at /content/drive

```
[34]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import seaborn as sns
      import datetime as dt
      from scipy import stats
      from scipy.stats import norm
      pd.options.display.float_format = '{:.0f}'.format
```

```
[35]: # Reading the data from csv file
      data=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/CRA/insurance.csv')
      data.head()
```

```
[35]:    age     sex  bmi  children smoker     region  charges
      0   19  female   28         0    yes  southwest    16885
      1   18    male   34         1     no  southeast     1726
      2   28    male   33         3     no  southeast     4449
      3   33    male   23         0     no  northwest    21984
      4   32    male   29         0     no  northwest     3867
```

2.1 Dataset Description

This dataset can be helpful in a simple yet illuminating study in understanding the risk underwriting in Health Insurance, the interplay of various attributes of the insured and see how they affect the insurance premium. This dataset contains 1338 rows of insured data, where the Insurance charges are given against the following attributes of the insured: Age, Sex, BMI, Number of Children, Smoker and Region. There are no missing or undefined values in the dataset.

```
[36]: # Get Data Summary
      def GetSummary(text,df):
              print(f'{text} shape: {df.shape}')
              summ = pd.DataFrame(df.dtypes, columns=['dtypes'])
              summ['null'] = df.isnull().sum()
              summ['unique'] = df.nunique()
              summ['min'] = df.min()
              summ['median'] = df.median()
              summ['max'] = df.max()
              summ['mean'] = df.mean()
              summ['std'] = df.std()
              summ['duplicate'] = df.duplicated().sum()
```

```
            return summ
```

[37]:
```
# Get summary data
GetSummary('data',data)
```

data shape: (1338, 7)

[37]:
	dtypes	null	unique	min	median	max	mean	std \
age	int64	0	47	18	39	64	39	14
sex	object	0	2	female	NaN	male	NaN	NaN
bmi	float64	0	548	16	30	53	31	6
children	int64	0	6	0	1	5	1	1
smoker	object	0	2	no	NaN	yes	NaN	NaN
region	object	0	4	northeast	NaN	southwest	NaN	NaN
charges	float64	0	1337	1122	9382	63770	13270	12110

	duplicate
age	1
sex	1
bmi	1
children	1
smoker	1
region	1
charges	1

[38]:
```
# Drop duplicate data
data.drop_duplicates(inplace = True)
GetSummary('data',data)
```

data shape: (1337, 7)

[38]:
	dtypes	null	unique	min	median	max	mean	std \
age	int64	0	47	18	39	64	39	14
sex	object	0	2	female	NaN	male	NaN	NaN
bmi	float64	0	548	16	30	53	31	6
children	int64	0	6	0	1	5	1	1
smoker	object	0	2	no	NaN	yes	NaN	NaN
region	object	0	4	northeast	NaN	southwest	NaN	NaN
charges	float64	0	1337	1122	9386	63770	13279	12110

