21. Scenario: You are working on a dataset that contains information about various types of fruits. The dataset includes features such as weight, color, and texture of the fruit. Your task is to build a k-Nearest Neighbors (kNN) classifier to predict the type of fruit based on these features.

Question: Given a dataset with features like 'weight', 'color', and 'texture' of fruits, and their respective 'type' (e.g., apple, orange, banana), how would you implement a k-Nearest Neighbors classifier to predict the type of an unknown fruit based on its 'weight', 'color', and 'texture' features? Additionally, discuss the process of choosing the optimal value of 'k' and handling categorical features (like 'color' or 'type') in a kNN classifier.

## CODE:

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Generate a synthetic dataset (replace this with your actual dataset)

data = {

'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'TotalAmountSpent': [100, 300, 150, 500, 200, 600, 350, 800, 400, 700],

'FrequencyOfVisits': [2, 4, 3, 5, 2, 6, 4, 8, 3, 7]

}

df = pd.DataFrame(data)

# Select features for clustering

X = df[['TotalAmountSpent', 'FrequencyOfVisits']]

# Standardize the features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Choose the number of clusters (you may need to tune this based on your data)

num\_clusters = 3

# Build the K-Means model

kmeans\_model = KMeans(n\_clusters=num\_clusters, random\_state=42)

df['Cluster'] = kmeans\_model.fit\_predict(X\_scaled)

# Visualize the clusters

plt.scatter(df['TotalAmountSpent'], df['FrequencyOfVisits'], c=df['Cluster'], cmap='viridis')

plt.xlabel('Total Amount Spent')

plt.ylabel('Frequency of Visits')

plt.title('Customer Segmentation with K-Means Clustering')

plt.show()

# Display the cluster centers

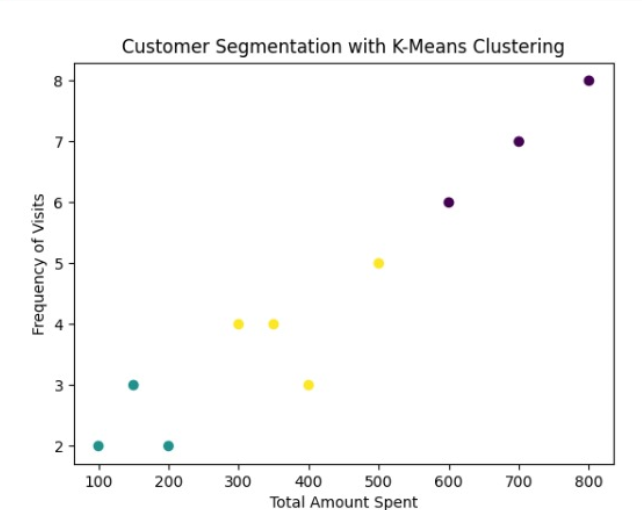
cluster\_centers = scaler.inverse\_transform(kmeans\_model.cluster\_centers\_)

cluster\_centers\_df = pd.DataFrame(cluster\_centers, columns=['TotalAmountSpent', 'FrequencyOfVisits'])

print("Cluster Centers:")

print(cluster\_centers\_df)

### OUTPUT:



22. Scenario: You are tasked with implementing a decision tree classifier in Python to predict whether an online shopper will make a purchase on an e-commerce platform. The dataset provided includes attributes such as 'age', 'income', 'browsing\_duration', 'device\_type', and the target variable 'purchase' (indicating whether a purchase was made or not).

Question: Given the dataset with the mentioned attributes and the 'purchase' label, how would you use Python's scikit-learn library to create a decision tree classifier? Provide code to preprocess categorical variables like 'device\_type' for model training and predict whether a new customer, with specific 'age', 'income', 'browsing\_duration', and 'device\_type', is likely to make a purchase or not.

