Part-B: DA Programs:

1. Probability

Program -1:

```
# 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("Probability of rolling a 4:", probability_4)
```

output:

Probability of rolling a 4: 0.16666666666666666

1. A) Calculating the simple probabilities

```
import random
num_trials = int(input("enter_no_of_trials"))
rolls_per_trial =int(input("for Each trail how many rolls"))
roll_up_value=int(input(" Enter rollup value"))
poss_outcomes=0
for i in range(num_trials):
    for j in range(rolls_per_trial):
        result = random.randint(1,6)
        print(result)
        if result == roll_up_value:
```

```
            poss_outcomes += 1
        print("--------")
total_outcomes = num_trials * rolls_per_trial
print(f"Number of times 6 appeared in {num_trials} trials of {rolls_per_trial} rolls each: {poss_outcomes}")
print("probability=",poss_outcomes / total_outcomes)
```

output:

```
enter_no_of_trials 5
for Each trail how many rolls 2
 Enter rollup value 6
3
2
--------
1
2
--------
6
2
--------
6
6
--------
1
5
--------
Number of times 6 appeared in 5 trials of 2 rolls each: 3
probability= 0.3
```

1. **b) Applications of Probability distributions to real life problems**

```
# Binomial Distribution - Decision Making example

# estimating probability of success or failure in fixed number of trials

from scipy.stats import binom

n = 10 # Number of trials

p = 0.5 # Probability of success

k_success = 2# Number of successes

prob_2_success = binom.pmf(k_success, n, p)

print(f"Probability of 2 successes out of 10 trials: {prob_2_success}")
```

output:

Probability of 2 successes out of 10 trials: 0.04394531250000005

Program 2: Test of significance

```
import pandas as pd

from scipy import stats

titanic_data = pd.read_csv('train.csv')

# 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: 7.255925461999272e-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

program 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.

Source Code:

```
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)