	duplicate
age	0
sex	0
bmi	0
children	0
smoker	0
region	0

```
charges              0
```

```python
[39]:  # Define variable type
       # (a).Plot the histograms of primary (important) continuous variables and␣
        ↪probability distributions of categorical variables.

       continuous= ['bmi','charges']
       # Plot histograms for continuous variables
       fig, axes = plt.subplots(nrows=len(continuous), ncols=1, figsize=(12, 10))
       for i, variable in enumerate(continuous):
           sns.kdeplot(x=variable, data=data, ax=axes[i],color='lightblue',fill=True)
           axes[i].set_title(f'Distribution of {variable}')
       plt.show()
```
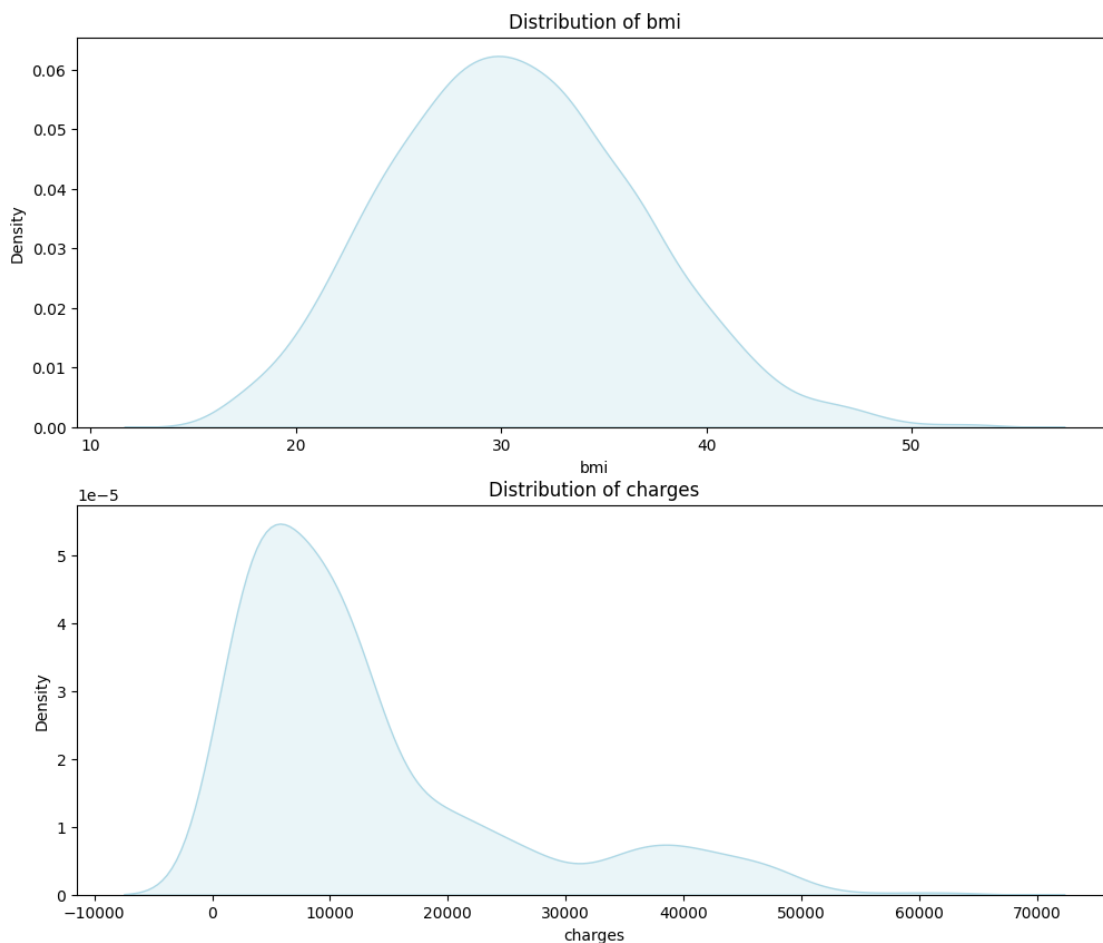


```python
[40]:  # Plot probability distributions for categorical variables
       categorical  = ['sex','children','smoker','region']
```

```python
fig, axes = plt.subplots(nrows=len(categorical), ncols=1, figsize=(5, 20))

for i, variable in enumerate(categorical):
