PRACTICAL JOURNAL

in

STATISTICAL THINKING AND DATA ANALYSIS

Submitted by

KSMSCIT039 MITALI VISHWEKAR

for the award of the Degree of

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY)

PART – II

DEPARTMENT OF INFORMATION TECHNOLOGY
KISHINCHAND CHELLARAM COLLEGE
(Affiliated to University of HSNCU)
MUMBAI, 400020
MAHARASHTRA

2024-25

SUBJECT CODE – BIT604B

STATISTICAL THINKING AND DATA ANALYSIS

KISHINCHAND CHELLARAM COLLEGE

CHURCHGATE, MUMBAI – 400 020.

DEPARTMENT OF INFORMATION TECHNOLOGY

M.SC.I.T PART- II

CERTIFICATE

This is to certify that the Practical conducted by

Mr. **MITALI PRASHANT VISHWEKAR** for M.Sc. (IT) Part- II Semester- III, Seat No: **KSMSCIT039** at Kishinchand Chellaram College in partial fulfillment for the MASTERS OF SCIENCE (INFORMATION TECHNOLOGY). Degree Examination for Semester III has been periodically examined and signed, and the course of term work has been satisfactorily carried out for the year 2024 - 2025. This Practical journal had not been submitted for any other examination and does not form part of any other course undergone by the candidate.

Signature	**Signature**	**Signature**
Lecturer-In-Charge	**External Examiner**	**Course Coordination**
Guided By	**Examined By**	**Certified By**

College Stamp

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

INDEX

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical 1

Aim: - Measure of central tendency and compare the measure of central tendency for two different datasets

Theory: -

Measures of central tendency provide insights into the "average" or "typical" values within a dataset. The three main measures of central tendency are:

1. **Mean**: The arithmetic average of a dataset. It is calculated by summing all values and dividing by the total number of values.
2. **Median**: The middle value when the dataset is ordered. It divides the data into two equal halves. For an even number of values, the median is the average of the two central numbers.
3. **Mode**: The most frequently occurring value in the dataset. A dataset can have multiple modes or no mode if no value repeats.

Each measure has its advantages and is appropriate for different types of data and distributions:

- The **mean** is sensitive to outliers, as extreme values can skew the average.
- The **median** is more robust against outliers, making it suitable for skewed distributions.
- The **mode** is useful for categorical data or to identify common values in a dataset.

Objective: -

To calculate the mean, median, and mode for two different datasets.
To compare the central tendency measures across these datasets to understand their distributions and typical values.

Formula: -

Mean

$$\bar{X} = \frac{\sum_{i=1}^{N} X_i}{N}$$

where X_i represents each data point, and N is the total number of observations.

Median:

- For an ordered dataset with an odd number of values, the median is the middle value.
- For an ordered dataset with an even number of values, the median is the average of the two middle values.

 Mode:

- The mode is the value that appears most frequently in the dataset.

Mean Code: -

```
print("Mitali KSMSCIT 039")

# Calculating Mean by Defining function

def group_mean(classes,frequency):

  midpoints = [(low+high)/2 for low,high in classes]

  # Extract the tuple of frequencies directly

  frequencies = frequency[0]

  total_points = sum(frequencies)

  return sum(f*m for f,m in zip(frequencies,midpoints))/total_points


classes = [(10,20),(20,30),(30,40),(40,50)]

frequency = [(5,10,20,15)]


print("Mean:-",group_mean(classes,frequency))
```

Output: -

```
Mitali KSMSCIT 039
Mean:- 34.0
```

Geometric Mean:-

```
print("Mitali KSMSCIT 039")

# Geometric Mean

from scipy.stats import gmean
```

```
data = [35,39,43,48,24]

print("Geometric Mean:-",gmean(data))
```

Output:-

```
Mitali KSMSCIT 039
Geometric Mean:- 36.81380581782271
```

Hormonic Mean: -

```
print("Mitali KSMSCIT 039")

#hormonic Mean

from scipy.stats import hmean

data = [35,39,43,48,24]

print("Hormonic Mean:-",hmean(data))
```

Output:-

```
Mitali KSMSCIT 039
Hormonic Mean:- 35.72238240507581
```

Median: -

```
print("Mitali KSMSCIT 039")

def group_median(classes,frequency):

  frequency = frequency[0]

  total_points = sum(frequency)

  median_points = total_points//2

  cum_freq = 0

  for i,freq in enumerate(frequency):

    cum_freq += freq
```

```
    if cum_freq > median_points:
      lower_bond, _ = classes[i]
      if i==0:
        c1=0
      else:
        c1 = cum_freq - freq
        class_width = classes[i][1] - classes[i][0]
        return lower_bond + (median_points - c1)/freq * class_width
classes = [(10,20),(20,30),(30,40),(40,50)]
frequency = [(5,10,20,15)]


print("Median:-",group_median(classes,frequency))
```

Output:-

```
Mitali KSMSCIT 039
Median:- 35.0
```

Mode:-

```
print("Mitali KSMSCIT 039")
def group_mode(classes,frequency):
  model_class_index = frequency[0].index(max(frequency[0]))
  lower_bond, _ = classes[model_class_index]
  class_width = classes[model_class_index][1] - classes[model_class_index][0]
  if model_class_index == 0:
    d1 = frequency[0] - 0
```

```
  else:
    d1 = frequency[0][model_class_index] - frequency[0][model_class_index-1]
  if model_class_index == len(frequency)-1:
     d2 = frequency[-1]-0
  else:
     d2 = frequency[0][model_class_index] - frequency[0][model_class_index+1]
  return lower_bond + (d1/(d1+d2))*class_width


classes = [(10,20),(20,30),(30,40),(40,50)]
frequency = [(5,10,20,15)]


print("Mode:-",group_mode(classes,frequency))
```

Output:-

```
Mitali KSMSCIT 039
Mode:- 36.666666666666664
```

Interpretation: -

1. **Mean Comparison**: A higher mean in one dataset compared to another suggests a higher average value. However, if outliers are present, the mean may not accurately represent the dataset's typical value.
2. **Median Comparison**: A higher median in one dataset than another indicates that at least half of the values in the first dataset are greater than half of the values in the second.
3. **Mode Comparison**: Differences in mode can provide insight into commonly occurring values within each dataset, which might be useful for categorical or discrete data.

In summary, comparing the mean, median, and mode for two datasets will reveal not only the central values of each but also offer clues about the distribution's shape and any skewness or outliers present.

Practical 2

Aim: - Hypothesis Testing Using Z-Score

Theory: - Hypothesis testing is a fundamental concept in statistics used to test assumptions (hypotheses) about population parameters based on sample data. In Z-score hypothesis testing, we use the Z-score to determine how many standard deviations a data point is from the mean. This method is particularly useful when we know the population variance or when the sample size is large.

1. **Null Hypothesis (H_0)**: This is the statement we aim to test. Usually, it states that there is no effect or no difference. For example, "The average height of students is 160 cm."
2. **Alternative Hypothesis (H_1)**: This is the opposite of the null hypothesis. It suggests that there is an effect or a difference. For example, "The average height of students is not 160 cm."
3. **Z-Score**: This standardized score represents the number of standard deviations a data point is from the mean. It is used to determine the probability of observing a value within a certain range in a standard normal distribution.
4. **Significance Level (α\alphaα)**: The probability threshold below which we reject the null hypothesis, commonly set at 0.05 (5%).

Objective:-

- To calculate the Z-score and use it to conduct hypothesis testing.
- To determine whether to accept or reject the null hypothesis based on the calculated Z-score and the critical value.
- To interpret the results of hypothesis testing within the context of the data.

Formula: -

Z-Score:

$$Z = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}}$$

where:

- $\bar{X}$: Sample mean
- μ: Population mean (under the null hypothesis)
- σ: Population standard deviation
- n: Sample size

Code: -

```
from scipy.stats import norm
```

```
population_mean=100
population_std=15
sample_mean=108
sample_size=36
print("Mitali KSMSCIT 039")
z_score=(sample_mean-population_mean)/(population_std/sample_size**0.5)
print("Z-score:",z_score)
p_value=2*(1-norm.cdf(abs(z_score)))
print("P-value:",p_value)
alpha=0.05
if p_value<alpha:
   print("Reject the null hypothesis")
else:
   print("Accept the null hypothesis")
```

Output:-

```
KSMSCIT039 Mitali Vishwekar
Z-score: 3.2
P-value: 0.0013742758758317208
Reject the null hypothesis
```

Interpretation:-

Reject H0: If the calculated Z-score lies beyond the critical values, we reject the null hypothesis, suggesting that the sample provides enough evidence to support the alternative hypothesis.

Fail to Reject H0 : If the Z-score falls within the range defined by the critical values, we do not reject the null hypothesis, meaning there is insufficient evidence to support H1H_1H1.

In summary, hypothesis testing using Z-scores allows us to make informed decisions about the population based on sample data by quantifying the probability of observing our results under the null hypothesis.

Practical 3

Aim: - Hypothesis Testing and Visualizing Normal Distribution.

Theory: -

Hypothesis testing with the normal distribution is commonly used in statistical inference to draw conclusions about a population based on sample data. In this process, we test assumptions (hypotheses) by comparing sample data to a theoretical normal distribution.

1. **Normal Distribution**: This is a continuous probability distribution that is symmetric and bell-shaped, characterized by its mean (μ) and standard deviation (σ). Many natural and social phenomena follow a normal distribution, which makes it suitable for hypothesis testing.
2. **Hypothesis Testing**: This involves setting up two opposing hypotheses, the null hypothesis (H0) and the alternative hypothesis (H1), and using sample data to determine which is more likely.
    - **Null Hypothesis (H_0)**: The hypothesis that there is no significant effect or difference.
    - **Alternative Hypothesis (H_1)**: The hypothesis that suggests a significant effect or difference exists.
    - **Significance Level (α)**: The probability threshold for rejecting H0, commonly set at 0.05 (5%).

Objective

To conduct hypothesis testing using sample data and compare it to the normal distribution.
To visualize the normal distribution, including critical regions, to illustrate acceptance and rejection areas based on the significance level.

