1. Probability

a. Calculating the simple probabilities.

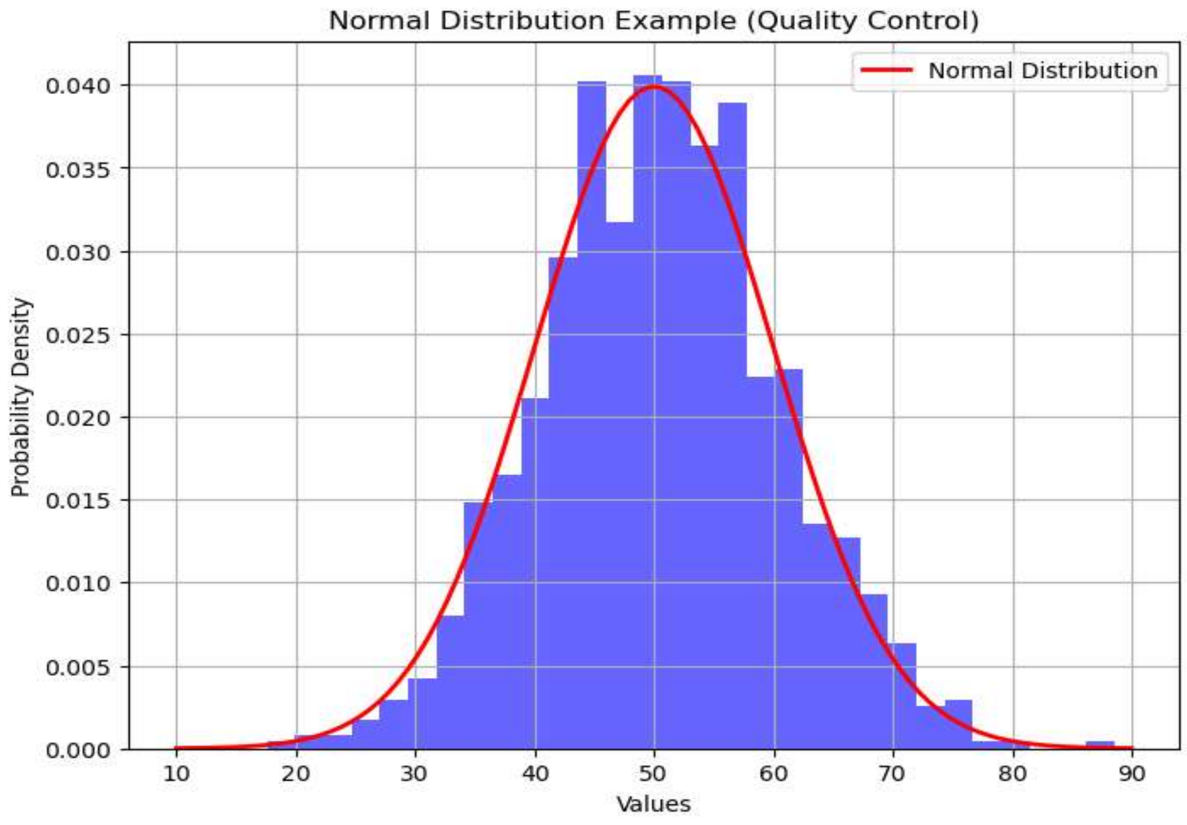
b. Applications of Probability distributions to real life problems.

```
# Simple probability
# Probability of rolling a 4 on a six-sided die
total_outcomes = 6
favorable_outcomes = 1  # Rolling a 4
probability_4 = favorable_outcomes / total_outcomes
print(f"Probability of rolling a 4: {probability_4}")

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, poisson, binom, expon
# Normal Distribution - Quality Control example
# Generating and plotting a normal distribution
mean = 50
std_dev = 10
samples = np.random.normal(mean, std_dev, 1000)
plt.figure(figsize=(8, 6))
plt.hist(samples, bins=30, density=True, alpha=0.6, color='blue')
x = np.linspace(mean - 4*std_dev, mean + 4*std_dev, 100)
plt.plot(x, norm.pdf(x, mean, std_dev), 'r-', lw=2, label='Normal Distribution')
plt.title('Normal Distribution Example (Quality Control)')
plt.xlabel('Values')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()

# Poisson Distribution - Service and Arrival Rates example
# Calculating the probability of a certain number of events occurring in a time frame
lambda_param = 5  # Arrival rate per hour
k = 3  # Number of events
prob_3_events = poisson.pmf(k, lambda_param)
print(f"Probability of 3 events occurring in an hour: {prob_3_events}")

# Binomial Distribution - Decision Making example
# Estimating probability of success or failure in fixed number of trials
n = 10  # Number of trials
p = 0.6  # Probability of success
k_success = 7  # Number of successes
prob_7_success = binom.pmf(k_success, n, p)
print(f"Probability of 7 successes out of 10 trials: {prob_7_success}")
```