## CODE:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import LabelEncoder

data = {

'age': [25, 30, 35, 40, 45, 28, 32, 38, 42, 48],

'income': [50000, 60000, 70000, 80000, 90000, 55000, 65000, 75000, 85000, 95000],

'browsing\_duration': [15, 20, 25, 30, 35, 18, 22, 28, 33, 40],

'device\_type': ['Mobile', 'Desktop', 'Mobile', 'Desktop', 'Mobile', 'Desktop', 'Mobile', 'Desktop', 'Mobile', 'Desktop'],

'purchase': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]

}

df = pd.DataFrame(data)

label\_encoder = LabelEncoder()

df['device\_type'] = label\_encoder.fit\_transform(df['device\_type'])

X = df[['age', 'income', 'browsing\_duration', 'device\_type']]

y = df['purchase']

# 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, random\_state=42)

# Create a Decision Tree classifier

decision\_tree\_model = DecisionTreeClassifier(random\_state=42)

# Train the model

decision\_tree\_model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = decision\_tree\_model.predict(X\_test)

# Evaluate model performance (optional)

from sklearn.metrics import accuracy\_score, classification\_report

accuracy = accuracy\_score(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:\n", classification\_rep)

# Now, let's predict whether a new customer will make a purchase or not

new\_customer\_data = pd.DataFrame({

'age': [30],

'income': [60000],

'browsing\_duration': [25],

'device\_type': ['Mobile'] # Replace with the actual device type of the new customer

})

# Convert 'device\_type' using label encoding

new\_customer\_data['device\_type'] = label\_encoder.transform(new\_customer\_data['device\_type'])

# Make predictions for the new customer

prediction = decision\_tree\_model.predict(new\_customer\_data)

# Display the prediction

print("\nPrediction for the new customer:")

print("Will make a purchase" if prediction[0] == 1 else "Will not make a purchase")

### OUTPUT:

Accuracy: 1.00

Classification Report:

precision recall f1-score support

0 1.00 1.00 1.00 1

1 1.00 1.00 1.00 1

accuracy 1.00 2

macro avg 1.00 1.00 1.00 2

weighted avg 1.00 1.00 1.00 2

Prediction for the new customer:

Will make a purchase

23. Scenario: You work for a financial institution, and your task is to develop a classification model to assess the credit risk associated with loan applicants. The dataset provided contains various attributes such as income, credit score, debt-to-income ratio, employment duration, and the final 'risk' label indicating whether an applicant is high-risk or low-risk for a loan.

Question: Using the Classification and Regression Trees (CART) algorithm in Python, build a predictive model to evaluate the credit risk of loan applicants based on features like 'income', 'credit score', 'debt-to-income ratio', and 'employment duration'? Provide a Python code that preprocesses the data, builds a CART classifier, and predicts the credit risk level for a new loan applicant with specific attribute values

## CODE:

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.tree import export\_text

# Create a sample dataset

data = pd.DataFrame({

'income': [50000, 80000, 60000, 75000, 90000, 55000, 72000, 85000],

'credit\_score': [600, 750, 700, 720, 780, 620, 700, 760],

'debt\_to\_income\_ratio': [0.4, 0.2, 0.3, 0.2, 0.1, 0.5, 0.3, 0.2],

'employment\_duration': [1, 3, 2, 2, 4, 1, 2, 3],

'risk': ['high', 'low', 'low', 'low', 'low', 'high', 'low', 'low']

})

# Display the first few rows of the dataset to understand its structure

print(data)

# Preprocess the data

X = data[['income', 'credit\_score', 'debt\_to\_income\_ratio', 'employment\_duration']]

y = data['risk']

# 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, random\_state=42)

# Build the CART classifier

classifier = DecisionTreeClassifier(random\_state=42)

classifier.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = classifier.predict(X\_test)

# Evaluate the model performance

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Model Accuracy: {accuracy \* 100:.2f}%')

# Display the decision tree rules

tree\_rules = export\_text(classifier, feature\_names=list(X.columns))

print("Decision Tree Rules:\n", tree\_rules)

# Example: Predict credit risk for a new loan applicant

new\_applicant = pd.DataFrame({

'income': [60000],

'credit\_score': [700],

'debt\_to\_income\_ratio': [0.3],

'employment\_duration': [2]

})

prediction = classifier.predict(new\_applicant)

print(f'Predicted Credit Risk for the new applicant: {prediction[0]}')

### OUTPUT:

income credit\_score debt\_to\_income\_ratio employment\_duration risk

0 50000 600 0.4 1 high

1 80000 750 0.2 3 low

2 60000 700 0.3 2 low

3 75000 720 0.2 2 low

4 90000 780 0.1 4 low

5 55000 620 0.5 1 high

6 72000 700 0.3 2 low

7 85000 760 0.2 3 low

Model Accuracy: 100.00%

Decision Tree Rules:

|--- income <= 55000.00

| |--- class: high

|--- income > 55000.00

| |--- class: low

Predicted Credit Risk for the new applicant: low

24. A company wants to know the most popular product they sell. They have a list of all the products they have sold in the past year, along with the number of times each product was sold. Write a program that will calculate the frequency distribution of products sold and print out the most popular product