To interpret the sample data's position within the normal distribution and draw conclusions about the hypotheses.

Formula: -

Z-Score:

$$Z = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}}$$

where:

- $\bar{X}$: Sample mean
- μ: Population mean (under the null hypothesis)
- σ: Population standard deviation
- n: Sample size

Code: -

```
print("Mitali KSMSCIT 039")

from scipy import stats

import matplotlib.pyplot as plt

import numpy as np

# given data

overall_mean = 74.5

overall_std = 8.0

sample_mean = 75.9

sample_size = 200

alpha = 0.05

null_hypo = overall_mean>sample_mean #assuming higher mean at the college(one tail)

alternative_hypo = !null_hypo

z_score = (sample_mean - overall_mean)/(overall_std/np.sqrt(sample_size))

print("Z Score:-",z_score)

pval = 1 - stats.norm.cdf(z_score)

print("P Value:-",pval)

if pval < alpha:

  print("reject")

else:
```

```
 print("accept")
```

Output: -

```
Mitali KSMSCIT 039
Z Score:- 2.4748737341529266
P Value:- 0.006664164390408622
reject
```

Code:-

```
print("Mitali KSMSCIT 039")

# two tail key value

pval2 = 2 * (1 - stats.norm.cdf(abs(z_score)))

print("P Value:-",pval2)

if pval2 < alpha:

 print("reject")

else:

 print("accept")
```

Output: -

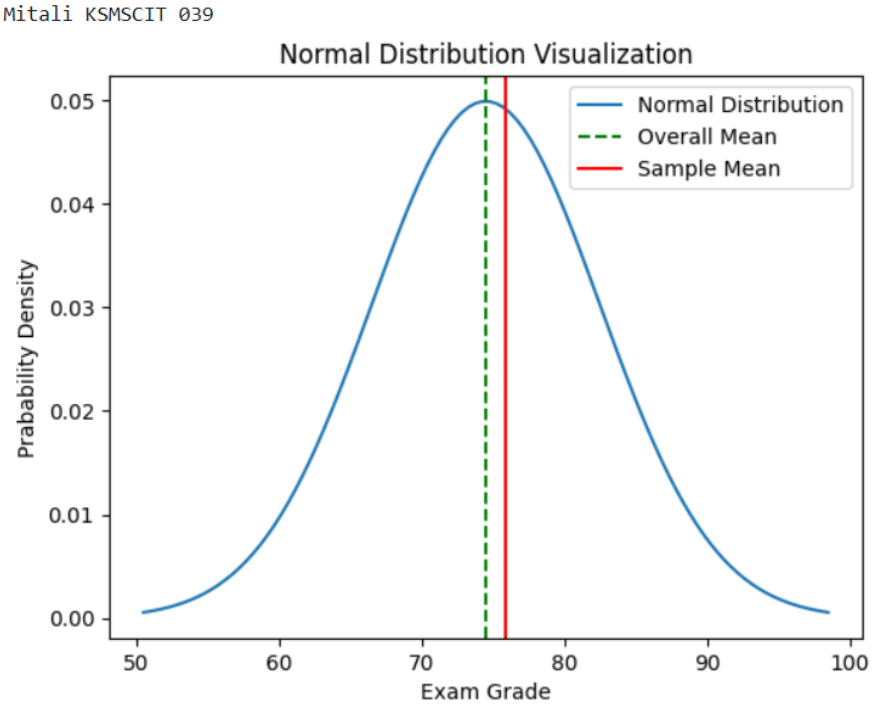
```
Mitali KSMSCIT 039
P Value:- 0.013328328780817245
reject
```

Code:-

```
print("Mitali KSMSCIT 039")

from scipy.stats import norm

x = np.linspace(overall_mean - 3 * overall_std, overall_mean + 3 * overall_std,100)

plt.plot(x, norm.pdf(x,overall_mean,overall_std), label = 'Normal Distribution')

plt.axvline(x = overall_mean,color = 'g',linestyle = '--', label = 'Overall Mean')
```

```
plt.axvline(x = sample_mean,color = 'r',linestyle = '-', label = 'Sample Mean')

plt.xlabel('Exam Grade')

plt.ylabel('Prabability Density')

plt.title('Normal Distribution Visualization')

plt.legend()

plt.show()
```

Output: -



Interpretation

A visualization helps illustrate the probability of observing sample data under H0H_0H0 and provides a clear picture of whether the result falls into an area of significance (critical region) or non-significance (central region).

Practical 4

Aim: - One-Way and Two-Factor ANOVA

Theory: -

ANOVA (Analysis of Variance) is used when comparing the means of two or more groups to see if there is evidence that the group means are significantly different. It works by analyzing the variance within and between groups.

1. **One-Way ANOVA**:
    - Used when we have one independent variable (factor) with multiple levels (groups).
    - Example: Testing whether the average scores differ between three teaching methods.
2. **Two-Factor ANOVA (Two-Way ANOVA)**:
    - Used when we have two independent variables.
    - Example: Testing whether the effects of different teaching methods vary across different student age groups.
    - This allows for the examination of both **main effects** (effects of each factor independently) and **interaction effects** (how the two factors jointly affect the outcome).

Objective:-

To determine if there are statistically significant differences between group means for one or more independent variables.
For Two-Factor ANOVA, to identify both main effects and any interaction effect between the factors.
To interpret and visualize the results to understand the influence of each factor and their interaction.

Formula:-

One-Way ANOVA:

- Total variance is split into:
    - **Between-group variance (SSB)**: Measures the variation between group means.
    - **Within-group variance (SSW)**: Measures the variation within each group.

Two-Factor ANOVA:

- For each factor, we calculate between-group and within-group variances.
- We also calculate an **interaction term** to see if the two factors combined influence the outcome.

- The F-ratio is calculated for each main effect and for the interaction effect.

Interpretation of F-Value and P-Value:

If the calculated F-value is greater than the critical F-value for a chosen significance level (usually 0.05), we reject the null hypothesis, indicating that there is a statistically significant difference.

Code:-

```
print("Mitali KSMSCIT 039")
print("One Way Annova")
import pandas as pd
from scipy.stats import f_oneway
data = {"Subject1":[85,95,98,90,84,88,87],
        "Subject2":[88,93,94,93,80,89,86],
        "Subject3":[83,90,93,94,88,92,84]}
df = pd.DataFrame(data)
df.head()
f,p = f_oneway(df["Subject1"],df["Subject2"],df["Subject3"])
print("F-statistic:",f)
print("p-value:",p)


if p<0.05:
   print("Reject the null hypothesis")
else:
   print("Accept the null hypothesis")
```

Output:-

```
Mitali KSMSCIT 039
One Way Annova
F-statistic: 0.026369168356997975
p-value: 0.9740130140001683
Accept the null hypothesis
```

Code:-

```
print("Mitali KSMSCIT 039")
print("Two Way Annova")
import pandas as pd
from scipy.stats import f_oneway
import statsmodels.api as sm
from statsmodels.formula.api import ols
import numpy as np

data = {'blend':['x','x','x','x','y','y','y','y','z','z','z','z'],
        'crop':['wheat','corn','soya','rice','wheat','corn','soya','rice','wheat','corn','soya','rice'],
        'values':[123,138,110,151,145,165,140,167,156,176,185,175]}
df=pd.DataFrame(data)
print(df)

model=ols('values~C(blend)+C(crop)+C(blend):C(crop)',data=df).fit()
print(model.summary())
```

Output:-

```
Mitali KSMSCIT 039
Two Way Annova
   blend   crop  values
0      x  wheat     123
1      x   corn     138
2      x   soya     110
3      x   rice     151
4      y  wheat     145
5      y   corn     165
6      y   soya     140
7      y   rice     167
8      z  wheat     156
9      z   corn     176
10     z   soya     185
11     z   rice     175
```

OLS Regression Results

Dep. Variable:	values	R-squared:	1.000
Model:	OLS	Adj. R-squared:	nan
Method:	Least Squares	F-statistic:	nan
Date:	Fri, 11 Oct 2024	Prob (F-statistic):	nan
Time:	11:18:59	Log-Likelihood:	344.14
No. Observations:	12	AIC:	-664.3
Df Residuals:	0	BIC:	-658.5
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	138.0000	inf	0	nan	nan	nan
C(blend)[T.y]	27.0000	inf	0	nan	nan	nan
C(blend)[T.z]	38.0000	inf	0	nan	nan	nan
C(crop)[T.rice]	13.0000	inf	0	nan	nan	nan
C(crop)[T.soya]	-28.0000	inf	-0	nan	nan	nan
C(crop)[T.wheat]	-15.0000	inf	-0	nan	nan	nan
C(blend)[T.y]:C(crop)[T.rice]	-11.0000	inf	-0	nan	nan	nan
C(blend)[T.z]:C(crop)[T.rice]	-14.0000	inf	-0	nan	nan	nan
C(blend)[T.y]:C(crop)[T.soya]	3.0000	inf	0	nan	nan	nan
C(blend)[T.z]:C(crop)[T.soya]	37.0000	inf	0	nan	nan	nan
C(blend)[T.y]:C(crop)[T.wheat]	-5.0000	inf	-0	nan	nan	nan
C(blend)[T.z]:C(crop)[T.wheat]	-5.0000	inf	-0	nan	nan	nan

Omnibus:	0.507	Durbin-Watson:	0.411
Prob(Omnibus):	0.776	Jarque-Bera (JB):	0.526
Skew:	-0.370	Prob(JB):	0.769
Kurtosis:	2.291	Cond. No.	17.9

Interpretation

1. **One-Way ANOVA**:
    - **Significant Result**: A significant F-value suggests that at least one group mean is significantly different from the others.
    - **Non-Significant Result**: Failing to reject the null hypothesis implies there is no evidence that the group means differ significantly.
2. **Two-Factor ANOVA**:
    - **Main Effects**: Significant main effects for each factor indicate that the factor influences the outcome independently of the other factor
    - **Non-Significant Results**: If neither main effects nor interaction effects are significant, it suggests that neither factor nor their interaction has a significant influence on the outcome.