    # Calculate the relative frequencies (probabilities) for each category
    counts = data[variable].value_counts(normalize=True)

    # Plot the bar plot
    sns.barplot(x=counts.index, y=counts.values, ax=axes[i],color='lightblue')
    axes[i].set_title(f'Probability Distribution of {variable}')
    axes[i].set_xlabel(variable)
    axes[i].set_ylabel('Probability')

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

Probability Distribution of sex

Probability Distribution of children

Probability Distribution of smoker

Probability Distribution of region

9
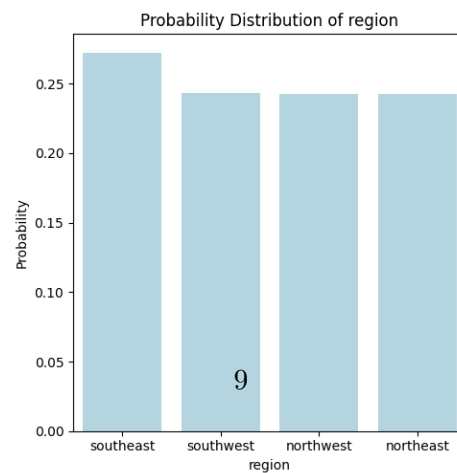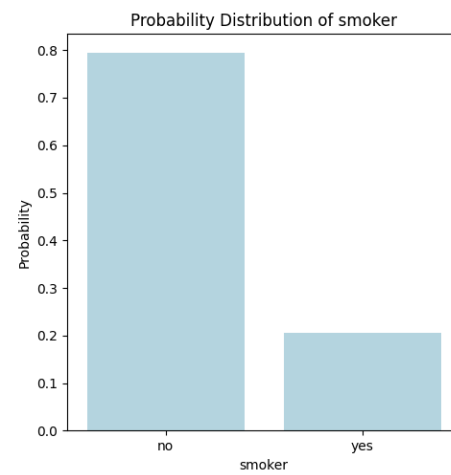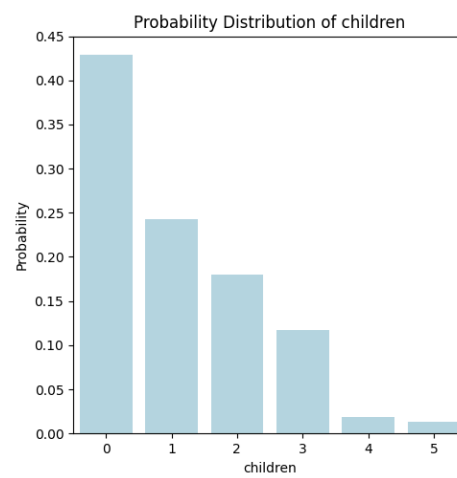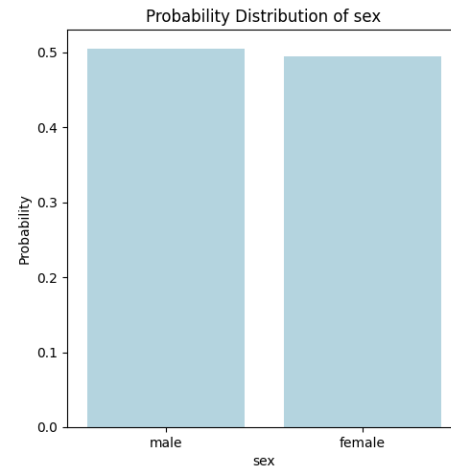
[41]:
```
# (b).Plot the box plots of all the primary variables in the data. Identify and
    ↪delete a couple of extreme outliers from the data if there are any.
plt.figure(figsize= (12,10))
plt.subplot(3,1,1)
sns.boxplot(x= data.bmi, color='lightblue')
plt.title('BoxPlot of BMI')
plt.xlabel('BMI')

