FUNDAMENTALS OF DATA SCIENCE

REGISTER NUMBER: 230701177

Experiment 1.D

Code :

```
from cryptography.fernet import Fernet
key=Fernet.generate_key()
f=Fernet(key)
token=f.encrypt(b"REC")
f.decrypt(token)
b"REC"
key=Fernet.generate_key()
cs=Fernet(key)
pt=b"REC"
ct=cs.encrypt(pt)
print(pt)
print(ct)
dt=cs.decrypt(ct)
print(dt)
```

Output :

```
b'REC'
b'gAAAAABmwrYFadvc5y4S3nP0GX_ZkFXYkd-eLSBwO0tYjiueilr27P0-IgdcstA0xc5M_l98f
3tUcbsL5iWP-z-LD9v-IRU94A=='
b'REC'
```

FUNDAMENTALS OF DATA SCIENCE

REG NO : 230701177

Experiment 1.C

Structured Data

Code :

```
import pandas as pd
data={
   'Name': ['John', 'Smith'],
   'Email': ['john@example.com', 'smith@example.com']
}
df=pd.DataFrame(data)
print(df)
```

Output :

```
Name               Email
0   John   john@example.com
1  Smith  smith@example.com
```

Unstructured Data

Code :

```
n = [1, 2, 3, 4, 5]

print("The numbers are : ")

for i in n:

   print(i,end=',')
```

Output :

```
The numbers are :
1,2,3,4,5
```

Semi-Structured Data

Code :

```
import pandas as pd
data={'NAMES':('John','Smith','Thor'),'ID':(108,179,170)}
age=[18,17,19]
structured_data=pd.DataFrame(data)
print(structured_data,age)
```

Output :

```
   NAMES   ID
0   John  108
1  Smith  179
2   Thor  170 [18, 17, 19]
```