In summary, One-Way and Two-Factor ANOVA help us understand whether group means differ significantly and whether two factors have independent or interacting effects on the outcome. Visualization of results, such as interaction plots, can further clarify these effects

Practical 5

Aim: - Linear Regression Analysis

Theory:-

Linear regression is a statistical method used to model the relationship between a dependent variable (outcome) and one or more independent variables (predictors). When there is only one independent variable, it is called **Simple Linear Regression**; with more than one, it is called **Multiple Linear Regression**.

1. **Simple Linear Regression**:
    - Models the relationship between a single independent variable (X) and a dependent variable (Y).
2. **Multiple Linear Regression**:
    - Extends the model to include multiple independent variables.

Objective:-

To determine the strength and nature of the relationship between the independent and dependent variables.
To develop a predictive model based on the identified relationships.
To assess the fit of the regression model and interpret the significance of the coefficients.

Formula:-

Simple Linear Regression is:

$Y=mx+c$

where:

- Y is the dependent variable (predicted outcome).
- X is the independent variable (predictor).
- m is the slope of the line (rate of change in Y for a one-unit change in x).
- C is the intercept (the value of Ywhen x=0).

Code:-

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

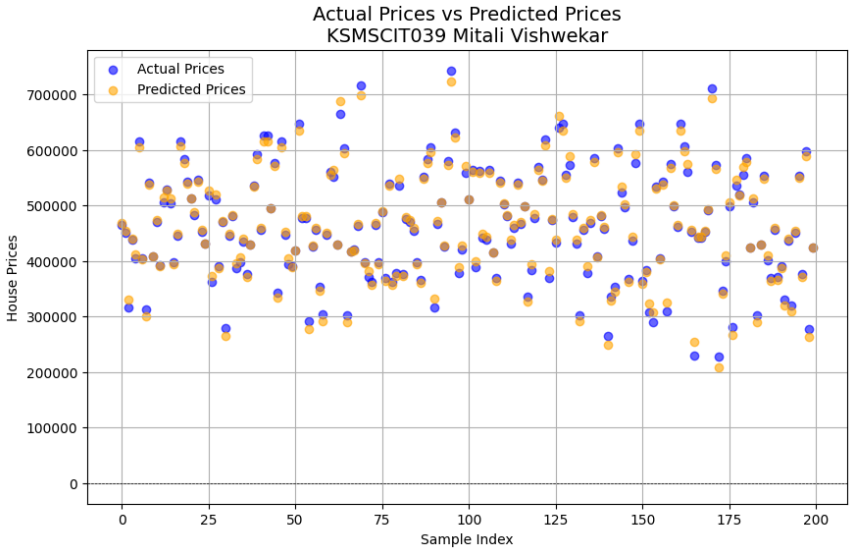
```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

np.random.seed(42)

sizes = np.random.normal(2000, 500, 1000).astype(int)

bedrooms = np.random.choice([1, 2, 3, 4, 5], 1000)

bathrooms = np.random.choice([1, 2, 3], 1000)

ages = np.random.normal(15, 10, 1000).astype(int)

locations = np.random.choice(['Urban', 'Suburban'], 1000)

def calculate_price(size, bedrooms, bathrooms, age, location):

    base_price = 50000

    price = base_price + size * 150 + bedrooms * 25000 + bathrooms * 12000 - age * 2000

    return price * (1.2 if location == 'Urban' else 1.0)

prices = [calculate_price(sizes[i], bedrooms[i], bathrooms[i], ages[i], locations[i]) for i in
range(1000)]

# Create DataFrame

house_data = pd.DataFrame({

    'Size': sizes,

    'Bedrooms': bedrooms,

    'Bathrooms': bathrooms,

    'Age': ages,

    'Location': locations,

    'Price': prices
```

```
})
# Linear Regression
house_data = pd.get_dummies(house_data, columns=['Location'], drop_first=True)
X = house_data.drop('Price', axis=1)
y = house_data['Price']
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)
# Evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mitali KSMSCIT 039")
print("\nModel Coefficients:")
for i, col in enumerate(X.columns):
    print(f"{col}: {model.coef_[i]}")
print("\nMean Squared Error:", mse)
print("R² Score:", r2)
# Visualization
plt.figure(figsize=(10, 6))
plt.scatter(np.arange(len(y_test)), y_test, color='blue', label='Actual Prices', alpha=0.6)
plt.scatter(np.arange(len(y_pred)), y_pred, color='orange', label='Predicted Prices', alpha=0.6)
```

```
plt.xlabel("Sample Index")

plt.ylabel("House Prices")

plt.title("Actual Prices vs Predicted Prices\nKSMSCIT011 Daniel Injeti", fontsize=14)

plt.axhline(0, color='black', linewidth=0.5, ls='--')

plt.legend()

plt.grid()

plt.show()
```

Output:-

```
Mitali KSMSCIT 039

Model Coefficients:
Size: 164.70358883573502
Bedrooms: 27503.061615436363
Bathrooms: 13745.384725040047
Age: -2153.0379636108264
Location_Urban: 84304.66431385632

Mean Squared Error: 67096111.61412644
R² Score: 0.9932709374803655
```



Interpretation

Linear regression analysis provides a powerful method for understanding and quantifying relationships between variables, allowing for predictions and insights into the strength and direction of these relationships. Visualizations like scatter plots with regression lines can further aid in interpreting the model's results

Practical 6

Aim: - Pearson correlation and Spearman rank correlation

Theory

Correlation measures the strength and direction of a relationship between two variables. There are two common types:

1. **Pearson Correlation**:
    - Measures the linear relationship between two continuous variables.
    - The Pearson correlation coefficient (r) ranges from -1 to +1:
        - r=+1: Perfect positive linear relationship.
        - r=−1: Perfect negative linear relationship.
        - r=0 : No linear relationship.
    - Assumes both variables are continuous, normally distributed, and linearly related.
2. **Spearman Rank Correlation**:
    - Measures the monotonic relationship between two variables using their ranked values.
    - The Spearman correlation coefficient (ρ) also ranges from -1 to +1, with interpretations similar to Pearson correlation.
    - Useful for ordinal data or when assumptions of Pearson correlation (such as linearity or normality) are not met, making it a non-parametric measure.

Objective:-

To determine the strength and direction of the relationship between two variables using Pearson and Spearman correlations.
To understand whether the relationship is linear (using Pearson) or monotonic (using Spearman).
To interpret the correlation coefficients to draw meaningful insights.

Code:-

```
import numpy as np

from scipy.stats import pearsonr, spearmanr

X = np.array([43, 21, 25, 42, 57, 59, 63, 55, 47, 58])

Y = np.array([99, 65, 79, 75, 87, 81, 109, 92, 80, 103])

pearson_corr, pearson_p_value = pearsonr(X, Y)

spearman_corr, spearman_p_value = spearmanr(X, Y)

print("Pearson Correlation Coefficient:")
```

```
print(f"Coefficient: {pearson_corr:.3f}, P-value: {pearson_p_value:.3f}")
print("\nSpearman Rank Correlation Coefficient:")
print(f"Coefficient: {spearman_corr:.3f}, P-value: {spearman_p_value:.3f}")
if pearson_p_value < 0.05:
    print("\nPearson correlation is statistically significant.")
else:
    print("\nPearson correlation is not statistically significant.")
if spearman_p_value < 0.05:
    print("Spearman correlation is statistically significant.")
else:
    print("Spearman correlation is not statistically significant.")
print("Mitali KSMSCIT 039")
```

Output:-

```
Pearson Correlation Coefficient:
Coefficient: 0.704, P-value: 0.023

Spearman Rank Correlation Coefficient:
Coefficient: 0.770, P-value: 0.009

Pearson correlation is statistically significant.
Spearman correlation is statistically significant.
Mitali KSMSCIT 039
```

Interpretation

Pearson Correlation (r):

- **r>0**: Positive linear relationship (as X increases, Y also tends to increase).

Spearman Rank Correlation (ρ):

- **ρ>0\rho > 0ρ>0**: Positive monotonic relationship (as XXX increases, YYY tends to increase in a ranked order).

Pearson correlation quantifies linear relationships, while Spearman captures monotonic relationships, making each suitable under different data conditions.

Practical 7

Aim: - Latin Square design

Theory

Latin Square Design is a statistical design used in experiments where the researcher aims to control for variability in two different directions (such as rows and columns), in addition to the primary factor being studied. It is especially useful in situations where there are three factors, with the third factor being the primary treatment or condition of interest.

In a Latin Square design:

- The design is arranged in a square grid format, where each row and each column represent a blocking factor (confounding variable), and each cell of the grid represents a unique combination of row and column treatments.
- Each treatment appears exactly once in each row and each column, ensuring balanced distribution and minimizing biases associated with the rows or columns.

For example, if we have 3 factors to control—let's say treatment type, day of the week, and testing time—then the layout of the design will ensure each treatment occurs once per day and once per testing time.

Objective:-

To control for two sources of variation in addition to the primary treatment, reducing confounding effects.
To improve the accuracy of experimental results by balancing the treatment distribution across the row and column blocking factors.
To determine whether the primary treatment has a significant effect on the outcome while accounting for the blocking factors.