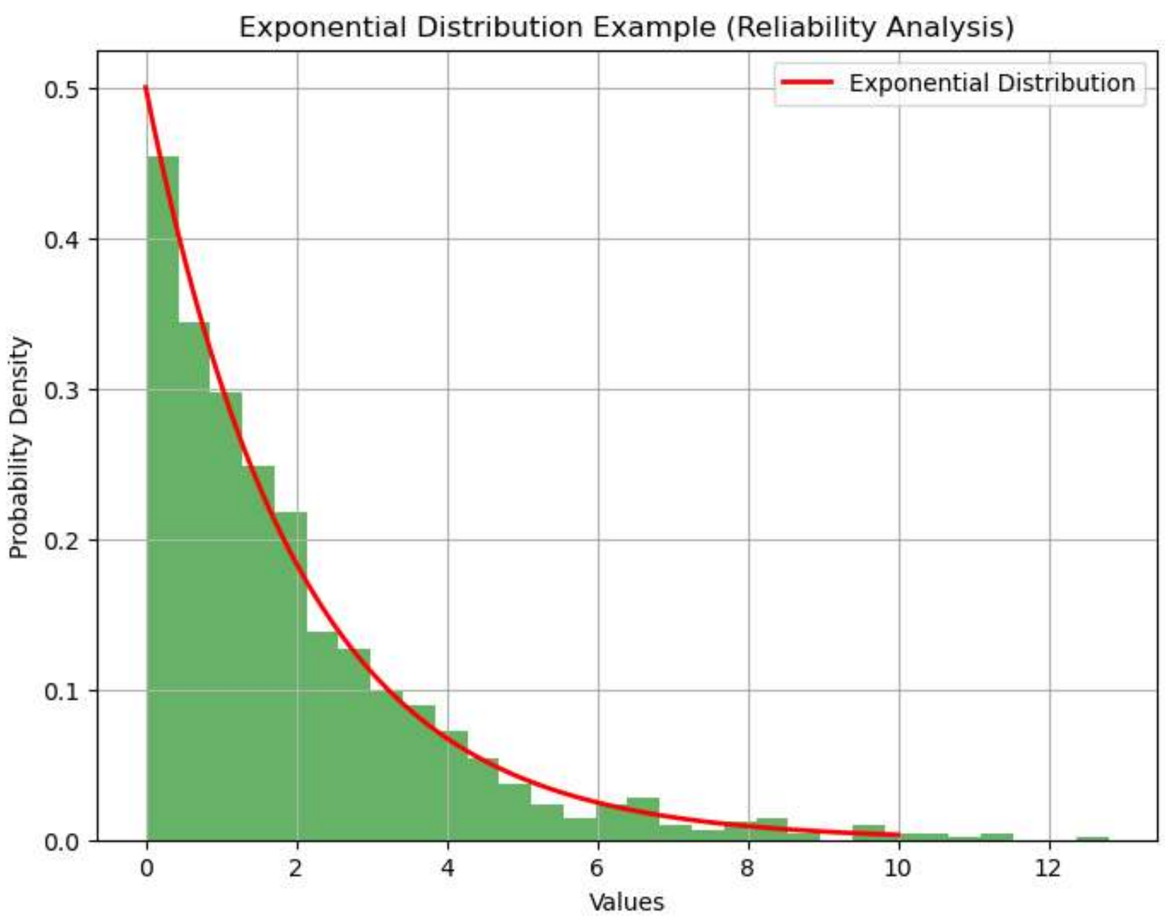
```
# Exponential Distribution - Reliability Analysis example
# Simulating and plotting an exponential distribution
exp_samples = np.random.exponential(scale=2, size=1000)
plt.figure(figsize=(8, 6))
plt.hist(exp_samples, bins=30, density=True, alpha=0.6, color='green')
x_exp = np.linspace(0, 10, 100)
plt.plot(x_exp, expon.pdf(x_exp, scale=2), 'r-', lw=2, label='Exponential
Distribution')
plt.title('Exponential Distribution Example (Reliability Analysis)')
plt.xlabel('Values')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT

```
Probability of rolling a 4: 0.16666666666666666
```



```
Probability of 3 events occurring in an hour: 0.1403738958142805
Probability of 7 successes out of 10 trials: 0.21499084799999976
```



2. Test of Significance

a. T-Test: one sample, two independent samples and paired

b. ANOVA & Chi-Square Test.