## CODE:

import numpy as np

product\_sales = {

'ProductA': 150,

'ProductB': 200,

'ProductC': 120,

'ProductD': 180,

'ProductE': 250,

'ProductF': 200,

}

# Calculate frequency distribution

frequency\_distribution = {}

for product, sales\_count in product\_sales.items():

frequency\_distribution[product] = sales\_count

# Find the most popular product

most\_popular\_product = max(frequency\_distribution, key=frequency\_distribution.get)

# Print the frequency distribution

print("Product Frequency Distribution:")

for product, sales\_count in frequency\_distribution.items():

print(f"{product}: {sales\_count} times")

# Print the most popular product

print(f"\nThe most popular product is: {most\_popular\_product} with {frequency\_distribution[most\_popular\_product]} sales.")

### OUTPUT:

Product Frequency Distribution:

ProductA: 150 times

ProductB: 200 times

ProductC: 120 times

ProductD: 180 times

ProductE: 250 times

ProductF: 200 times

The most popular product is: ProductE with 250 sales.

25. Given a dataset, write a program to perform estimation techniques such as mean estimation, variance estimation, and sampling techniques to infer population characteristics.

## CODE:

import numpy as np

import pandas as pd

np.random.seed(42)

data = pd.DataFrame({

'value': np.random.randint(1, 100, 1000)

})

print(data.head())

mean\_estimate = np.mean(data['value'])

print(f'Mean Estimate: {mean\_estimate}')

variance\_estimate = np.var(data['value'], ddof=1)

print(f'Variance Estimate: {variance\_estimate}')

sample\_size = 100

random\_sample = data['value'].sample(n=sample\_size, random\_state=42)

sample\_mean = np.mean(random\_sample)

sample\_variance = np.var(random\_sample, ddof=1)

print(f'\nSample Mean: {sample\_mean}')

print(f'Sample Variance: {sample\_variance}')

### OUTPUT:

value

0 52

1 93

2 15

3 72

4 61

Mean Estimate: 49.56

Variance Estimate: 857.7681681681681

Sample Mean: 50.99

Sample Variance: 762.131212121212

26. A weather station wants to know if there is a correlation between the temperature and the amount of rainfall in a city. They have data on the temperature and rainfall each day for the past year in that city. Write a program that will calculate the correlation coefficient between temperature and rainfall, and create a scatter plot of the data.

## CODE:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

np.random.seed(42)

data = pd.DataFrame({

'temperature': np.random.randint(60, 100, 365),

'rainfall': np.random.uniform(0, 10, 365)

})

print(data.head())

correlation\_coefficient = data['temperature'].corr(data['rainfall'])

print(f'Correlation Coefficient: {correlation\_coefficient}')

plt.figure(figsize=(10, 6))

plt.scatter(data['temperature'], data['rainfall'], alpha=0.5)

plt.title('Temperature vs Rainfall')

plt.xlabel('Temperature (°F)')

plt.ylabel('Rainfall (mm)')

plt.grid(True)

plt.show()