Code:-

```
import numpy as np

import pandas as pd

# Define the treatments, subjects, and seasons

treatments = ['Drip Irrigation', 'Sprinkler Irrigation', 'Surface Irrigation']

crops = ['Crop 1', 'Crop 2', 'Crop 3']

seasons = ['Spring', 'Summer', 'Fall']

# Create a 3x3 Latin Square matrix
```

```
latin_square = np.array([
    [treatments[0], treatments[1], treatments[2]],  # Spring
    [treatments[1], treatments[2], treatments[0]],  # Summer
    [treatments[2], treatments[0], treatments[1]],  # Fall
])
# Create a DataFrame for the Latin Square Design
experiment_design = pd.DataFrame(latin_square, columns=crops, index=seasons)
print("Mitali KSMSCIT 039")
# Display the experiment design
print("Latin Square Design:")
print(experiment_design)
# Data collection: heights of crops after one month (in cm)
data = {
    'Crop 1': [25, 30, 28],  # Growth corresponding to the design
    'Crop 2': [35, 32, 30],
    'Crop 3': [29, 33, 31]
}
# Create a DataFrame for the collected data
growth_data = pd.DataFrame(data, index=seasons)
print("\nCollected Crop Growth Data (cm):")
print(growth_data)
import statsmodels.api as sm
from statsmodels.formula.api import ols
# Reshape the data for ANOVA
growth_data_melted = growth_data.reset_index().melt(id_vars='index', var_name='Crop',
value_name='Growth')
growth_data_melted.columns = ['Season', 'Crop', 'Growth']
# Fit the ANOVA model
model = ols('Growth ~ C(Crop) + C(Season)', data=growth_data_melted).fit()
```

```
anova_table = sm.stats.anova_lm(model, typ=2)

# Display the ANOVA results

print("\nANOVA Results:")

print(anova_table)
```

Output:-

```
Mitali KSMSCIT 039
Latin Square Design:
                     Crop 1                Crop 2                Crop 3
Spring       Drip Irrigation  Sprinkler Irrigation    Surface Irrigation
Summer  Sprinkler Irrigation    Surface Irrigation       Drip Irrigation
Fall      Surface Irrigation       Drip Irrigation  Sprinkler Irrigation

Collected Crop Growth Data (cm):
        Crop 1  Crop 2  Crop 3
Spring      25      35      29
Summer      30      32      33
Fall        28      30      31

ANOVA Results:
              sum_sq   df         F    PR(>F)
C(Crop)    34.666667  2.0  2.736842  0.178272
C(Season)   8.000000  2.0  0.631579  0.577600
Residual   25.333333  4.0       NaN       NaN
```

Interpretation

In a Latin Square design, significant row or column effects indicate that the blocking factors (e.g., time or location) influence the outcome, confirming the need to control for them. A significant treatment effect means the primary treatment has a meaningful impact on the outcome. Low error suggests the design successfully controls variability, making the results more reliable and reducing bias.

Practical 8

Aim: - Time Series Analysis

A) Trend Analysis

Theory: -

A trend is the long-term movement or direction in the data over time. It can be upward, downward, or stable, indicating a general tendency in the data.

Objective: -

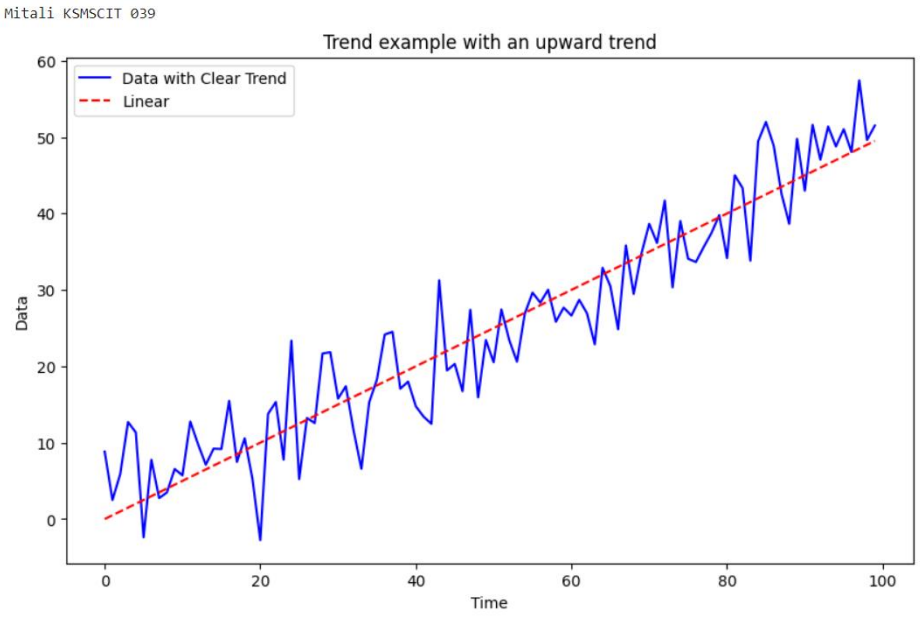
To identify and quantify the direction and nature of the trend in the data over a specified period.

Code:-

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

#Generate the data with a clear upward trend

np.random.seed(0)

time= np.arange(100)

trend= 0.5*time

noise= np.random.normal(0,5,size=100)

data = trend + noise

df_trend = pd.DataFrame({'Time':time,'data':data})

print("Mitali KSMSCIT 039")

plt.figure(figsize=(10,6))

plt.plot(df_trend['Time'],df_trend['data'],label='Data with Trend', color='blue')

plt.plot(df_trend['Time'],trend,label='Linear',linestyle='--',color='red')
```

```
plt.title('Trend Example with an upward trend')

plt.xlabel('Time')

plt.ylabel('Data')

plt.legend()

plt.show()
```

Output: -



Interpretation: A positive slope b indicates an upward trend, while a negative b suggests a downward trend. No slope (b=0) indicates no trend.

B) Seasonal Variation

Theory: Seasonal variation refers to regular, repeating patterns within a fixed period, such as days, weeks, months, or quarters, caused by seasonal factors.

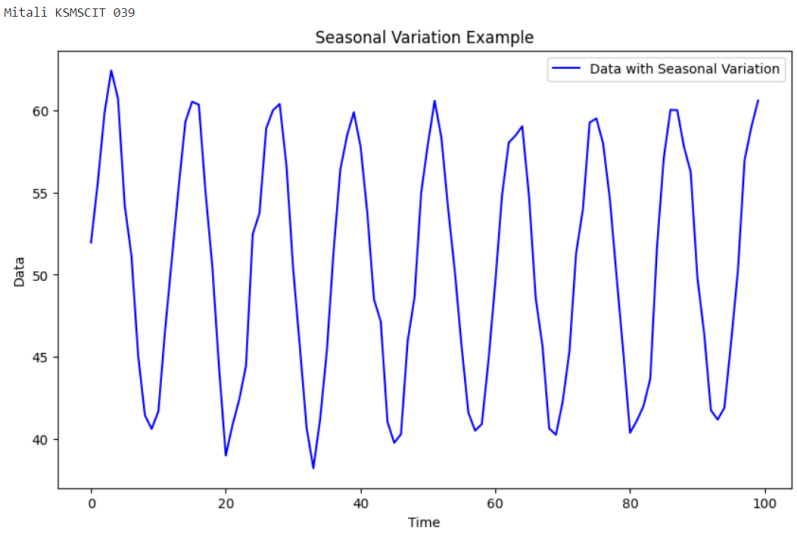
Objective: To capture the repetitive short-term fluctuation in the time series that occurs due to seasonality.

Formula: Seasonal component can often be isolated by averaging or through Fourier series for more complex patterns.

Code:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#Generate the data with a clear seasonal variation
np.random.seed(0)
time = np.arange(0,100)
seasonality = 10*np.sin(2*np.pi*time/12)
noise = np.random.normal(0.2,size=100)
data = 50 + seasonality + noise
#Create a dataframe
df_seasonal = pd.DataFrame({'Time':time,'data':data})
print("Mitali KSMSCIT 039")
plt.figure(figsize=(10,6))
plt.plot(df_seasonal['Time'],df_seasonal['data'],label='Data with Seasonal Variation',color='blue')
plt.plot(df_seasonal['Time'],seasonality,label='Seasonal Variation',linestyle='--',color='red')
plt.title('Seasonal Variation Example')
plt.xlabel('Time')
plt.ylabel('Data')
plt.legend()
plt.show()
```

Output: -



Interpretation: High seasonal variation may indicate periodic demand peaks, while low variation suggests a more stable time series without strong seasonal components.

C) Cyclic Variation

Theory: Cyclic variation is the fluctuation in data over periods longer than a year, typically tied to economic, business, or other cycles.

Objective: To identify and analyze cycles that affect the time series over medium- to long-term durations, distinct from seasonal effects.

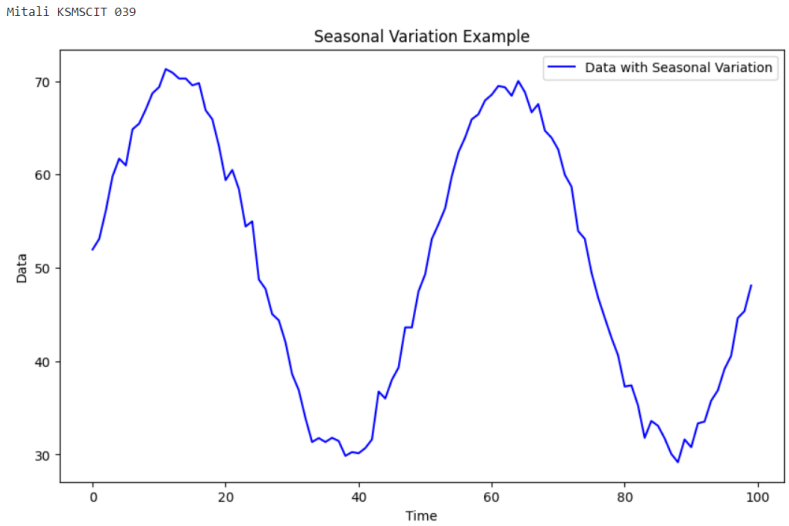
Formula: Cycles can be analyzed using methods like moving averages or by fitting models that account for cyclic patterns.

