1.import pandas as pd
import numpy as np
from scipy.stats import zscore
from sklearn.decomposition import PCA
# 1 Create a sample dataset
data = {
'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],
'Age': [25, 30, 29, 28, 90],
'Salary': [50000, 54000, 58000, 56000, 200000],
'Experience': [1, 3, 2, 4, 25]
}
# Create DataFrame
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
# 2 Detect outliers using Z-score
z_scores = np.abs(zscore(df[['Age', 'Salary', 'Experience']]))
print("\nZ-Scores:")
print(z_scores)
# Rows with Z-score > 3 are outliers
outliers = (z_scores > 3).any(axis=1)
print("\nRows with Outliers (Z > 3):")
print(df[outliers])
# Remove outliers
cleaned_df = df[~outliers]
print("\nCleaned DataFrame (after removing outliers):")
print(cleaned_df)

2. from sklearn.decomposition import PCA
import numpy as np
# Sample data: 5 samples with 3 features each
data = np.array([
[2.5, 2.4, 1.2],
[0.5, 0.7, 0.9],
[2.2, 2.9, 1.4],
[1.9, 2.2, 1.1],
[3.1, 3.0, 1.6]
])
# Apply PCA to reduce dimensions to 2
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(data)
print("Original Data:")
print(data)
print("\nReduced Data:")
print(reduced_data)
import numpy as np
from sklearn.preprocessing import MinMaxScaler
# Sample data
data = np.array([[10, 20], [30, 40], [50, 60]])
# Create MinMaxScaler with custom range [-1, 1]
scaler = MinMaxScaler(feature_range=(-1, 1))
data_normalized = scaler.fit_transform(data)
print("Normalized Data with Range [-1, 1]:")
print(data_normalized)

from sklearn.preprocessing import StandardScaler
# Standardize data
scaler = StandardScaler()
data_standardized = scaler.fit_transform(data)
print("Standardized Data:")
print(data_standardized)

3. import numpy as np
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
# Step 1: Create data

```
X = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1)    # hours studied
y = np.array([0, 0, 0, 1, 1, 1])                   # pass/fail (0 or 1)
# Step 2: Create model
model = LogisticRegression()
# Step 3: Train the model
model.fit(X, y)
# Step 4: Predict
x_test = np.array([[2.5], [4]])  # test data
y_pred = model.predict(x_test)
print("Predictions:", y_pred)
plt.scatter(X, y, color='blue', label='Training Data')
plt.plot(X_range, y_prob, color='red', label='Sigmoid Curve')
plt.xlabel('Hours Studied')
plt.ylabel('Probability of Passing')
plt.legend()
plt.grid(True)
plt.show()
```

4.
```
import matplotlib.pyplot as plt
import numpy as np

# Data
categories = ['A', 'B', 'C', 'D']
values = [10, 20, 15, 25]
colors = ['skyblue', 'lightgreen', 'orange', 'pink']

# Create bar chart
plt.figure(figsize=(8, 6))
bars = plt.bar(categories, values, color=colors, edgecolor='black')

# Add title and labels
plt.title('Bar Chart Example', fontsize=16, fontweight='bold')
plt.xlabel('Categories', fontsize=14)
plt.ylabel('Values', fontsize=14)

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

# Add legend
plt.legend(bars, categories, title='Categories', loc='upper right')

# Show plot
plt.show()
```

5.
```
import matplotlib.pyplot as plt
import numpy as np

# Data
x = np.arange(10)
y = x ** 2

# Create line chart
plt.figure(figsize=(8, 6))
plt.plot(x, y, marker='o', linestyle='-', color='blue',
        label='y = x^2', linewidth=2, markersize=8)

# Add title and labels
plt.title('Line Chart Example', fontsize=16, fontweight='bold')
plt.xlabel('X-axis', fontsize=14)
plt.ylabel('Y-axis', fontsize=14)

# Add grid
plt.grid(linestyle='--', alpha=0.5)

# Add annotations
for (xi, yi) in zip(x, y):
```

```
    plt.text(xi, yi, f'({xi}, {yi})', fontsize=10, ha='right')

# Add legend
plt.legend(loc='upper left')

# Show plot
plt.show()
```