### OUTPUT:

temperature rainfall

0 98 9.758521

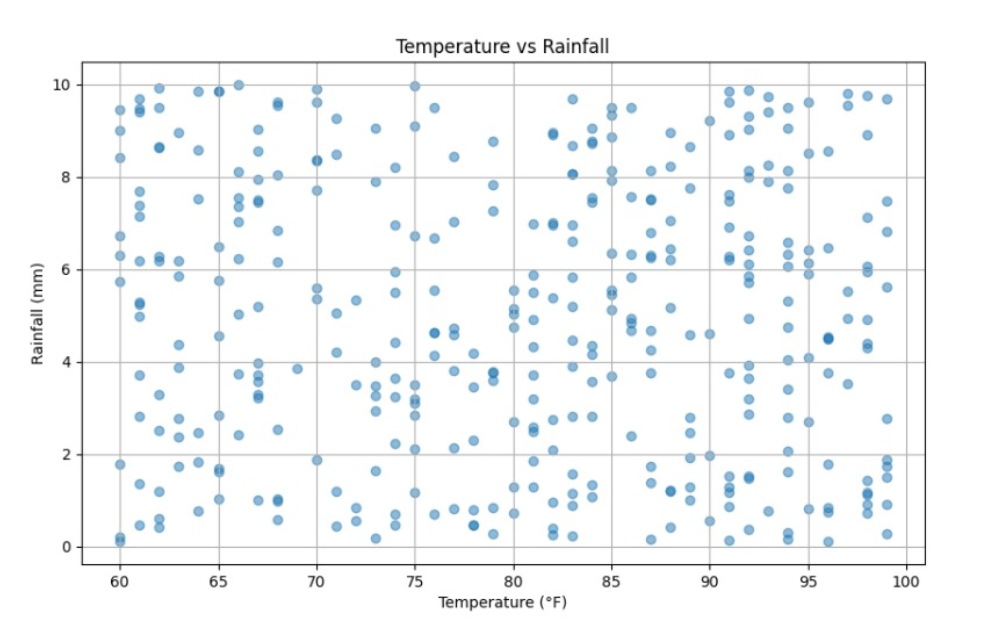
1 88 5.163003

2 74 3.229565

3 67 7.951862

4 80 2.708323

Correlation Coefficient: -0.04157827695624714



27. Scenario:Suppose you are a data analyst working for a maDErketing firm. The firm is interested in estimating the average revenue generated from a recent marketing campaign conducted on social media. You've collected a sample of the revenue generated by 100 customers who made purchases after clicking on the ads.

Question: Using Python, how would you calculate the confidence interval for the average revenue from these 100 customers? Provide Python code that computes the confidence interval at a specified confidence level (e.g., 95%) for the mean revenue.

## CODE:

import numpy as np

from scipy import stats

np.random.seed(42)

sample\_revenue = np.random.normal(loc=50, scale=10, size=100)

sample\_mean = np.mean(sample\_revenue)

standard\_error = stats.sem(sample\_revenue)

confidence\_level = 0.95

degrees\_of\_freedom = len(sample\_revenue) - 1

confidence\_interval = stats.t.interval(confidence\_level, degrees\_of\_freedom, loc=sample\_mean, scale=standard\_error)

print(f"Sample Mean: {sample\_mean}")

print(f"Confidence Interval ({confidence\_level\*100}%): {confidence\_interval}")

### OUTPUT:

Sample Mean: 48.96153482605907

Confidence Interval (95.0%): (47.159531636381004, 50.76353801573714)

28. Consider a sample car dataset and plot Mutlivariate graphs to show the distribution of data from multiple variables for Multivariate Scatterplot and Scatter Plot Matrix

## CODE:

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

np.random.seed(42)

n\_samples = 100

car\_data = pd.DataFrame({

'mpg': np.random.uniform(10, 40, n\_samples),

'horsepower': np.random.uniform(80, 300, n\_samples),

'weight': np.random.uniform(2000, 5000, n\_samples),

'acceleration': np.random.uniform(5, 20, n\_samples),

'origin': np.random.choice(['USA', 'Europe', 'Japan'], n\_samples)

})

print(car\_data.head())

plt.figure(figsize=(10, 6))

sns.scatterplot(x='horsepower', y='mpg', hue='acceleration', size='weight', data=car\_data, palette='viridis')

plt.title('Multivariate Scatterplot')

plt.show()

sns.set(style="ticks")

sns.pairplot(car\_data, vars=['mpg', 'horsepower', 'weight', 'acceleration'], hue='origin', palette='viridis')

plt.suptitle("Scatter Plot Matrix", y=1.02)

plt.show()