Code:-

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

#Generate the data with a clear seasonal variation

np.random.seed(0)

time = np.arange(0,100)
```

```
cycle = 20*np.sin(2*np.pi*time/50)
noise = np.random.normal(0.2,size=100)
data = 50 + cycle + noise
#Create a dataframe
df_cycle = pd.DataFrame({'Time':time,'data':data})
print("Mitali KSMSCIT 039")
plt.figure(figsize=(10,6))
plt.plot(df_cycle['Time'],df_cycle['data'],label='Data with Seasonal Variation',color='blue')
plt.title('Seasonal Variation Example')
plt.xlabel('Time')
plt.ylabel('Data')
plt.legend()
plt.show()
```

Output:-



Interpretation: Cyclic behavior indicates the presence of larger, recurrent periods of highs and lows that are not as regular as seasonal patterns.

D) Irregular Variation

Theory: Irregular variation (or noise) represents random, unpredictable fluctuations in a time series.

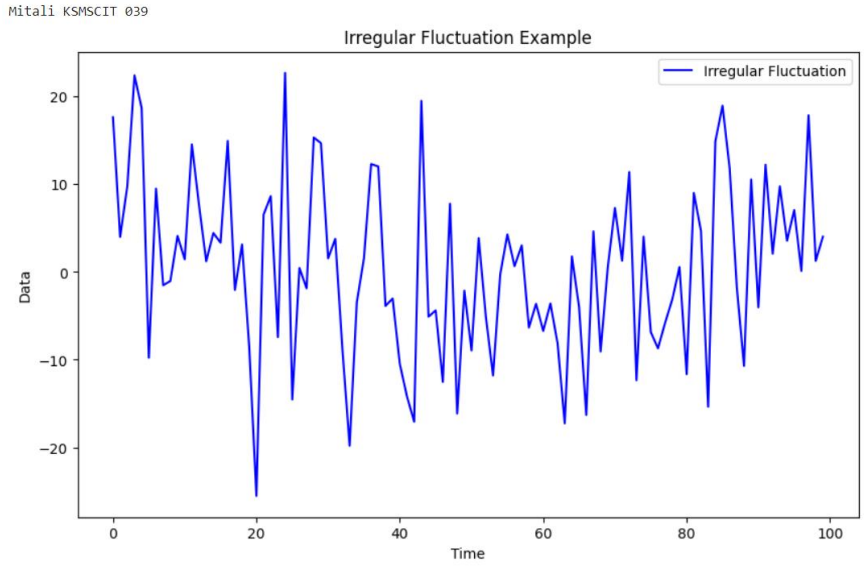
Objective: To isolate and account for the erratic, unsystematic changes that do not follow any of the trends, seasonality, or cyclic patterns.

Formula: Residuals from a fitted model (such as the trend and seasonality) can be considered the irregular component

Code:-

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

np.random.seed(0)

time = np.arange(0,100)

noise = np.random.normal(0,10,size=100)

df_irregular = pd.DataFrame({'Time':time,'data':noise})


print("Mitali KSMSCIT 039")


plt.figure(figsize=(10,6))

plt.plot(df_irregular['Time'],df_irregular['data'],label='Data with Seasonal Variation',color='blue')

plt.xlabel('Time')

plt.ylabel('Data')

plt.title('Irregular Fluctuation Example')

plt.legend()

plt.show()
```

Output:-



Interpretation: High irregular variation may signal external shocks or noise, which is usually minimized or ignored when making forecasts.

E) Decomposition of Time Series

Theory: Time series decomposition involves breaking down a time series into its component parts, typically into trend (TtT_tTt), seasonal (StS_tSt), cyclic (CtC_tCt), and irregular (ItI_tIt) components.

Objective: To isolate and understand each element of the time series for improved forecasting and insights.

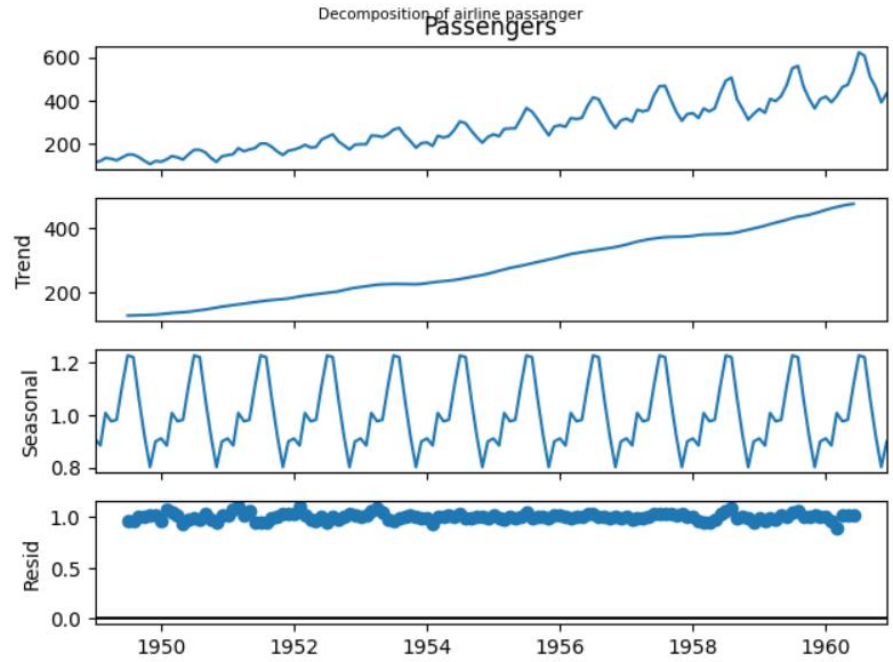
Code:-

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.seasonal import seasonal_decompose

data = pd.read_csv('airline-passengers.csv', index_col='Month', parse_dates=True)

#decompose the time series into trend, seasonal and residual component

result = seasonal_decompose(data['Passengers'],model='multiplicative')
```

```
print("Mitali KSMSCIT 039")

result.plot()

plt.suptitle('Decomposition of Airline Passenger',fontsize=7.5)

plt.show()
```

Output:-

```
Mitali KSMSCIT 039
```



Interpretation: Decomposition helps in recognizing which components drive the data, allowing for better forecasting by focusing on individual components.

Practical 9

Aim: -Time Series Analysis and Forecasting

A. Stock Price Analysis

Theory:

Stock prices are highly volatile and affected by various market factors, investor sentiments, and economic indicators. Despite this volatility, trends and patterns may emerge over time in historical stock price data, providing a basis for short-term and, to some extent, long-term forecasting.

Objective:

To use time series techniques to analyze and forecast stock prices, identifying possible trends, seasonality, or cyclic behaviors that may guide investment decisions or risk management.

Code: -

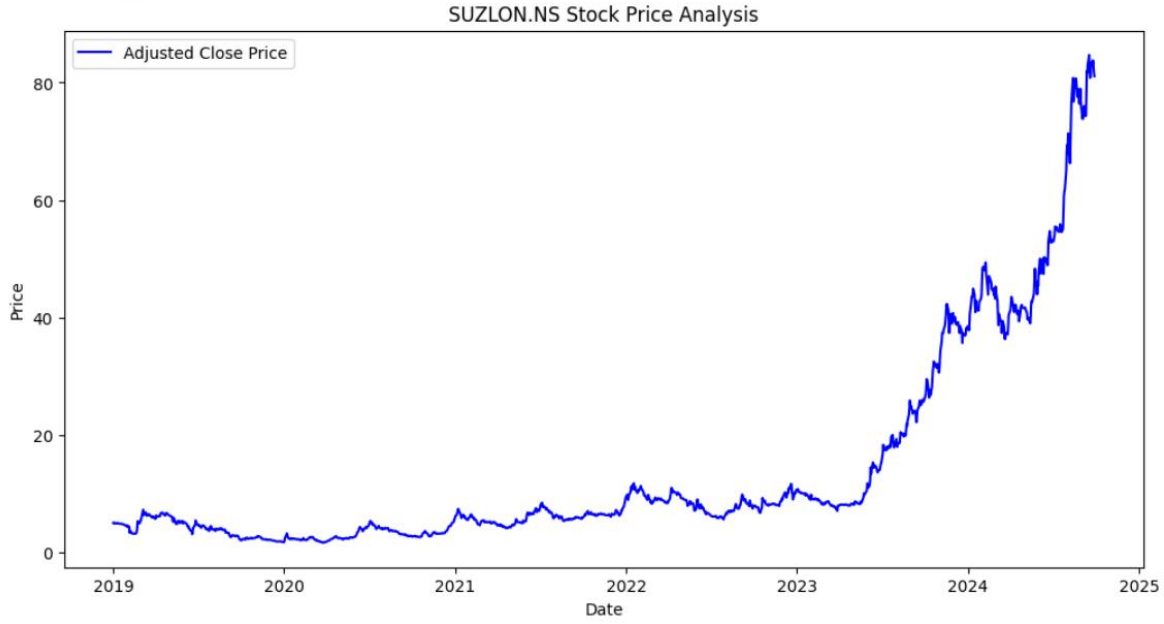
```
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose


stock_symbol = 'SUZLON.NS'
stock_data = yf.download(stock_symbol,start='2019-01-01', end='2024-09-28')
print("Mitali KSMSCIT 039")
print(stock_data.head())
print(stock_data.tail())
print("Mitali KSMSCIT 039")
plt.figure(figsize=(12,6))
plt.plot(stock_data.index, stock_data['Adj Close'], label='Adjusted Close Price', color='blue')
plt.title('Stock Price Analysis')
plt.xlabel('Date')
```

```
plt.ylabel('Adjusted Close Price')
plt.legend()
plt.show()

stock_data_monthly = stock_data['Adj Close'].resample('M').mean()
decompose_result = seasonal_decompose(stock_data_monthly, model='multiplicative')
print("Mitali KSMSCIT 039")
decompose_result.plot()
plt.suptitle('Decomposition of Stock Price',fontsize=10)
plt.show()