plt.subplot(3,1,2)
sns.boxplot(x= data.age, color='lightblue')
plt.title('BoxPlot of Age')
plt.xlabel('Age')

plt.subplot(3,1,3)
sns.boxplot(x= data.charges, color='lightblue')
plt.title('BoxPlot of Charges')
plt.xlabel('Charges')
plt.tight_layout(pad=3.5)
plt.show()
```

BoxPlot of BMI

BoxPlot of Age

BoxPlot of Charges
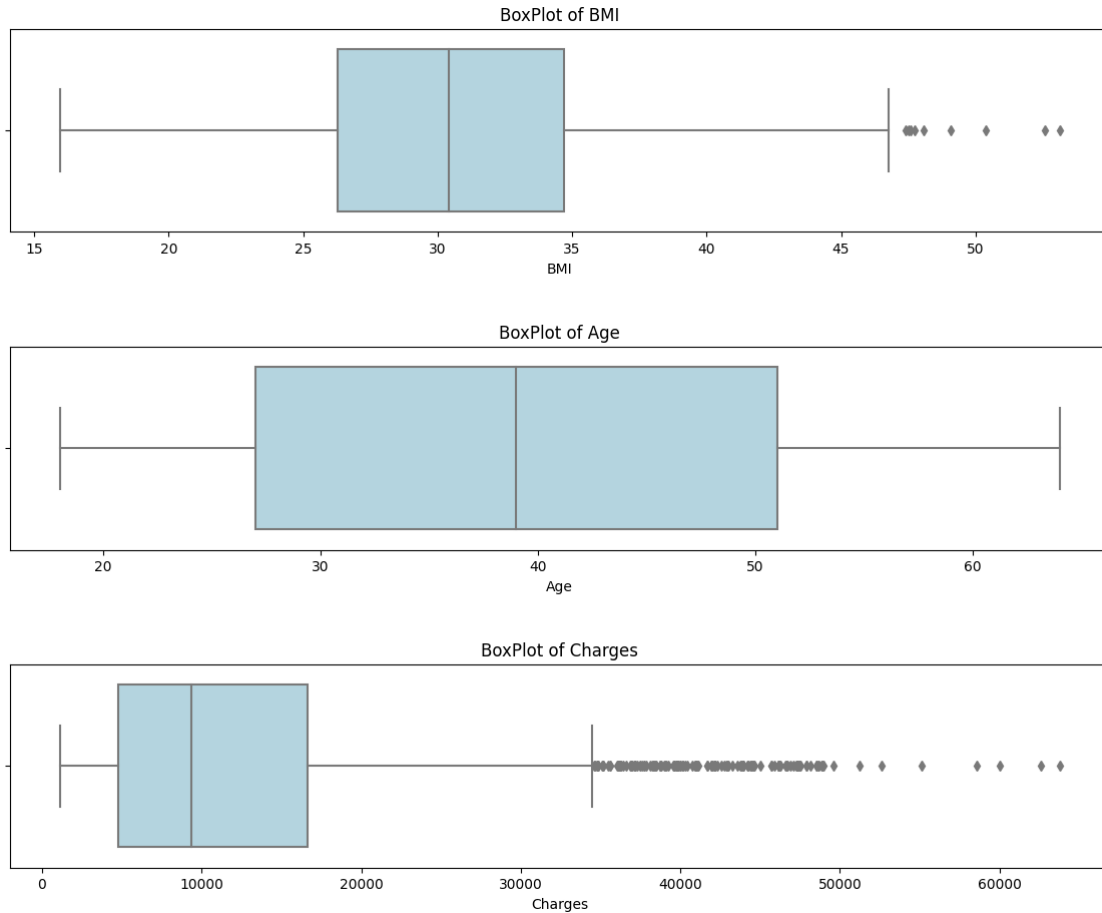

```python
[42]: # Detect outlier based on quantiles
      def detect_outliers_iqr(column, threshold=1.5):
          q1 = column.quantile(0.25)
          q3 = column.quantile(0.75)
          iqr = q3 - q1
          lower_bound = q1 - threshold * iqr
          upper_bound = q3 + threshold * iqr
          outliers = (column < lower_bound) | (column > upper_bound)
          return outliers

      # Select the specific columns you want to test for outliers
      columns_to_test = ['age', 'bmi', 'charges']
      outliers = data[columns_to_test].apply(detect_outliers_iqr)

      print("Number of outliers in each selected column:")
      print(outliers.sum())
```

```
Number of outliers in each selected column:
age          0
```

```
bmi           9
charges     139
dtype: int64
```

[43]:
```python
# Delete the data from dataset after removing outliers
df_cleaned = data[~outliers.any(axis=1)]
print("Original DataFrame shape:", data.shape)
print("Cleaned DataFrame shape:", df_cleaned.shape)
```

