Learning from data Assignment

April 10, 2024

[14]:
```
import pandas as ol
import csv
import math
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
```

[15]:
```
with open(r"C:\Users\Laptop\OneDrive\Documents\Shopping_data.csv", 'r') as f:
 csv_reader = csv.reader(f, delimiter=";")
 data_list = [row for row in csv_reader if row]
column_names = data_list[0]
df = ol.DataFrame(data_list[1:], columns=column_names)
print(df)

#Notes
#This data is used in a supermarket
#This dataset is based on a supermarket where consumer's gender, annual income␣
 ↪are recorded into the dataset.
#The spending score(1 to 100) is base on how much the consumer spends on␣
 ↪groceries in the supermarket.
#In this dataset there are 88 males and 112 females and with the statistics␣
 ↪recorded, the bar graph in the workbook shows
#that females have a higher spending score as opposed to the male.
#In the workbook the lowest value in the dataset is 1 which is a male and the␣
 ↪highest is 99 which is female.
#What the dataset is trying to show is that females in the data do more␣
 ↪shopping in the supermarket as there are more females than males
#and the sum of the spending score is higher than males.
```

```
    CustomerID   Genre Age Annual Income (k$) Spending Score (1-100)
0            1    Male  19                 15                     39
1            2    Male  21                 15                     81
2            3  Female  20                 16                      6
3            4  Female  23                 16                     77
4            5  Female  31                 17                     40
..         …     …  ..                …                    … ..
195        196  Female  35                120                     79
196        197  Female  45                126                     28
```

```
197        198    Male   32                 126                      74
198        199    Male   32                 137                      18
199        200    Male   30                 137                      83

[200 rows x 6 columns]
```

[16]:
```
#Mean
sum_age = ol.to_numeric(df['Age'].str.replace(',', ''), errors='coerce').sum()
sum_income =  ol.to_numeric(df['Annual Income (k$)'].str.replace(',', ''),␣
↪errors='coerce').sum()
sum_spendscore = ol.to_numeric(df['Spending Score (1-100)'].str.replace(',',␣
↪''), errors='coerce').sum()


count_age = ol.to_numeric(df['Age'].str.replace(',', ''), errors='coerce').
↪count()
count_income = ol.to_numeric(df['Annual Income (k$)'].str.replace(',', ''),␣
↪errors='coerce').count()
count_spendscore = ol.to_numeric(df['Spending Score (1-100)'].str.replace(',',␣
↪''), errors='coerce').count()


mean_age = sum_age / count_age
mean_income = sum_income / count_income
mean_spendscore = sum_spendscore / count_spendscore


print("Mean Age ", mean_age)
print("Mean Income ", mean_income)
print("Mean Spending Score ", mean_spendscore)
```

```
Mean Age  38.85
Mean Income  60.56
Mean Spending Score  50.2
```

[30]:
```
#Median

df[['Age', 'Annual Income (k$)','Spending Score (1-100)']] = df[['Age', 'Annual␣
↪Income (k$)','Spending Score (1-100)']].apply(ol.to_numeric, errors='coerce')


df_sorted = df.sort_values(by=['Age', 'Annual Income (k$)', 'Spending Score␣
↪(1-100)'])
median_index = int((len(df_sorted) - 1) / 2)


median_age = df_sorted['Age'].iloc[median_index]
```

```
median_income = df_sorted['Annual Income (k$)'].iloc[median_index]
median_spendscore = df_sorted['Spending Score (1-100)'].iloc[median_index]


print("Median Age:", median_age)
print("Median Income:", median_income)
print("Median Spend Score:", median_spendscore)
```

```
Median Age: 36
Median Income: 85
Median Spend Score: 75
```

[31]:
```
#Mode
df[['Age', 'Annual Income (k$)']] = df[['Age', 'Annual Income (k$)']].apply(ol.
 ↪to_numeric, errors='coerce')
value_counts = {}
for col in df.columns:
  value_counts[col] = df[col].value_counts().sort_values(ascending=False)


modes = []
for col, counts in value_counts.items():
  modes.append(counts.index[0])


#print("Mode Age:", modes[0])
#print("Mode Income:", modes[1])
print("Mode Spending Score:", modes[2])
```

```
Mode Spending Score: 32
```