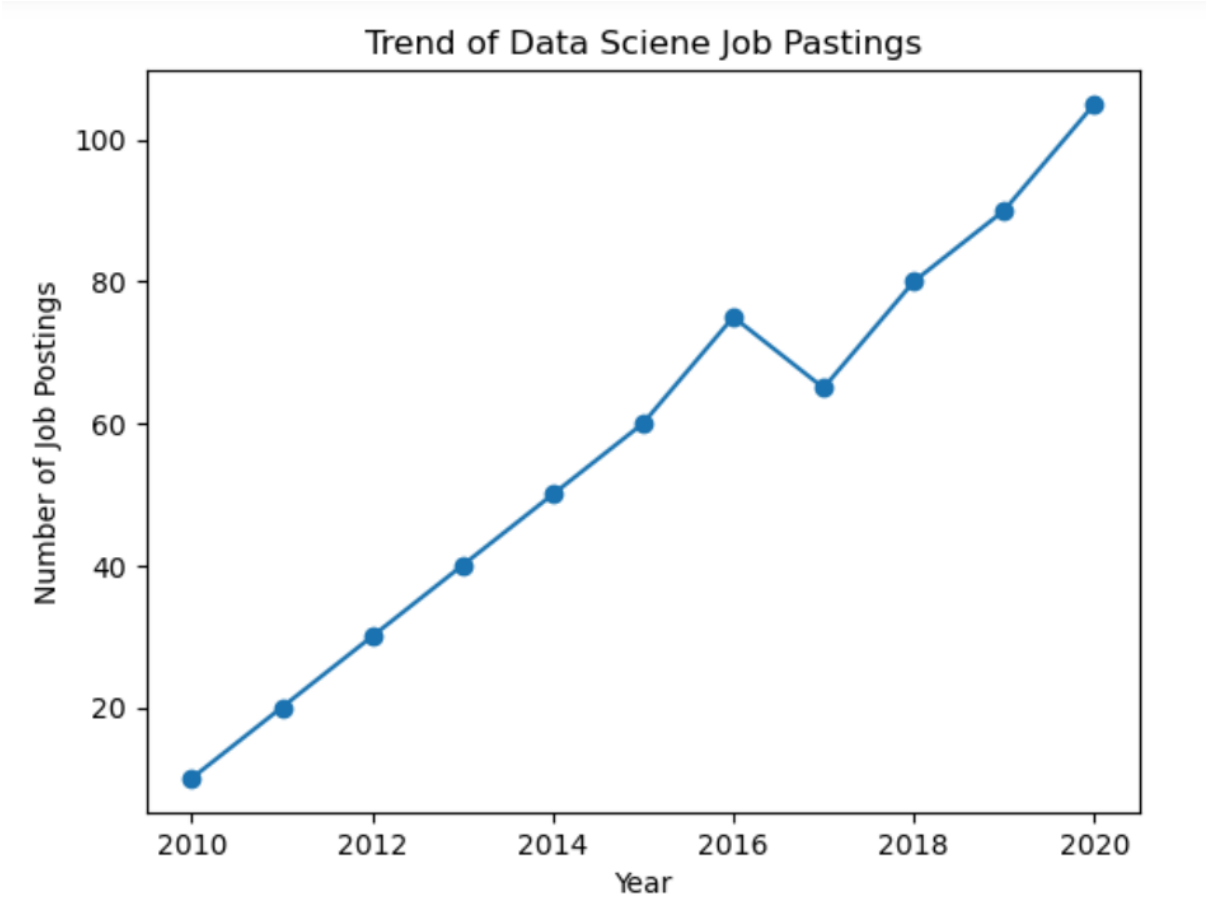
FUNDAMENTALS OF DATA SCIENCE

REG NO : 230701177

Experiment 1.A

Code :

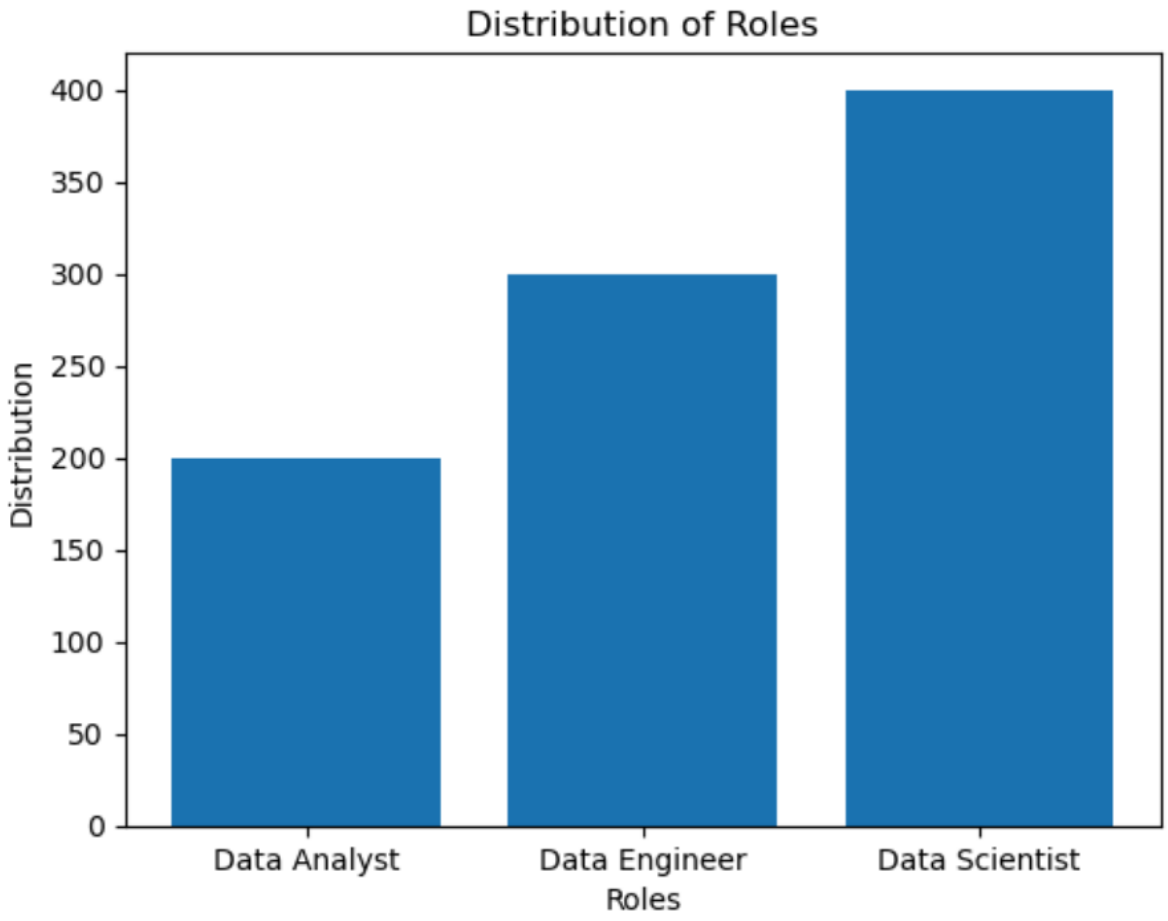
```
import pandas as pd
import matplotlib.pyplot as plt
data={'Year':list(range(2010,2021)), 'Job
Postings':[10,20,30,40,50,60,75,65,80,90,105]}
df=pd.DataFrame(data)
plt.plot(df['Year'],df['Job Postings'],marker='o')
plt.title('Trend of Data Sciene Job Pastings')
plt.xlabel('Year')
plt.ylabel('Number of Job Postings')
plt.show()
```