#Calculate Simple moving average of 15 - 200 days

stock_data['SMA_50'] = stock_data['Adj Close'].rolling(window=15).mean()
stock_data['SMA_200'] = stock_data['Adj Close'].rolling(window=200).mean()
print("Mitali KSMSCIT 039")
plt.figure(figsize=(12,6))
plt.plot(stock_data.index, stock_data['Adj Close'], label='Adjusted Close Price', color='blue')
plt.plot(stock_data.index, stock_data['SMA_50'], label='SMA_50', color='green')
plt.plot(stock_data.index, stock_data['SMA_200'], label='SMA_200', color='red')
plt.title('Stock Price Analysis with Moving Average')
plt.xlabel('Date')
plt.ylabel('Adjusted Close Price')
plt.legend()
plt.show()
```

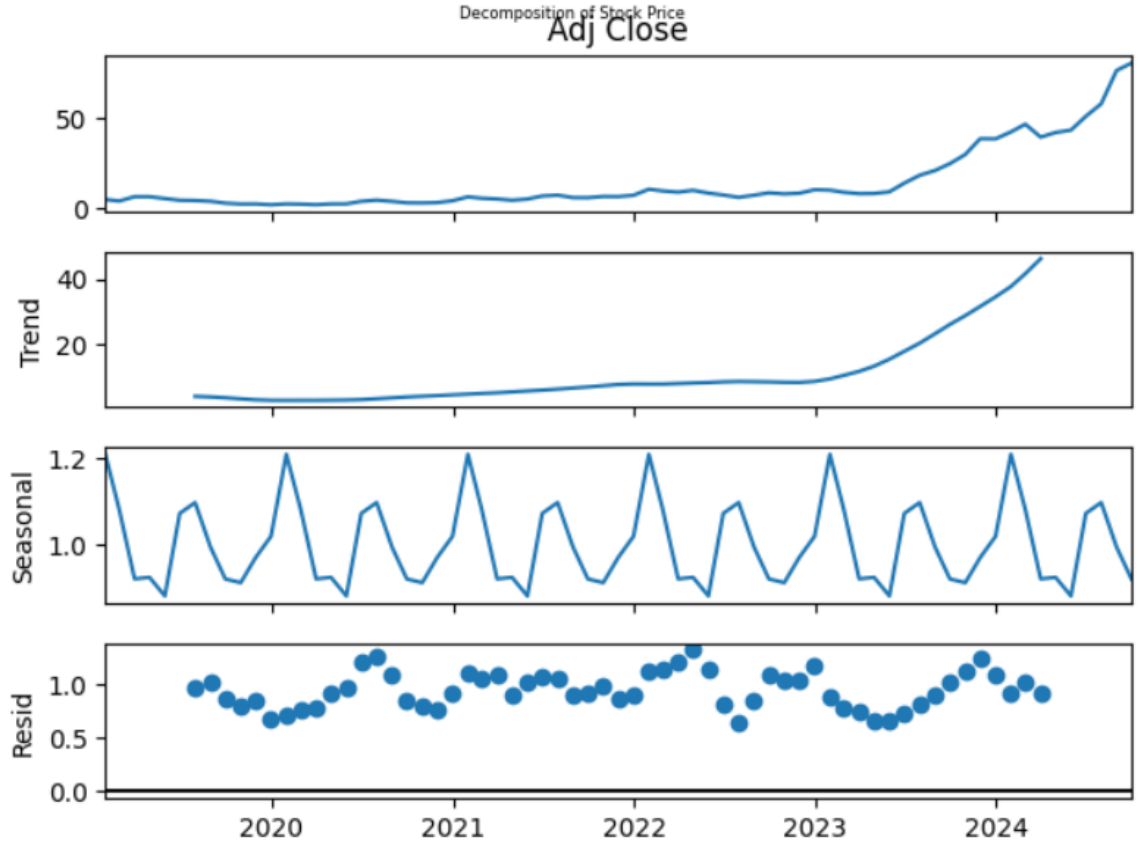
Output:-

```
[*********************100%***********************]  1 of 1 completed
Mitali KSMSCIT 039
                 Open      High       Low     Close  Adj Close    Volume
Date
2019-01-01  5.004266  5.050176  4.912444  5.004266   5.004266  10679462
2019-01-02  5.004266  5.004266  4.866534  4.912444   4.912444  14752240
2019-01-03  4.958355  4.958355  4.866534  4.866534   4.866534  10516981
2019-01-04  4.912444  5.050176  4.728801  5.004266   5.004266  37736957
2019-01-07  5.050176  5.096087  4.866534  4.912444   4.912444  25786558
                Open       High        Low      Close  Adj Close    Volume
Date
2024-09-23  84.190002  84.989998  82.699997  82.889999  82.889999  35307207
2024-09-24  83.000000  84.199997  82.320000  83.790001  83.790001  34414038
2024-09-25  83.699997  83.699997  82.500000  82.959999  82.959999  27040040
2024-09-26  82.949997  83.360001  81.500000  81.839996  81.839996  24929699
2024-09-27  81.750000  82.769997  80.500000  81.089996  81.089996  42523359
```

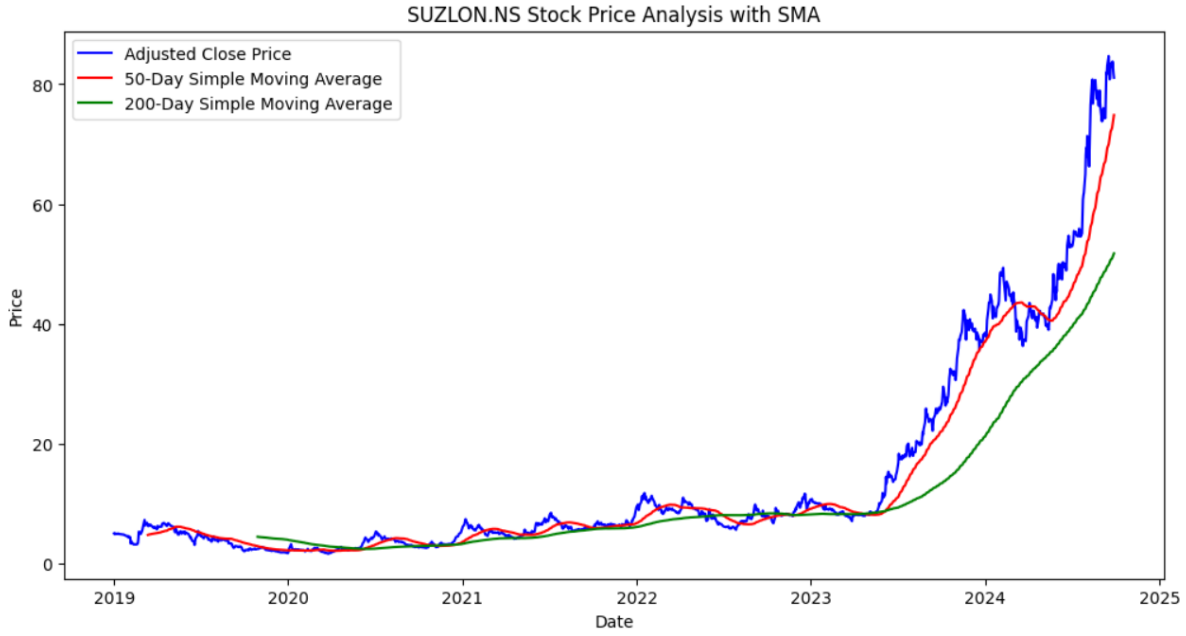
Mitali KSMSCIT 039



```
Mitali KSMSCIT 039
```




```
Mitali KSMSCIT 039
<matplotlib.legend.Legend at 0x7eea2f6c0250>
```



B. Rain Forecasting Using Trend and Seasonal

Theory:

Rainfall often follows seasonal patterns, with certain times of the year receiving more rainfall due to climate conditions. Long-term climate data may also show trends related to climate change or natural cycles, making time series analysis effective for rainfall forecasting.

Objective:

To analyze historical rainfall data to identify seasonal patterns and trends, enabling the forecasting of future rainfall. This aids in water resource planning, agriculture, and flood management.

Code: -

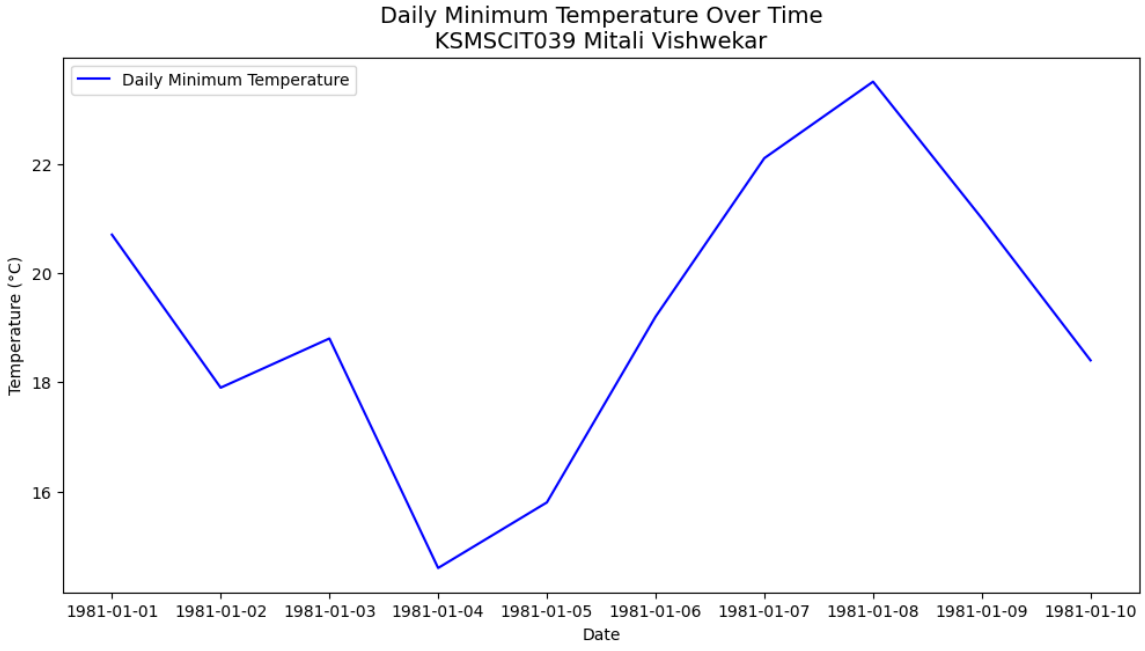
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
# Load the dataset
data = {
    'Date': [
        '1981-01-01', '1981-01-02', '1981-01-03', '1981-01-04', '1981-01-05',
        '1981-01-06', '1981-01-07', '1981-01-08', '1981-01-09', '1981-01-10'
    ],
    'Temp': [
        20.7, 17.9, 18.8, 14.6, 15.8,
        19.2, 22.1, 23.5, 21.0, 18.4
    ]
}
# Create a DataFrame
df = pd.DataFrame(data)
```