```
Original DataFrame shape: (1337, 7)
Cleaned DataFrame shape: (1192, 7)
```

[44]:
```python
# (c).Compute the matrix of sample correlations between every pair of variables.
numeric_df = data.select_dtypes(include=['number'])
# Compute the matrix of sample correlations
correlation_matrix = data.corr()
# Plot a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',
    ↪linewidths=0.5)
plt.title('Correlation Matrix of Insurance')
plt.show()
```

Correlation Matrix of Insurance
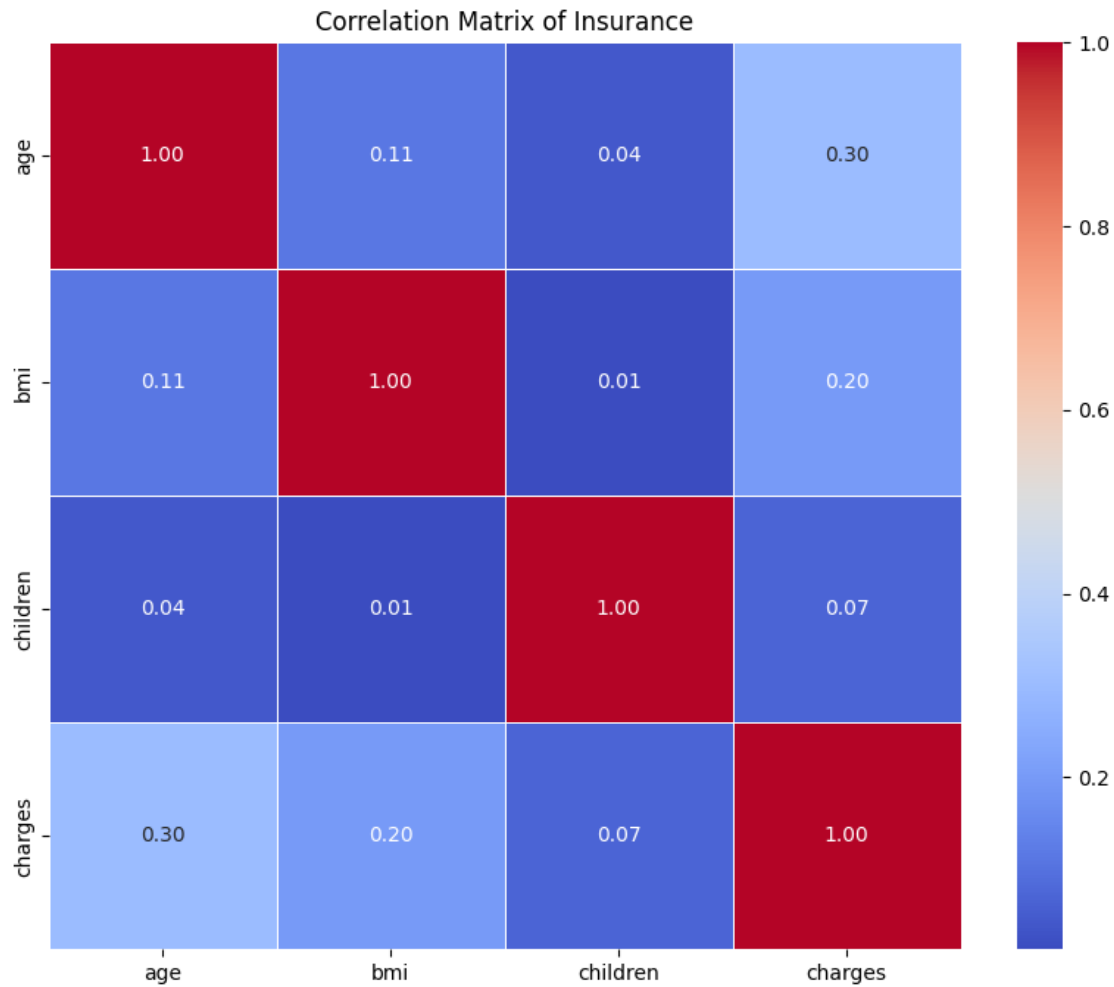
[45]:
```
# (d).Choose two variables that are highly correlated (positively or␣
 ↪negatively) and plot their scatter plot and the best regression line