Experiment 1.B

Code :

```
import matplotlib.pyplot as plt
job=['Data Analyst','Data Engineer','Data Scientist']
data=[200,300,400]
plt.bar(job,data)
plt.title("Distribution of Roles")
plt.xlabel("Roles")
plt.ylabel("Distribution")
plt.show()
```



jupyter Untitled Last Checkpoint: 21 minutes ago

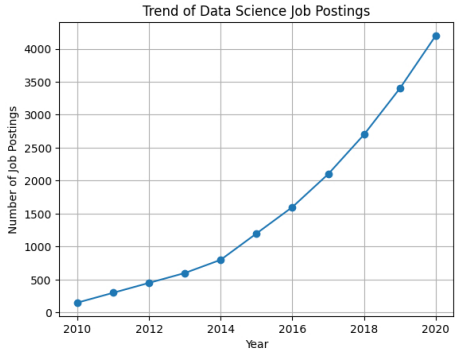
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Data
data = {
    'Year': list(range(2010, 2021)),
    'Job Postings': [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]
}

df = pd.DataFrame(data)

# Plot
plt.plot(df['Year'], df['Job Postings'], marker='o')
plt.title('Trend of Data Science Job Postings')
plt.xlabel('Year')
plt.ylabel('Number of Job Postings')
plt.grid(True)
plt.show()
```



[2]:
```
import matplotlib.pyplot as plt

# Data
roles = ['Data Analyst', 'Data Engineer', 'Data Scientist', 'ML Engineer', 'Business Analyst']
counts = [300, 500, 450, 200, 150]

plt.bar(roles, counts, color='skyblue')
plt.title('Distribution of Data Science Roles')
plt.xlabel('Role')
plt.ylabel('Count')
plt.xticks(rotation=15)
plt.show()
```

[3]:
```
import pandas as pd

# Structured data example
structured_data = pd.DataFrame({
    'ID': [1, 2, 3],
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35]
})
print("Structured Data:\n", structured_data)

# Unstructured data example
unstructured_data = "This is an example of unstructured data. It can be a piece of text, an image, or a video file."
print("\nUnstructured Data:\n", unstructured_data)

# Semi-structured data example (JSON-like format)
semi_structured_data = {'ID': 1, 'Name': 'Alice', 'Attributes': {'Height': 165, 'Weight': 68}}
print("\nSemi-structured Data:\n", semi_structured_data)
```

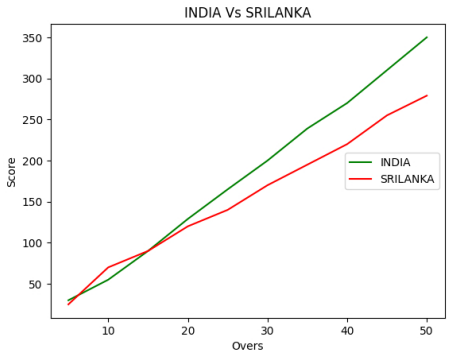
```
Structured Data:
    ID     Name  Age
0   1    Alice   25
1   2      Bob   30
2   3  Charlie   35

Unstructured Data:
 This is an example of unstructured data. It can be a piece of text, an image, or a video file.

Semi-structured Data:
 {'ID': 1, 'Name': 'Alice', 'Attributes': {'Height': 165, 'Weight': 68}}
```

[4]:
```
from cryptography.fernet import Fernet

# Generate a key
key = Fernet.generate_key()
cipher_suite = Fernet(key)

# Original data
plain_text = b"Rajalakshmi Engineering College"

# Encrypt data
cipher_text = cipher_suite.encrypt(plain_text)

# Decrypt data
decrypted_text = cipher_suite.decrypt(cipher_text)
```

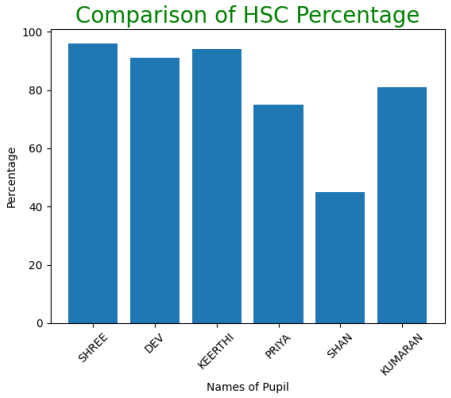
```
# Output
print("Original Data:", plain_text.decode())
print("Encrypted Data:", cipher_text.decode())
print("Decrypted Data:", decrypted_text.decode())
```