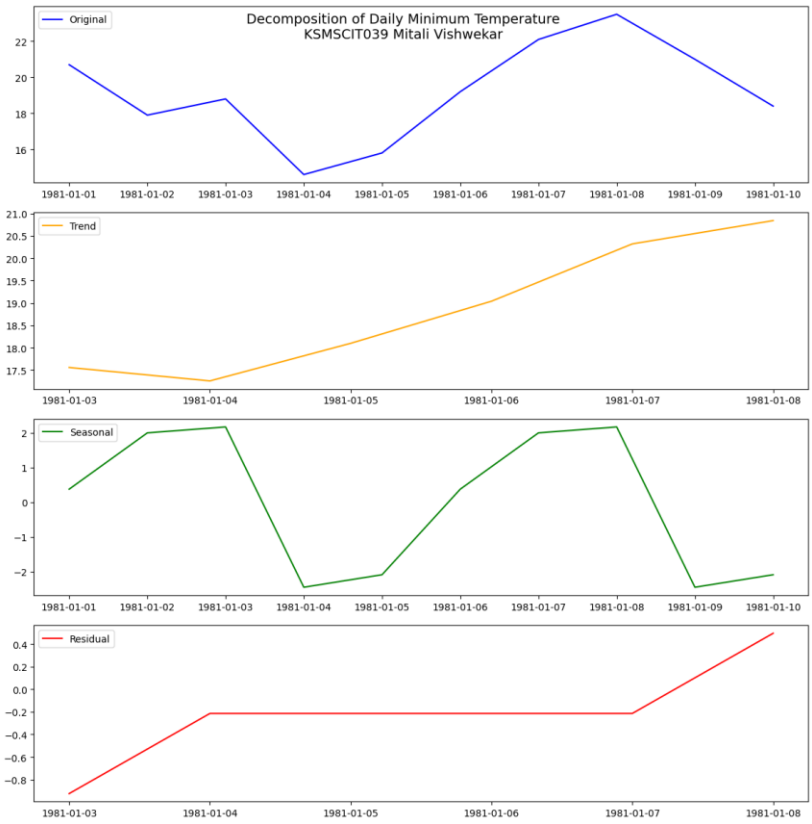
```
# Data Preprocessing
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
# Display the first few rows of the dataset
print("Mitali KSMSCIT 039")
print(df.head())
print(df.tail())
# Trend Analysis
plt.figure(figsize=(12, 6))
plt.plot(df['Temp'], label='Daily Minimum Temperature', color='blue')
plt.title('Daily Minimum Temperature Over Time\nKSMSCIT011 Daniel Injeti', fontsize=14)
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.legend()
plt.show()
# Seasonal Decomposition with period of 5
decomposition = seasonal_decompose(df['Temp'], model='additive', period=5)  # Adjusted period
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid
# Plot the decomposition
plt.figure(figsize=(12, 12))
plt.subplot(4, 1, 1)
plt.plot(df['Temp'], label='Original', color='blue')
plt.legend(loc='upper left')
plt.subplot(4, 1, 2)
plt.plot(trend, label='Trend', color='orange')
plt.legend(loc='upper left')
```

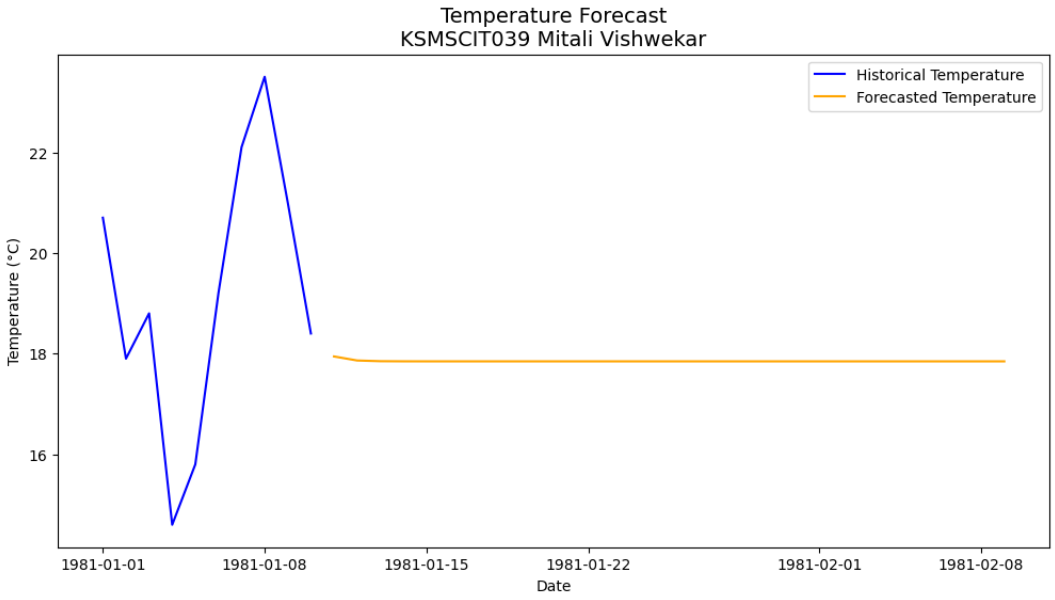
```
plt.subplot(4, 1, 3)
plt.plot(seasonal, label='Seasonal', color='green')
plt.legend(loc='upper left')


plt.subplot(4, 1, 4)
plt.plot(residual, label='Residual', color='red')
plt.legend(loc='upper left')
plt.tight_layout()
plt.suptitle('Decomposition of Daily Minimum Temperature\nKSMSCIT011 Daniel Injeti', fontsize=14)
plt.show()
# Forecasting with ARIMA
model = ARIMA(df['Temp'].dropna(), order=(1, 1, 0))  # Adjust parameters as needed
model_fit = model.fit()
# Generate forecast for the next 30 days
forecast = model_fit.forecast(steps=30)
# Prepare forecast dates
forecast_dates = pd.date_range(start=df.index[-1] + pd.Timedelta(days=1), periods=30)
# Visualization of forecast results
plt.figure(figsize=(12, 6))
plt.plot(df['Temp'], label='Historical Temperature', color='blue')
plt.plot(forecast_dates, forecast, label='Forecasted Temperature', color='orange')
plt.title('Temperature Forecast\nKSMSCIT011 Daniel Injeti', fontsize=14)
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.legend()
plt.show()
```

Output: -

```
Mitali KSMSCIT 039
            Temp
Date
1981-01-01  20.7
1981-01-02  17.9
1981-01-03  18.8
1981-01-04  14.6
1981-01-05  15.8
            Temp
Date
1981-01-06  19.2
1981-01-07  22.1
1981-01-08  23.5
1981-01-09  21.0
1981-01-10  18.4
```




Decomposition of Daily Minimum Temperature
KSMSCIT039 Mitali Vishwekar
Original
Trend
Seasonal
Residual


Temperature Forecast
KSMSCIT039 Mitali Vishwekar
Historical Temperature
Forecasted Temperature
Temperature (°C)
Date

Practical 10

Aim: - Sales Forecasting Using SARIMAX

Theory:

SARIMAX extends the ARIMA model by adding seasonal components and exogenous variables (e.g., holidays, promotions) that impact sales. It is ideal for capturing repeating patterns and external influences on sales.

Objective:

To produce accurate sales forecasts by modeling internal patterns and the effects of external factors.

Code:-

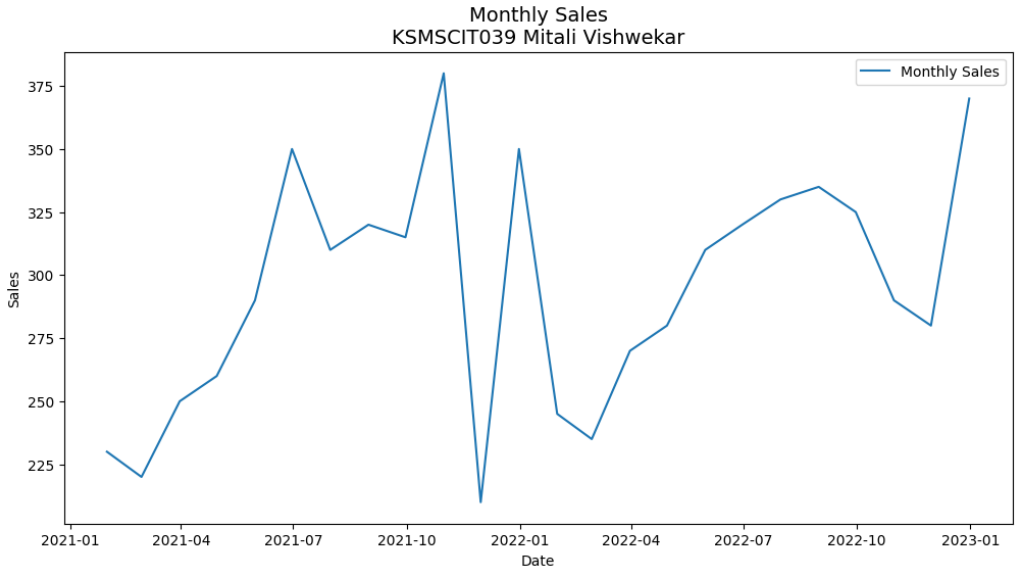
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.seasonal import seasonal_decompose

from statsmodels.tsa.arima.model import ARIMA

# Load the dataset

data = {

    'Date': [

        '1981-01-01', '1981-01-02', '1981-01-03', '1981-01-04', '1981-01-05',

        '1981-01-06', '1981-01-07', '1981-01-08', '1981-01-09', '1981-01-10'