numeric_df = data.select_dtypes(include=['number'])
# Compute the matrix of sample correlations
correlation_matrix = numeric_df.corr()
# Create a scatter plot with the best-fitting regression line
plt.figure(figsize=(10, 8))
sns.regplot(x=data.age, y=data.charges, data=data, scatter_kws={'s': 20,␣
 ↪'alpha': 0.7}, line_kws={'color': 'red'})
plt.title(f'Scatter Plot Betweeen Age and Charges')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.show()
```
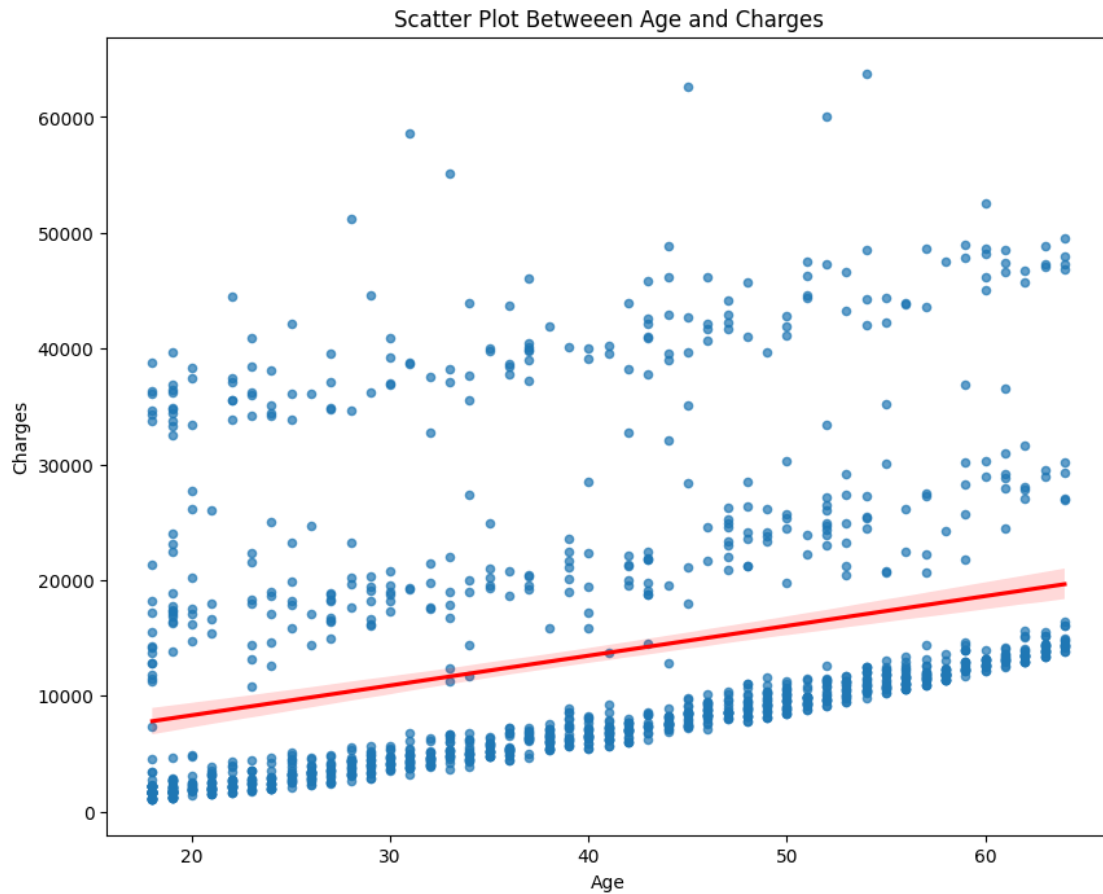
Scatter Plot Betweeen Age and Charges