### OUTPUT:

mpg horsepower weight acceleration origin

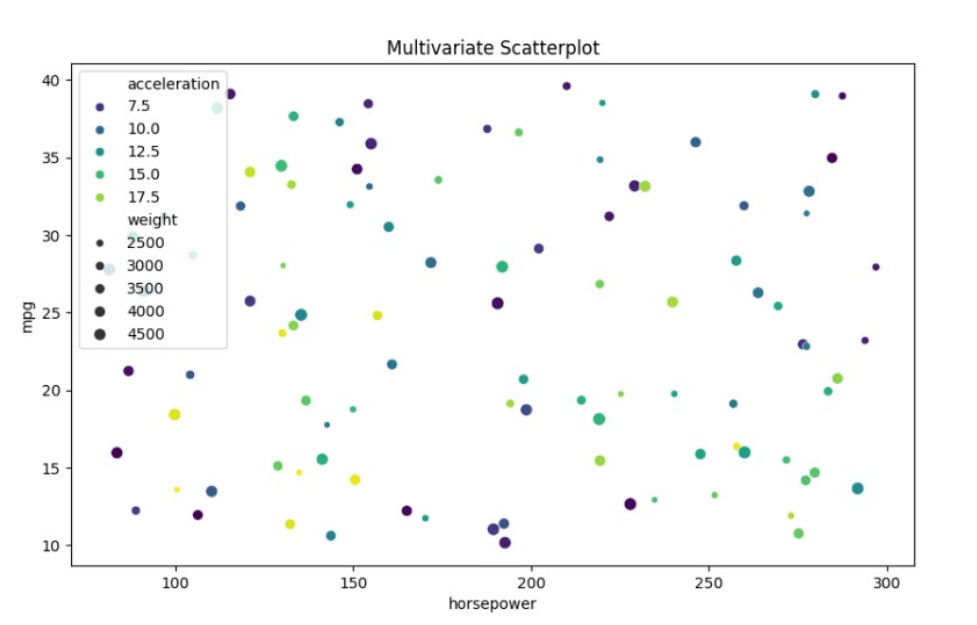
0 21.236204 86.914421 3926.094938 5.775226 USA

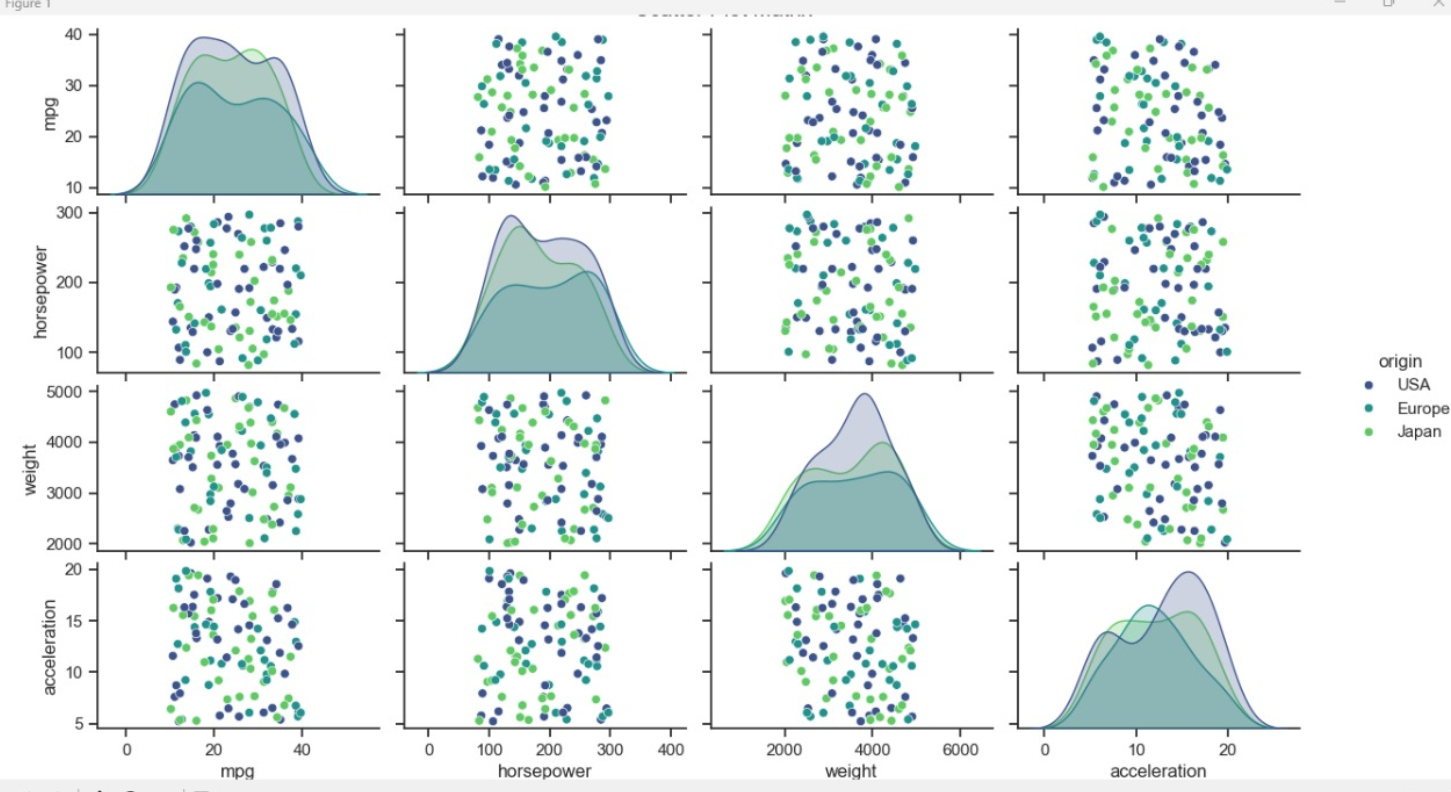
1 38.521429 220.010290 2252.419895 12.970319 Europe