[32]:
```
#Geometric Mean of Spending Score
df[['Age', 'Annual Income (k$)']] = df[['Age', 'Annual Income (k$)']].apply(ol.
 ↪to_numeric, errors='coerce')
product = 1
for score in df['Spending Score (1-100)']:
    product *= score
geometric_mean_score = math.exp(math.log(product)/ len(df))

print("Geometric Mean of Spending Score:", geometric_mean_score)
```

```
Geometric Mean of Spending Score: 39.921161228635
```

[33]:
```
#80th Percentile
df[['Age', 'Annual Income (k$)']] = df[['Age', 'Annual Income (k$)']].apply(ol.
 ↪to_numeric, errors='coerce')
income80th = df['Annual Income (k$)'].quantile(0.8)
score80th = df['Spending Score (1-100)'].quantile(0.8)
```

```python
print("80th Percentile of Annual Income (k$):", income80th)
print("80th Percentile of Spending Score:", score80th)
```

```
80th Percentile of Annual Income (k$): 78.20000000000002
80th Percentile of Spending Score: 75.0
```

[34]:
```python
#Third Quartile

income75th = df['Annual Income (k$)'].quantile(0.75)
score75th = df['Spending Score (1-100)'].quantile(0.75)
print("Third Quartile of Annual Income (k$):", income75th)
print("Third Quartile of Spending Score:", score75th)
```

```
Third Quartile of Annual Income (k$): 78.0
Third Quartile of Spending Score: 73.0
```

[35]:
```python
#First Quartile

income25th = df['Annual Income (k$)'].quantile(0.25)
score25th = df['Spending Score (1-100)'].quantile(0.25)
print("First Quartile of Annual Income (k$):", income25th)
print("First Quartile of Spending Score:", score25th)
```

```
First Quartile of Annual Income (k$): 41.5
First Quartile of Spending Score: 34.75
```

[36]:
```python
#Range

min_age= df['Age'].min()
max_age= df['Age'].max()
range_age = max_age - min_age + 1

min_income= df['Annual Income (k$)'].min()
max_income= df['Annual Income (k$)'].max()
range_income = max_income - min_income + 1

min_score= df['Spending Score (1-100)'].min()
max_score= df['Spending Score (1-100)'].max()
range_score = max_score - min_score + 1
print("Range of age: ", range_age)
print("Range of income: ", range_income)
print("Range of spending Score: ", range_score)
```

```
Range of age:  53
Range of income:  123
Range of spending Score:  99
```

[37]:
```python
#Interquartile range
iqr_income = income75th - income25th
```

```
iqr_score = score75th - score25th

print("Interquartile range of Annual Income:", iqr_income)
print("Interquartile range of Spending Score:", iqr_score)
```

```
Interquartile range of Annual Income: 36.5
Interquartile range of Spending Score: 38.25
```

[38]:
```
#Variance

df[['Age', 'Spending Score (1-100)']] = df[['Age', 'Spending Score (1-100)']].
 ↪apply(ol.to_numeric, errors='coerce')

mean_age = df['Age'].mean()
mean_score = df['Spending Score (1-100)'].mean()

squared_deviations_age = (df['Age'] - mean_age) ** 2
squared_deviations_score = (df['Spending Score (1-100)'] - mean_score) ** 2

population_variance_age = squared_deviations_age.sum() / len(df)
population_variance_score = squared_deviations_score.sum() / len(df)

print("Population variance of Age:", population_variance_age)
print("Population variance of Spending Score:", population_variance_score)
```

```
Population variance of Age: 194.1575
Population variance of Spending Score: 663.52
```