```
Original Data: Rajalakshmi Engineering College
Encrypted Data: gAAAAABnPa2yKlPfpLIVqHIm80wUNUaGZw-sMA5CrbiJlMKVRE5WvinYatxKUH8TnTYq2DiBvBIyi4OgA3PKHIUb7zNx0p-jrg9nQ1Jeri82TZAxfXIi6Fo=
Decrypted Data: Rajalakshmi Engineering College
```

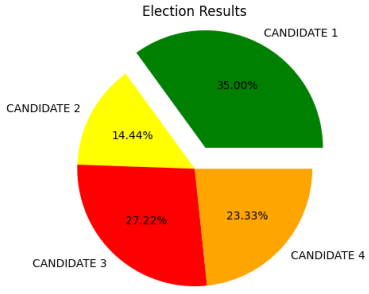
[15]:
```
import matplotlib.pyplot as cricket
Overs = list(range(5, 51, 5))
Indian_Score = [30, 55, 90, 129, 165, 200, 239, 270, 310, 350]
Srilankan_Score = [25, 70, 90, 120, 140, 170, 195, 220, 255, 279]
cricket.plot(Overs, Indian_Score, color="green", label="INDIA")
cricket.plot(Overs, Srilankan_Score, color="red", label="SRILANKA")
cricket.title("INDIA Vs SRILANKA")
cricket.xlabel("Overs")
cricket.ylabel("Score")
cricket.legend(loc="center right")
cricket.show()
```



[16]:
```
import matplotlib.pyplot as hscmark
import numpy as np
Names = ['SHREE', 'DEV', 'KEERTHI', 'PRIYA', 'SHAN', 'KUMARAN']
xaxis = np.arange(len(Names))
Percentage_hsc = [96, 91, 94, 75, 45, 81]
hscmark.bar(Names, Percentage_hsc)
hscmark.xticks(xaxis, Names, rotation=45)
hscmark.xlabel("Names of Pupil")
hscmark.ylabel("Percentage")
hscmark.title("Comparison of HSC Percentage", fontsize=20, color="green")
hscmark.show()
```



[17]:
```
import matplotlib.pyplot as election
labels = ['CANDIDATE 1', 'CANDIDATE 2', 'CANDIDATE 3', 'CANDIDATE 4']
Votes = [315, 130, 245, 210]
colors = ['green', 'yellow', 'red', 'orange']
explode = (0.2, 0, 0, 0)
election.pie(Votes, labels=labels, colors=colors, explode=explode, autopct='%0.2f%%')
election.title('Election Results')
election.show()
```



[18]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import gutenberg
nltk.download('gutenberg')
nltk.download('punkt')
sample = gutenberg.raw("austen-emma.txt")
token = word_tokenize(sample)
wlist = [token[i] for i in range(50)]
wordfreq = [wlist.count(w) for w in wlist]
print("Pairs\n" + str(list(zip(wlist, wordfreq))))
```

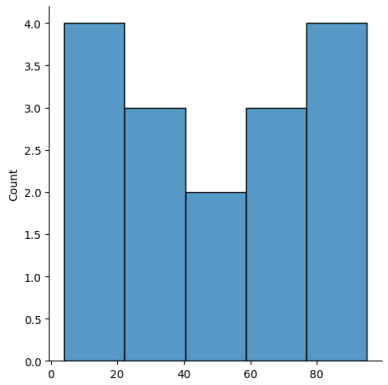
```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\MITHESH\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
```

```
[nltk_data]     C:\Users\MITHESH\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Pairs
[('[', 1), ('Emma', 2), ('by', 1), ('Jane', 1), ('Austen', 1), ('1816', 1), (']', 1), ('VOLUME', 1), ('I', 2), ('CHAPTER', 1), ('I', 2), ('Emma', 2), ('Woo
dhouse', 1), (',', 5), ('handsome', 1), (',', 5), ('clever', 1), (',', 5), ('and', 3), ('rich', 1), (',', 5), ('with', 2), ('a', 1), ('comfortable', 1),
('home', 1), ('and', 3), ('happy', 1), ('disposition', 1), (',', 5), ('seemed', 1), ('to', 1), ('unite', 1), ('some', 1), ('of', 2), ('the', 2), ('best',
1), ('blessings', 1), ('of', 2), ('existence', 1), (';', 1), ('and', 3), ('had', 1), ('lived', 1), ('nearly', 1), ('twenty-one', 1), ('years', 1), ('in',
1), ('the', 2), ('world', 1), ('with', 2)]
```

