1.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data = {
 'Rainfall': [100, 150, 200, 250, 300],
 'Temperature': [22, 25, 21, 20, 23],
 'Humidity': [85, 80, 90, 95, 88]
}
df = pd.DataFrame(data)
correlation_matrix = df.corr()
print(correlation_matrix)
sns.pairplot(df)
plt.show()
```

2.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
df = pd.read_csv('house-prices.csv')
print(df.head())
print(df.describe())
z_scores = np.abs(stats.zscore(df.select_dtypes(include=[np.number])))
outliers_z = np.where(z_scores > 3, True, False)
print("Outliers detected by Z-score method:\n", df[outliers_z.any(axis=1)])
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
```

```
IQR = Q3 - Q1
outliers_iqr = (df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))
print("Outliers detected by IQR method:\n", df[outliers_iqr.any(axis=1)])
numerical_columns = df.select_dtypes(include=[np.number]).columns
plt.figure(figsize=(15, 10))
df[numerical_columns].boxplot()
plt.xticks(rotation=90)
plt.title("Box plot of numerical features")
plt.show()
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='SqFt', y='Price')
plt.title('Scatter plot of Price vs Area')
plt.show()
```

3.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('house-prices.csv')
print(df.head())
mean_prices_by_neighborhood = df.groupby('Neighborhood')['Price'].mean().reset_index()
print(mean_prices_by_neighborhood)
plt.figure(figsize=(14, 8))
sns.barplot(x='Neighborhood', y='Price', data=mean_prices_by_neighborhood)
plt.xticks(rotation=90)
plt.title('Mean Housing Prices by Neighborhood')
plt.xlabel('Neighborhood')
plt.ylabel('Mean Price')
plt.show()
```

4.

```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
data = {
    'temperature': [23, 25, 22, 24, 23, 24, 25, 26, 24, 23, 22, 24, 25, 26, 27, 28],
    'humidity': [65, 70, 68, 72, 69, 71, 70, 73, 67, 66, 68, 72, 71, 74, 75, 76],
    'rainfall': [100, 110, 95, 105, 98, 102, 99, 101, 104, 106, 97, 103, 108, 100, 99, 101]
}
df = pd.DataFrame(data)
def calculate_confidence_interval(feature, confidence_level=0.95):
    mean = np.mean(feature)
    sem = stats.sem(feature)
    ci = stats.t.interval(confidence_level, len(feature)-1, loc=mean, scale=sem)
    return mean, ci
features = ['temperature', 'humidity', 'rainfall']
confidence_level = 0.95
means = []
conf_intervals = []

for feature_name in features:
    feature = df[feature_name]
    mean, ci = calculate_confidence_interval(feature, confidence_level)
    means.append(mean)
    conf_intervals.append(ci)

fig, ax = plt.subplots()
x_pos = np.arange(len(features))
```

```
means = np.array(means)
conf_intervals = np.array(conf_intervals)
error = np.array([(mean - ci[0], ci[1] - mean) for mean, ci in zip(means, conf_intervals)]).T
ax.bar(x_pos, means, yerr=error, align='center', alpha=0.7, capsize=10)
ax.set_ylabel('Mean values')
ax.set_xticks(x_pos)
ax.set_xticklabels(features)
ax.set_title('Mean and 95% Confidence Interval for Features')
plt.show()
```

5.

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()
sep_length = iris.data[:,0]
a_1, a_2 = train_test_split(sep_length, test_size=0.4, random_state=0)
b_1, b_2 = train_test_split(sep_length, test_size=0.4, random_state=1)
mu1 = np.mean(a_1)
mu2 = np.mean(b_1)
np.std(a_1)
np.std(b_1)
stats.ttest_ind(a_1, b_1, equal_var = False)
```

6.

```
import pandas as pd
import numpy as np
import scipy.stats as stats
df = pd.read_csv('house-prices.csv')
data = df['Price']
mean = np.mean(data)
std_err = stats.sem(data)
confidence = 0.95
margin_of_error = std_err * stats.t.ppf((1 + confidence) / 2., len(data) - 1)
confidence_interval = (mean - margin_of_error, mean + margin_of_error)
print(f"Mean: {mean}")
print(f"Standard Error: {std_err}")
print(f"Confidence Interval: {confidence_interval}")
```

7.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"
df = pd.read_csv(url)
print(df.head())
target = 'medv'
features = df.drop(columns=[target])
labels = df[target]
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")
```

8.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
df = pd.read_csv('https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv')
df = df.drop(columns=['Name', 'Ticket', 'Cabin', 'PassengerId'])
df = df.dropna(subset=['Fare'])
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
X = df.drop(columns=['Fare'])
y = df['Fare']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
```

9.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

10.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
df = pd.read_csv('https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv')
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
X = df[features]
y = df['Survived']
X['Age'] = X['Age'].fillna(X['Age'].median())
X['Embarked'] = X['Embarked'].fillna(X['Embarked'].mode()[0])
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=400)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```