```
import pandas as pd
from scipy import stats
# Load Titanic dataset
titanic_data = pd.read_csv('train.csv')  # Replace 'train.csv' with your dataset file
# One Sample T-Test: Checking mean age against a hypothetical mean
hypothetical_mean_age = 30
ttest_one_sample = stats.ttest_1samp(titanic_data['Age'].dropna(),
hypothetical_mean_age)
print("One Sample T-Test:")
print("T-statistic:", ttest_one_sample.statistic)
print("p-value:", ttest_one_sample.pvalue)
```

```
# Two Independent Samples T-Test: Comparing ages of male and female passengers
male_ages = titanic_data[titanic_data['Sex'] == 'male']['Age'].dropna()
female_ages = titanic_data[titanic_data['Sex'] == 'female']['Age'].dropna()
ttest_two_ind_samples = stats.ttest_ind(male_ages, female_ages)
print("\nTwo Independent Samples T-Test:")
print("T-statistic:", ttest_two_ind_samples.statistic)
print("p-value:", ttest_two_ind_samples.pvalue)
# Paired T-Test: Comparing fares before and after
before_fares = titanic_data['Fare'].dropna()
after_fares = before_fares * 1.2  # Assuming a 20% increase in fares
ttest_paired = stats.ttest_rel(before_fares, after_fares)
print("\nPaired T-Test:")
print("T-statistic:", ttest_paired.statistic)
print("p-value:", ttest_paired.pvalue)
# ANOVA Test: Impact of passenger class on fares
anova_result = stats.f_oneway(titanic_data[titanic_data['Pclass'] == 1]['Fare'].dropna(),
                              titanic_data[titanic_data['Pclass'] == 2]['Fare'].dropna(),
                              titanic_data[titanic_data['Pclass'] == 3]['Fare'].dropna())
print("\nANOVA Test Result:")
print("F-statistic:", anova_result.statistic)
print("p-value:", anova_result.pvalue)
# Chi-Square Test: Relationship between survival status and passenger class
chi2_table = pd.crosstab(titanic_data['Survived'], titanic_data['Pclass'])
chi2_result = stats.chi2_contingency(chi2_table)
print("\nChi-Square Test Result:")
print("Chi-Square statistic:", chi2_result[0])
print("p-value:", chi2_result[1])
```

OUTPUT

```
One Sample T-Test:

T-statistic: -0.5534583115970276
p-value: 0.5801231230388639

Two Independent Samples T-Test:
T-statistic: 2.499206354920835
p-value: 0.012671296797013709

Paired T-Test:
T-statistic: -19.344277455944212
p-value: 7.255925461999273e-70

ANOVA Test Result:
F-statistic: 242.34415651744814
p-value: 1.0313763209141171e-84