[39]:
```
#Standard deviation

squared_deviations_age = (df['Age'] - mean_age) ** 2
squared_deviations_income = (df['Annual Income (k$)'] - mean_income) ** 2
squared_deviations_score = (df['Spending Score (1-100)'] -  mean_spendscore) **␣
 ↪2

population_variance_age = squared_deviations_age.sum() / len(df)
population_variance_income = squared_deviations_income.sum() / len(df)
population_variance_score = squared_deviations_score.sum() / len(df)

standard_deviation_age = np.sqrt(population_variance_age)
standard_deviation_income = np.sqrt(population_variance_income)
standard_deviation_score = np.sqrt(population_variance_score)

print("Standard deviation Age:", standard_deviation_age)
print("Standard deviation Income:", standard_deviation_income)
print("Standard deviation Spending Score:", standard_deviation_score)
```

```
Standard deviation Age: 13.934041050606963
Standard deviation Income: 26.19897707926781
```

Standard deviation Spending Score: 25.7588819633151

[40]:
```
#Coefficient of variation

income_array = df['Annual Income (k$)'].to_numpy()
income_filtered = income_array[~np.isnan(income_array)]

if len(income_filtered) > 0:
    cv_income = stats.variation(income_filtered)
    print("Coefficient of Variation of Annual Income (k$):", cv_income)
else:
    print("Cannot calculate CV: All values in 'Annual Income (k$)' are missing␣
↪or NaN")
```

Coefficient of Variation of Annual Income (k$): 0.43261190685713025

[41]:
```
#Weighted Mean
weighted_mean_amount = 0
total_weight = 0

for i in range(len(df)):
    income_numeric = ol.to_numeric(df.loc[i, 'Annual Income (k$)'],␣
↪errors='coerce')
    customer_id_numeric = ol.to_numeric(df.loc[i, 'CustomerID'],␣
↪errors='coerce')  # No comma replacement needed

    weighted_mean_amount += income_numeric * customer_id_numeric
    total_weight += customer_id_numeric


print("Weighted mean is:", total_weight/20100)
```

Weighted mean is: 1.0

[42]:
```
#z-Score of lowest value

z_score = (min_score- mean_spendscore)/ standard_deviation_score

print("Z-score of the lowest value:", z_score)
```

Z-score of the lowest value: -1.9100207870073291

[43]:
```
#Outliers
lower_score= score25th - 1.5 * iqr_score
upper_score= score25th + 1.5 * iqr_score

lower_income= income25th - 1.5 * iqr_income
upper_income= income75th + 1.5 * iqr_income
```

```
print("Upper fence (Annual income):", upper_income)
print("Lower fence (Annual income):", lower_income)

print("Upper fence (Spending Score):", upper_score)
print("Lower fence (Spending Score):", lower_score)
```

```
Upper fence (Annual income): 132.75
Lower fence (Annual income): -13.25
Upper fence (Spending Score): 92.125
Lower fence (Spending Score): -22.625
```

[44]:

```
#Covariance
import numpy as np


income_array = df['Annual Income (k$)'].to_numpy()
spending_array = df['Spending Score (1-100)'].to_numpy()

covariance = np.cov(income_array, spending_array)[0][1]


covariance_matrix = np.cov(income_array, spending_array)

print("Covariance between Annual Income and Spending Score:", covariance)
```

```
Covariance between Annual Income and Spending Score: 6.716582914572865
```

[49]:

```
#Correlation
if 'Genre' not in df.columns:
    print("Column 'Genre' not found in DataFrame")
else:
    correlation_matrix = df.corr()
    #df_filtered = df.drop('Genre', axis=1)
    income_correlation = correlation_matrix['Annual Income (k$)']['Spending␣
 ↪Score (1-100)']

print("Correlation coefficient between Annual Income and Spending Score:",␣
 ↪income_correlation)


plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'])
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score vs. Annual Income')
plt.show()
```

```
Correlation coefficient between Annual Income and Spending Score:
```

0.009902848094037497

[46]:

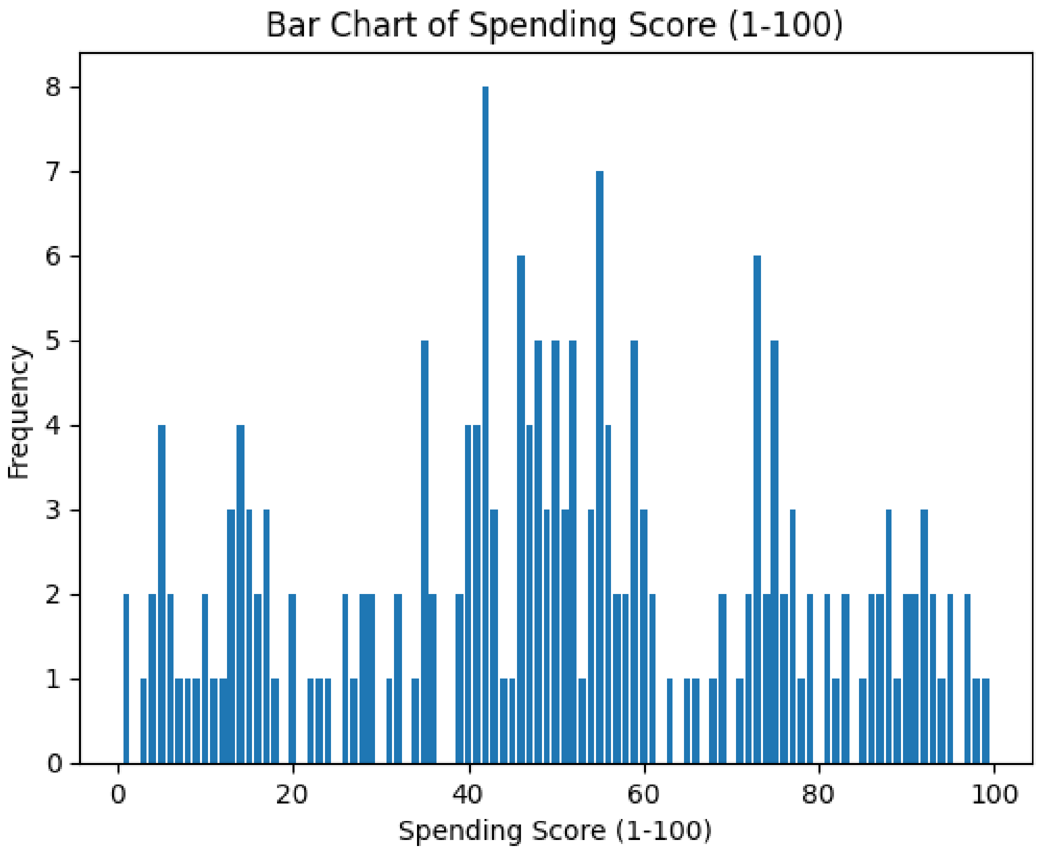
```
#Chebyshev's theorem
df = df.apply(ol.to_numeric, errors='coerce')
k = 1
means = df.mean()
squared_deviations = df.sub(means, axis=0).pow(2)
population_variances = squared_deviations.mean(axis=0)
chebyshev_proportions = 1 - 1 / (k**2)
print("Chebyshev's Theorem (", k, " standard deviations):")
for col, mean in means.items():
  variance = population_variances[col]
  std_dev = np.sqrt(variance)
  proportion = 1 - 1 / (k**2)
  print(f"  Column: {col}")
  print(f"    Minimum Proportion Within {k} Std Devs: {proportion:.2f}")
  print(f"    Expected Range: {mean:.2f} +/- {k * std_dev:.2f}")
```

```
Chebyshev's Theorem ( 1  standard deviations):
  Column: CustomerID
    Minimum Proportion Within 1 Std Devs: 0.00
```