2 31.959818 149.158316 2484.886142 13.109527 USA

3 27.959755 191.885552 4695.662566 14.561449 USA

4 14.680559 279.664624 3819.287179 15.891370  Europe





29. Imagine you are an analyst for a popular online shopping website. Your task is to analyze customer reviews and provide insights on the average rating and customer satisfaction level for a specific product category. You will use the pandas library to calculate confidence intervals to estimate the true population mean rating. You have been provided with a CSV file named "customer\_reviews.csv," which contains customer ratings for products in the chosen category.

## CODE:

import numpy as np

import pandas as pd

from scipy import stats

np.random.seed(42)

n\_reviews = 500

customer\_reviews = pd.DataFrame({

'rating': np.random.choice([1, 2, 3, 4, 5], n\_reviews, p=[0.05, 0.1, 0.2, 0.3, 0.35])

})

print(customer\_reviews.head())

sample\_mean = customer\_reviews['rating'].mean()

standard\_error = stats.sem(customer\_reviews['rating'])

confidence\_level = 0.95

degrees\_of\_freedom = len(customer\_reviews['rating']) - 1

confidence\_interval = stats.t.interval(confidence\_level, degrees\_of\_freedom, loc=sample\_mean, scale=standard\_error)

print(f"Sample Mean Rating: {sample\_mean:.2f}")

print(f"Confidence Interval ({confidence\_level\*100}%): {confidence\_interval}")

### OUTPUT:

rating

0 4

1 5

2 5

3 4

4 3

Sample Mean Rating: 3.75

Confidence Interval (95.0%): (3.640837710168692, 3.8551622898313083)

30. You are a data scientist working for a company that sells shoes. You are tasked with writing a program that will calculate the frequency distribution of shoe sizes sold in the past year. The data is stored in a file called shoe\_sales.csv. The file contains the following columns:

• shoe\_size: The size of the shoe sold.

• quantity: The number of shoes sold in that size.

Write a program that will read the data from the file and calculate the frequency distribution of shoe sizes. The program should output the frequency distribution table, as well as a bar chart showing the frequency of each shoe size.

## CODE:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

np.random.seed(42)

n\_sales = 500

shoe\_sales = pd.DataFrame({

'shoe\_size': np.random.choice(['US 7', 'US 8', 'US 9', 'US 10', 'US 11'], n\_sales),

'quantity': np.random.randint(1, 20, n\_sales)

})

print(shoe\_sales.head())

size\_distribution = shoe\_sales.groupby('shoe\_size')['quantity'].sum()

print("Frequency Distribution Table:")

print(size\_distribution)

plt.figure(figsize=(10, 6))

size\_distribution.plot(kind='bar', color='skyblue')

plt.title('Shoe Size Frequency Distribution')

plt.xlabel('Shoe Size')

plt.ylabel('Quantity Sold')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

### OUTPUT:

shoe\_size quantity

0 US 10 16

1 US 11 13

2 US 9 19

3 US 11 17

4 US 11 4

Frequency Distribution Table:

shoe\_size

US 10 1118

US 11 886

US 7 1001

US 8 965

US 9 941

Name: quantity, dtype: int32