```
numeric_df = data.select_dtypes(include=['number'])
# Compute the matrix of sample correlations
correlation_matrix = numeric_df.corr()
# Create a scatter plot with the best-fitting regression line
plt.figure(figsize=(10, 8))
sns.regplot(x=data.age, y=data.bmi, data=data, scatter_kws={'s': 20, 'alpha': 0.
 ⇢7}, line_kws={'color': 'red'})
plt.title(f'Scatter Plot Betweeen Age and Bmi')
plt.xlabel('Age')
plt.ylabel('Bmi')
plt.show()
```

Scatter Plot Betweeen Age and Bmi
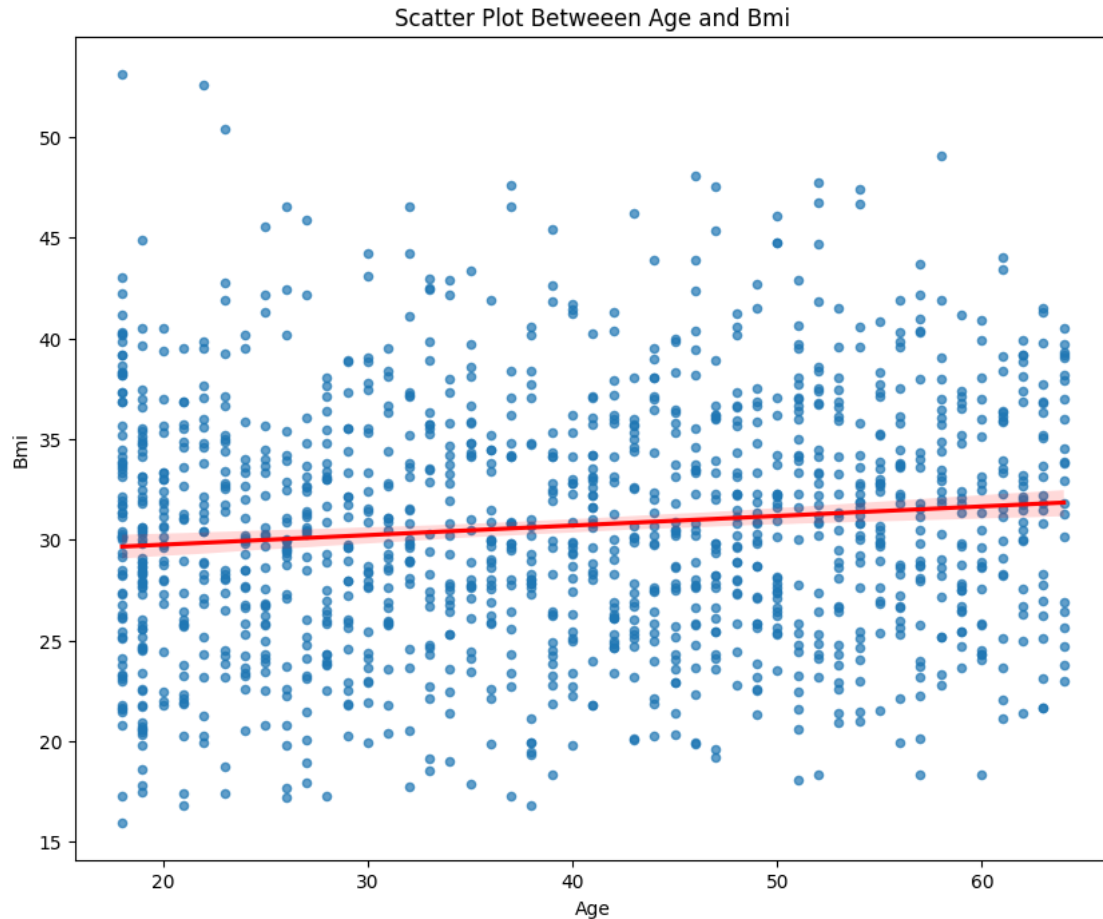
(4). Write a brief description of your data and summarize your findings on the variables.

2.2 Observations

- Age shows uniform distribution.
- Bmi shows normal distribution.
- Charges shows normal distribution, but it skewed to right.
- BMI shows 9 row of outliers and charges show 139 rwo outliers.
- Many customers are no smoke.
- Sex distribution is relatively even.
- Male smokers are more than the female smoker.
- Female no smoker are more than male no smoker.
- Region distribution is fairly balanced, with the exception of the Southeast which has the highest number of customers.
- Many customers have one children and fewer customers have more than 4 childrens.
- As customer age increases, charges also tend to rise.
- Two customer groups exist: smokers and non-smokers
- Smoker customers incur higher charges compared to non-smokers.
- As customer age increases, charges also tend to rise/

- Typically, the highest charges are from male and female smoker customers.
- The lowest charges are from male and female no smoker customers.

```
[ ]: %%capture
     !pip install nbconvert
     !sudo apt-get install texlive-xetex texlive-fonts-recommended␣
      ↪texlive-plain-generic
```

```
[ ]: # https://saturncloud.io/blog/convert-google-colab-notebook-to-pdf-html/
     !jupyter nbconvert  '/content/drive/MyDrive/Colab Notebooks/
      ↪STAT603-Fall2023-FinalExam.ipynb' --to pdf
```

THE HONOR PLEDGE

"I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagiarism."

I pledge that I have neither given nor received any assistance on this final exam. The submitted work is my own.

Signed _Ashish Verma_

Date _12/11/2023._