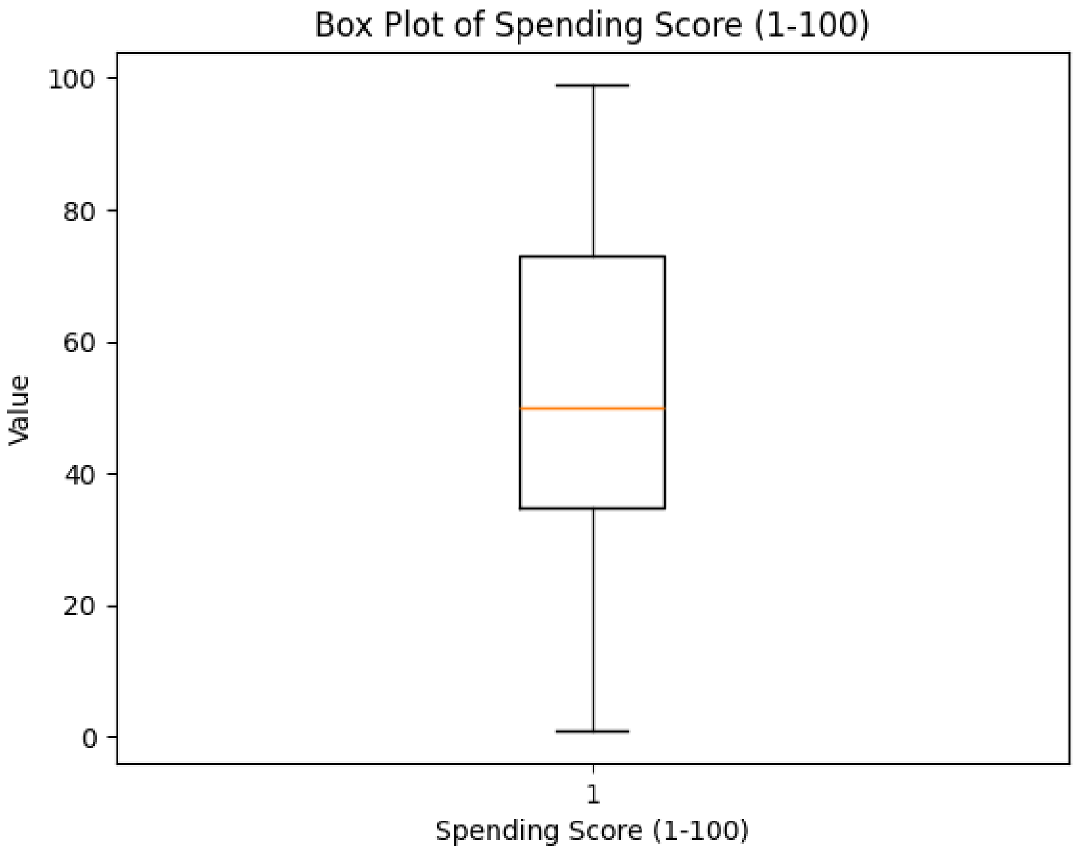
```
  Expected Range: 100.50 +/- nan
Column: Genre
  Minimum Proportion Within 1 Std Devs: 0.00
  Expected Range: nan +/- nan
Column: Age
  Minimum Proportion Within 1 Std Devs: 0.00
  Expected Range: 38.85 +/- nan
Column: Annual Income (k$)
  Minimum Proportion Within 1 Std Devs: 0.00
  Expected Range: 60.56 +/- nan
Column: Spending Score (1-100)
  Minimum Proportion Within 1 Std Devs: 0.00
  Expected Range: 50.20 +/- nan
Column:
  Minimum Proportion Within 1 Std Devs: 0.00
  Expected Range: nan +/- nan
```

[46]:

```
#Bar Chart
column_to_analyze = 'Spending Score (1-100)'
plt.bar(df[column_to_analyze].value_counts().index, df['Spending Score (1-100)'␣
 ↪].value_counts().values)
plt.xlabel(column_to_analyze)
plt.ylabel('Frequency')
plt.title('Bar Chart of ' + column_to_analyze)
plt.show()
```



[48]:
```
#Box plot
plt.boxplot(df[column_to_analyze])
plt.xlabel(column_to_analyze)
plt.ylabel('Value')
plt.title('Box Plot of ' + column_to_analyze)
plt.show()
```