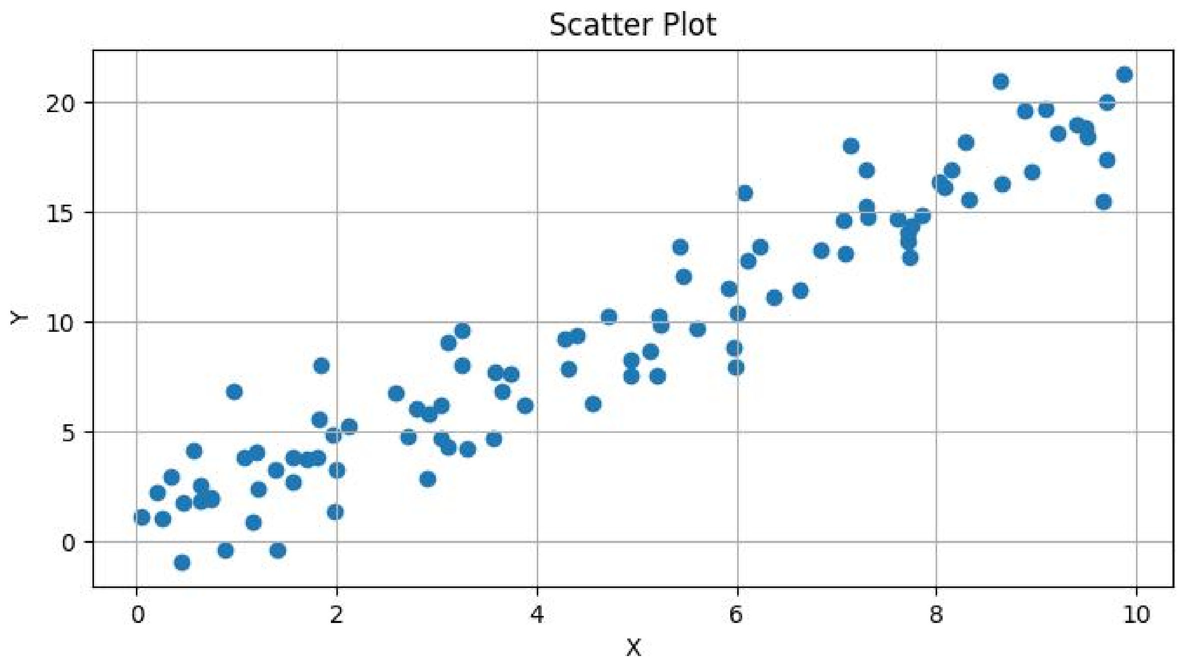
```
# Fitting logistic regression (using Iris dataset as an example)
iris = load_iris()
X_iris = iris.data[:, :2] # Using only the first two features for simplicity
y_iris = iris.target
log_reg = LogisticRegression()
log_reg.fit(X_iris, y_iris)
# Generating a meshgrid for decision boundary visualization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

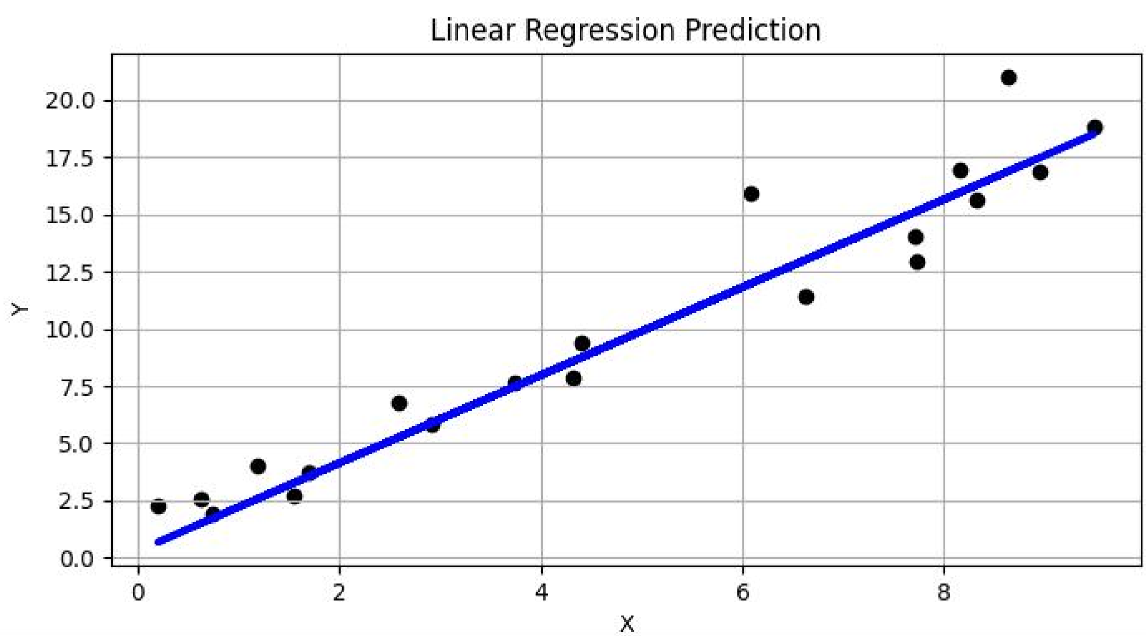
output:

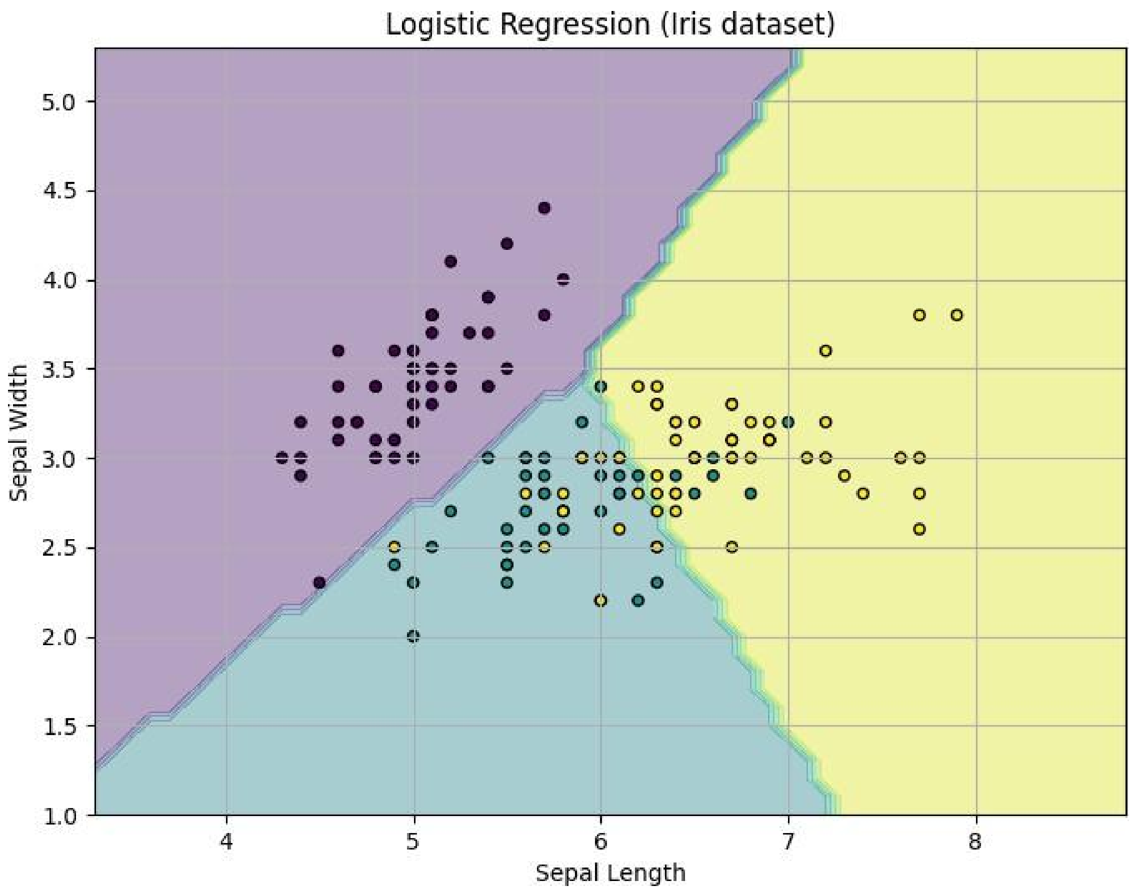
Correlation Coefficient: 0.9529657473628446

Mean Squared Error: 2.6147980548680088

R-squared Score: 0.9287298556395622






Logistic Regression (Iris dataset)
Sepal Width
Sepal Length
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8