[20]:
```
import numpy as np
array=np.random.randint(1,100,16)
def outDetection(array):
    sorted(array)
    Q1,Q3=np.percentile(array,[25,75])
    IQR=Q3-Q1
    lr=Q1-(1.5*IQR)
    ur=Q3+(1.5*IQR)
    return lr,ur
lr,ur=outDetection(array)
print(lr,ur)
import seaborn as sns
%matplotlib inline
sns.displot(array)
```

```
-59.375 155.625
```

[20]:
```
<seaborn.axisgrid.FacetGrid at 0x16b23e6c350>
```

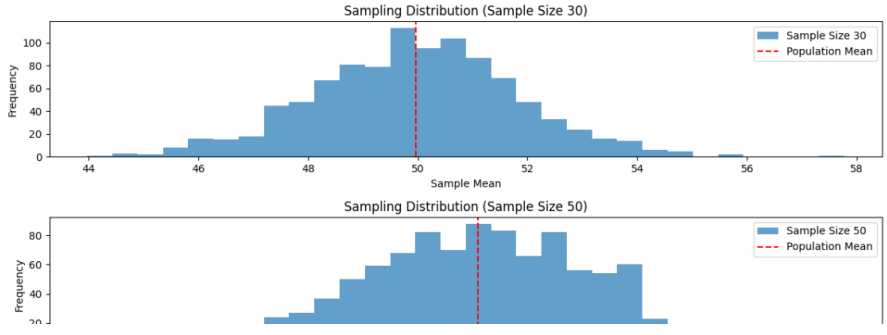


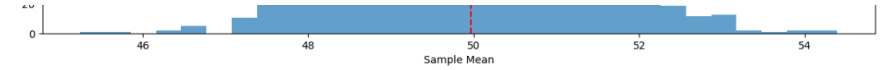
[24]:
```
import numpy as np
import pandas as pd
df=pd.read_csv("Hotel_Dataset.csv")
df.drop_duplicates(inplace=True)
index=np.array(list(range(0,len(df))))
df.set_index(index,inplace=True)
df.drop(['Age_Group.1'],axis=1,inplace=True)
df.Age_Group.unique()
df.Hotel.unique()
df.Hotel.replace(['Ibys'],'Ibis',inplace=True)
df.FoodPreference.unique
df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=True)
df.FoodPreference.replace(['non-Veg'],'Non-Veg',inplace=True)
df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)
df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)
df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)
df.Bill.fillna(round(df.Bill.mean()),inplace=True)
df
```

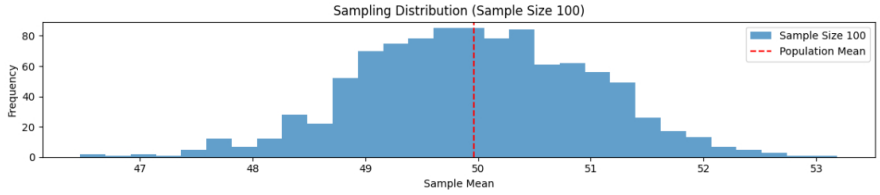
[24]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1	20-25	4	Ibis	Veg	1300	2	40000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000
2	3	25-30	6	RedFox	Veg	1322	2	30000
3	4	20-25	-1	LemonTree	Veg	1234	2	120000
4	5	35+	3	Ibis	Veg	989	2	45000
5	6	35+	3	Ibis	Non-Veg	1909	2	122220
6	7	35+	4	RedFox	Veg	1000	-1	21122
7	8	20-25	7	LemonTree	Veg	2999	-10	345673
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999
9	10	30-35	5	RedFox	Non-Veg	-6755	4	87777