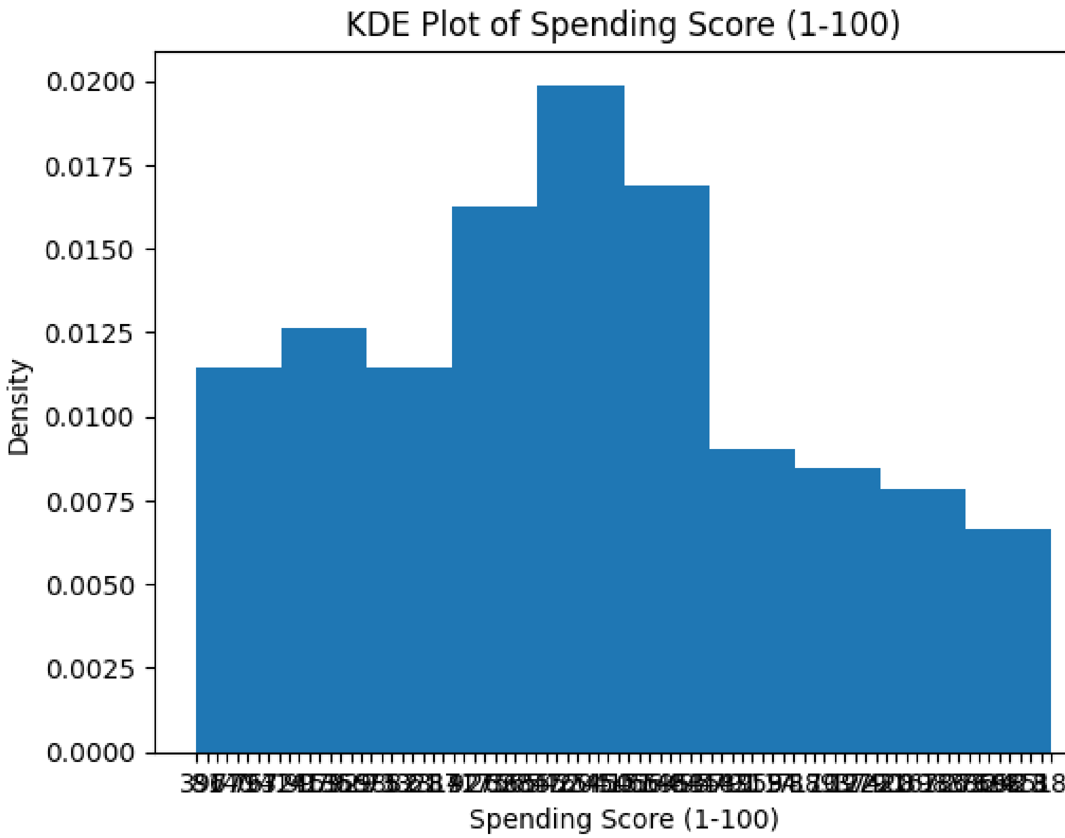


[49]:
```
#Histogram
plt.hist(df[column_to_analyze])
plt.xlabel(column_to_analyze)
plt.ylabel('Frequency')
plt.title('Histogram of ' + column_to_analyze)
plt.show()

#In the histogram graph it shows that the most spending score in a supermarket␣
↪is between 40 to 60.
```