Chi-Square Test Result:
Chi-Square statistic: 102.88898875696056
p-value: 4.549251711298793e-23
```

3. Correlation and Regression analysis

a. Scattered diagram, calculating of correlation coefficient

b. Linear regression: fitting, testing model adequacy and prediction

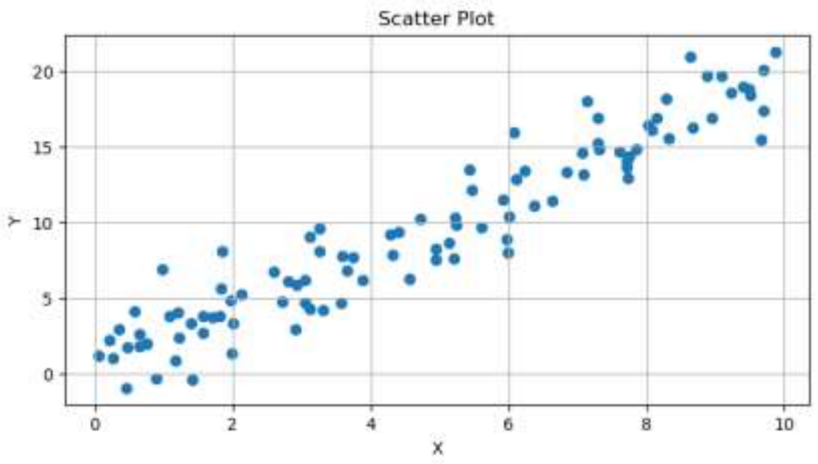
c. Fitting of logistic regression.

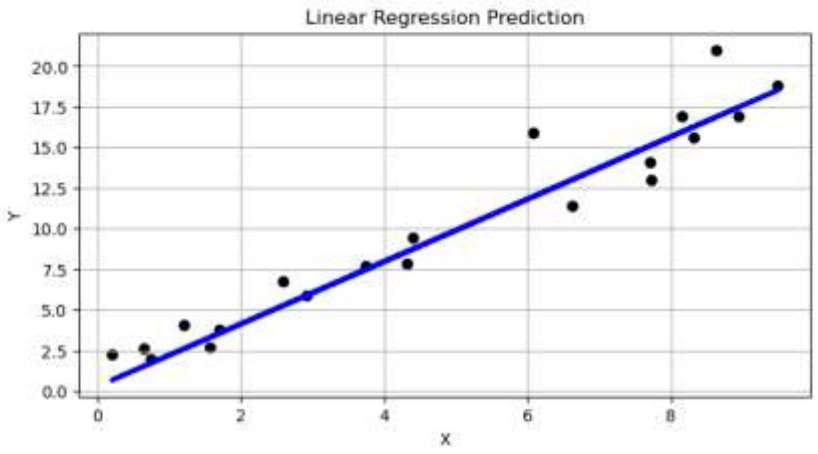
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

# Generating sample data
np.random.seed(42)
X = np.random.rand(100, 1) * 10
y = 2 * X.squeeze() + np.random.randn(100) * 2

# Scatter plot and correlation coefficient
plt.figure(figsize=(8, 4))
plt.scatter(X, y)
plt.title('Scatter Plot')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
correlation_coefficient = np.corrcoef(X.squeeze(), y)[0, 1]
print(f"Correlation Coefficient: {correlation_coefficient}")

# Linear regression fitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
```

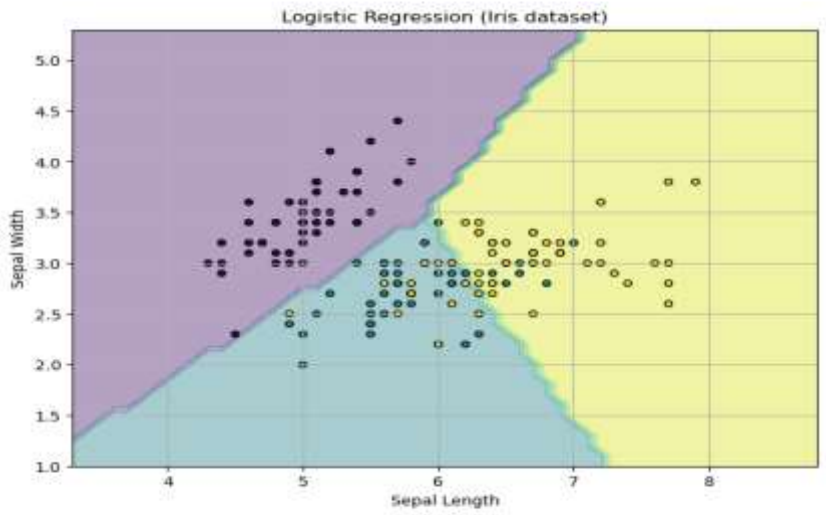
```
# Testing model adequacy and prediction
y_pred = lin_reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")

plt.figure(figsize=(8, 4))
plt.scatter(X_test, y_test, color='black')
plt.plot(X_test, y_pred, color='blue', linewidth=3)
plt.title('Linear Regression Prediction')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)

# Fitting logistic regression (using Iris dataset as an example)
iris = load_iris()
X_iris = iris.data[:, :2]  # Using only the first two features for simplicity
y_iris = iris.target
log_reg = LogisticRegression()
log_reg.fit(X_iris, y_iris)

# Generating a meshgrid for decision boundary visualization
x_min, x_max = X_iris[:, 0].min() - 1, X_iris[:, 0].max() + 1
y_min, y_max = X_iris[:, 1].min() - 1, X_iris[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max,
0.1))
Z = log_reg.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X_iris[:, 0], X_iris[:, 1], c=y_iris, s=20, edgecolor='k')
plt.title('Logistic Regression (Iris dataset)')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.grid(True)
plt.show()
```

OUTPUT

```
Correlation Coefficient: 0.9529657473628446
Mean Squared Error: 2.6147980548680083
R-squared Score: 0.9287298556395622
```


Scatter Plot
X
Y


Linear Regression Prediction
X
Y


Logistic Regression (Iris dataset)
Sepal Length
Sepal Width