[26]:
```
import numpy as np
import matplotlib.pyplot as plt

# Generate a population
population = np.random.normal(50, 10, 100000)

# Sampling and plotting
sample_sizes = [30, 50, 100]
num_samples = 1000

plt.figure(figsize=(12, 8))
for i, size in enumerate(sample_sizes):
    sample_means = [np.mean(np.random.choice(population, size=size, replace=False)) for _ in range(num_samples)]
    plt.subplot(len(sample_sizes), 1, i+1)
    plt.hist(sample_means, bins=30, alpha=0.7, label=f'Sample Size {size}')
    plt.axvline(np.mean(population), color='red', linestyle='dashed', linewidth=1.5, label='Population Mean')
    plt.title(f'Sampling Distribution (Sample Size {size})')
    plt.xlabel('Sample Mean')
    plt.ylabel('Frequency')
    plt.legend()

plt.tight_layout()
plt.show()
```







[27]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
df = pd.read_csv('E:\\PythonFORDBMS\\Fds\\Salary_data.csv')
df.dropna(inplace=True)
features = df.iloc[:, [0]].values
label = df.iloc[:, [1]].values
x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(x_train, y_train)
train_score = model.score(x_train, y_train)
test_score = model.score(x_test, y_test)
coef = model.coef_
intercept = model.intercept_
print(f"Training Score: {train_score:.2f}")
print(f"Testing Score: {test_score:.2f}")
print(f"Coefficient: {coef[0][0]:.2f}")
print(f"Intercept: {intercept[0]:.2f}")
```

```
Training Score: 0.96
Testing Score: 0.90
Coefficient: 9423.82
Intercept: 25321.58
```

[28]:
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
df = pd.read_csv('E:\\PythonFORDBMS\\Fds\\Social_Network_Ads.csv')
features = df.iloc[:, [2, 3]].values
label = df.iloc[:, 4].values
best_state = None
best_test_score = -np.inf
for i in range(1, 401):
    x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2, random_state=i)
    model = LogisticRegression()
    model.fit(x_train, y_train)
    train_score = model.score(x_train, y_train)
    test_score = model.score(x_test, y_test)
    if test_score > train_score and test_score > best_test_score:
        best_state = i
        best_test_score = test_score
        print(f"Test: {test_score:.2f}, Train: {train_score:.2f}, Random State: {i}")
x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0.2, random_state=best_state)
final_model = LogisticRegression()
final_model.fit(x_train, y_train)

print("Final Model Training Score:", final_model.score(x_train, y_train))
print("Final Model Testing Score:", final_model.score(x_test, y_test))

print("\nClassification Report:")
print(classification_report(label, final_model.predict(features)))
```

```
Test: 0.90, Train: 0.84, Random State: 4
Test: 0.91, Train: 0.83, Random State: 47
Test: 0.93, Train: 0.84, Random State: 61
Test: 0.96, Train: 0.82, Random State: 220
Final Model Training Score: 0.81875
Final Model Testing Score: 0.9625

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.92      0.89       257
           1       0.84      0.71      0.77       143

    accuracy                           0.85       400
   macro avg       0.84      0.82      0.83       400
weighted avg       0.85      0.85      0.84       400
```

[29]:
```python
import numpy as np
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# Generate data
np.random.seed(42)
n = 25
growth_A = np.random.normal(loc=10, scale=2, size=n)
growth_B = np.random.normal(loc=12, scale=3, size=n)
growth_C = np.random.normal(loc=15, scale=2.5, size=n)

# ANOVA test
f_stat, p_val = f_oneway(growth_A, growth_B, growth_C)

# Output
print(f"Means: A={np.mean(growth_A):.2f}, B={np.mean(growth_B):.2f}, C={np.mean(growth_C):.2f}")
print(f"F: {f_stat:.4f}, P: {p_val:.4f}")
if p_val < 0.05:
    print("Reject null: Significant differences between treatments.")
    # Tukey's HSD
    labels = ['A'] * n + ['B'] * n + ['C'] * n
    tukey = pairwise_tukeyhsd(np.concatenate([growth_A, growth_B, growth_C]), labels, alpha=0.05)
    print("\nTukey's HSD Results:")
    print(tukey)
else:
    print("Fail to reject null: No significant differences.")
```

```
Means: A=9.67, B=11.14, C=15.27
F: 36.1214, P: 0.0000
Reject null: Significant differences between treatments.

Tukey's HSD Results:
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower  upper  reject
---------------------------------------------------
     A      B   1.4647 0.0877 -0.1683 3.0977  False
     A      C   5.5923    0.0  3.9593 7.2252   True
     B      C   4.1276    0.0  2.4946 5.7605   True
---------------------------------------------------
```

[30]:
```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Generate a small dataset
```