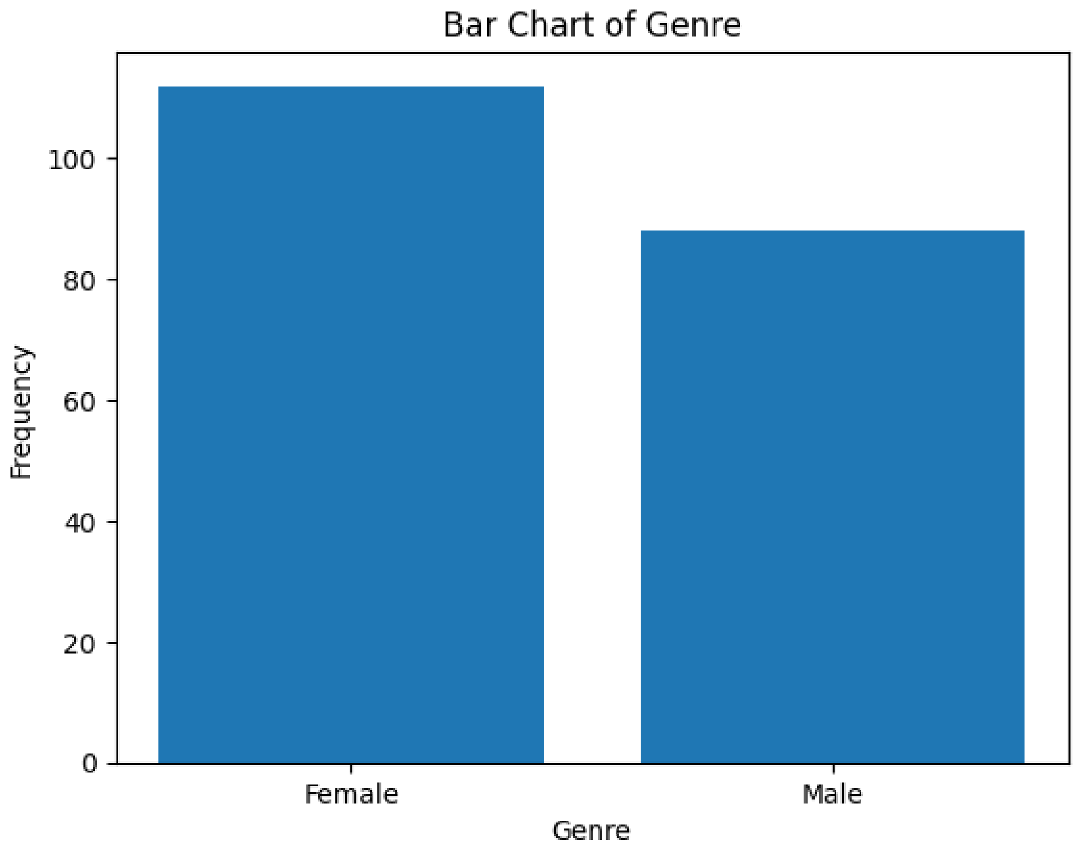
[59]:
```
#KDE plot graph

plt.hist(df[column_to_analyze], density=True)
plt.xlabel(column_to_analyze)
plt.ylabel('Density')
plt.title('KDE Plot of ' + column_to_analyze)
plt.show()
```



[60]:

```
#Bar Graph gender
column_to_analyze = 'Genre'
plt.bar(df[column_to_analyze].value_counts().index, df[column_to_analyze].
 ↪value_counts().values)
plt.xlabel(column_to_analyze)
plt.ylabel('Frequency')
plt.title('Bar Chart of ' + column_to_analyze)
plt.show()
```



[58]:

```
#Reflection

#My thoughts on this assignment have shown me that my knowledge of Python is␣
↪still lacking,
#as I struggled when using the descriptive statistics on the dataset and trying␣
↪to make the code run properly without errors.
#Using descriptive statistics on a large dataset can have challenges because␣
↪when you use methods such as variance or coefficient you have to ensure that␣
↪the results make sense,
#I think it shows that picking a dataset to work with is important because the␣
↪dataset chosen can work well with python but not so much with excel as␣
↪python can work with different data types.
#When using Python, it is easier to create custom functions compared to Excel,␣
↪and Python is much better to use when you have a large dataset to work with.
#After working on the assignment what I would do differently is choose a␣
↪different dataset or add more descriptive data to the dataset that was used.
#I would use Python when working with a large dataset rather than Excel and␣
↪Python work well with diverse datasets.
```

```
#For Excel I would use it for a very simple dataset otherwise for data that is␣
 ↪more complex I would use Python.
#Another difficulty I found when using Python was integrating graphs into the␣
 ↪notebook.
```

```
[ ]:
```