    ],

    'Temp': [

        20.7, 17.9, 18.8, 14.6, 15.8,
```

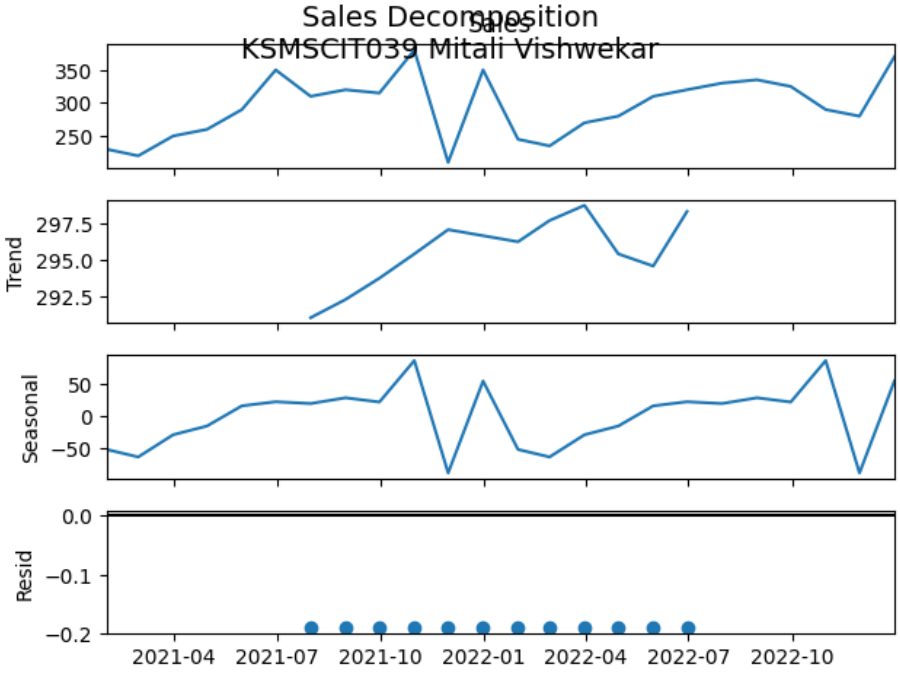
```
        19.2, 22.1, 23.5, 21.0, 18.4
    ]
}
# Create a DataFrame
df = pd.DataFrame(data)
# Data Preprocessing
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
# Display the first few rows of the dataset
print("Mitali KSMSCIT 039")
print(df.head())
print(df.tail())
# Trend Analysis
plt.figure(figsize=(12, 6))
plt.plot(df['Temp'], label='Daily Minimum Temperature', color='blue')
plt.title('Daily Minimum Temperature Over Time\nKSMSCIT039 Mitali', fontsize=14)
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.legend()
plt.show()
# Seasonal Decomposition with period of 5
decomposition = seasonal_decompose(df['Temp'], model='additive', period=5)  # Adjusted
period
```

```
trend = decomposition.trend

seasonal = decomposition.seasonal

residual = decomposition.resid

# Plot the decomposition

plt.figure(figsize=(12, 12))

plt.subplot(4, 1, 1)

plt.plot(df['Temp'], label='Original', color='blue')

plt.legend(loc='upper left')

plt.subplot(4, 1, 2)

plt.plot(trend, label='Trend', color='orange')

plt.legend(loc='upper left')

plt.subplot(4, 1, 3)

plt.plot(seasonal, label='Seasonal', color='green')

plt.legend(loc='upper left')

plt.subplot(4, 1, 4)

plt.plot(residual, label='Residual', color='red')

plt.legend(loc='upper left')

plt.tight_layout()

plt.suptitle('Decomposition of Daily Minimum Temperature\nKSMSCIT039 Mitali',
fontsize=14)

plt.show()

# Forecasting with ARIMA

model = ARIMA(df['Temp'].dropna(), order=(1, 1, 0))  # Adjust parameters as needed
```

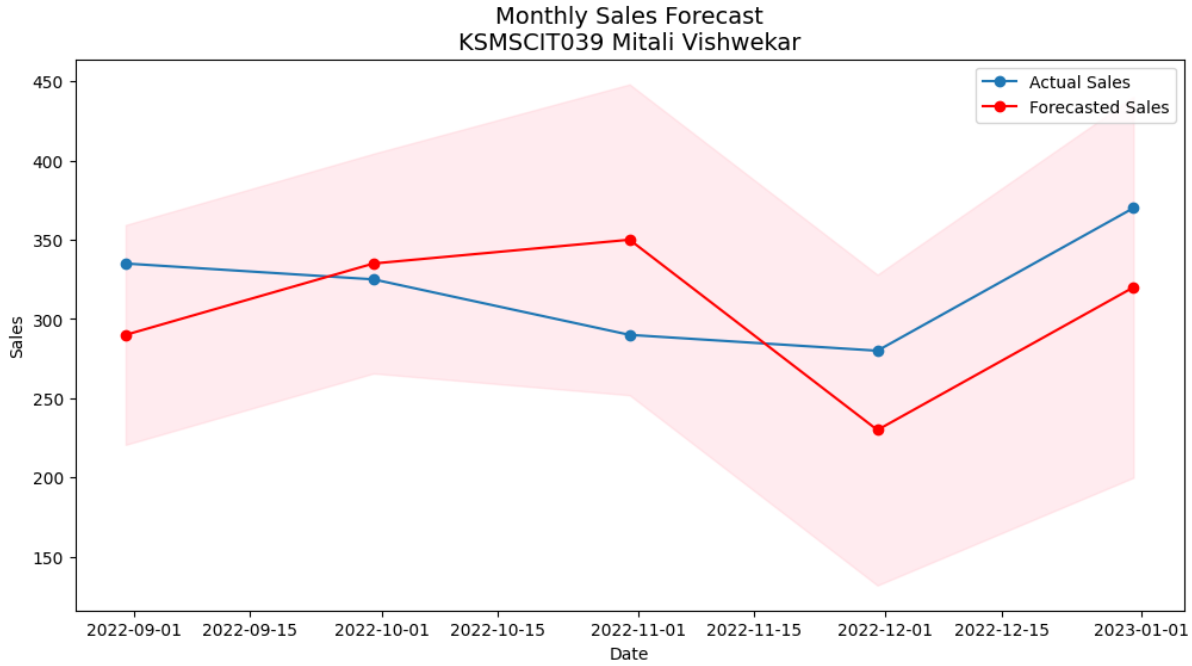
```
model_fit = model.fit()

# Generate forecast for the next 30 days

forecast = model_fit.forecast(steps=30)

# Prepare forecast dates

forecast_dates = pd.date_range(start=df.index[-1] + pd.Timedelta(days=1), periods=30)

# Visualization of forecast results

plt.figure(figsize=(12, 6))

plt.plot(df['Temp'], label='Historical Temperature', color='blue')

plt.plot(forecast_dates, forecast, label='Forecasted Temperature', color='orange')

plt.title('Temperature Forecast\nKSMSCIT039 Mitali', fontsize=14)

plt.xlabel('Date')

plt.ylabel('Temperature (°C)')

plt.legend()

plt.show()
```

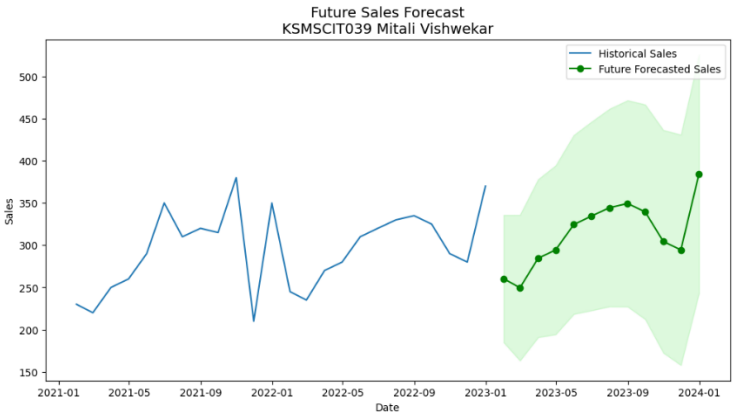
Output: -


Monthly Sales
KSMSCIT039 Mitali Vishwekar
Monthly Sales
Sales
Date
375
350
325
300
275
250
225
2021-01
2021-04
2021-07
2021-10
2022-01
2022-04
2022-07
2022-10
2023-01


Sales Decomposition
KSMSCIT039 Mitali Vishwekar
Sales
Trend
Seasonal
Resid
350
300
250
297.5
295.0
292.5
50
0
−50
0.0
−0.1
−0.2
2021-04
2021-07
2021-10
2022-01
2022-04
2022-07
2022-10

```
Train size: 19 Test size: 5
                                SARIMAX Results
==========================================================================================
Dep. Variable:                             Sales   No. Observations:                   19
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood                0.000
Date:                            Fri, 11 Oct 2024   AIC                           10.000
Time:                                    11:16:43   BIC                              nan
Sample:                                01-31-2021   HQIC                             nan
                                     - 07-31-2022
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.0000         -0       inf      0.000      -1.000      -1.000
ma.L1       1.369e-17         -0      -inf      0.000    1.37e-17    1.37e-17
ar.S.L12            0         -0       nan        nan           0           0
ma.S.L12            0         -0       nan        nan           0           0
sigma2      1250.0000         -0      -inf      0.000    1250.000    1250.000
===================================================================================
Ljung-Box (L1) (Q):                   nan   Jarque-Bera (JB):                 nan
Prob(Q):                              nan   Prob(JB):                         nan
Heteroskedasticity (H):               nan   Skew:                             nan
Prob(H) (two-sided):                  nan   Kurtosis:                         nan
===================================================================================
```





Practical 11

Aim: - Least Square Method for Linear Regression

Theory:

The Least Squares Method is a standard approach in linear regression to find the line that best describes the relationship between a dependent variable y and an independent variable xxx. It works by minimizing the sum of the squared differences between the observed values and the values predicted by the model.

In linear regression, the relationship between x and y is modeled as:

$Y = mx+b$

Where, m is the slope and b is the y-intercept.

Objective: To find the values of mmm and b that minimize the error, making the line of best fit as close as possible to the data points.

Code:-

```
import scipy.stats as stats

import numpy as np

# Sample data (x and y values)

x = np.array([59, 65, 45, 52, 60, 62, 70, 55, 45, 49])

y = np.array([76, 70, 55, 65, 60, 64, 80, 65, 54, 61])

slope_yx, intercept_yx, r_value_yx, p_value_yx, std_err_yx = stats.linregress(y, x)

slope_xy, intercept_xy, r_value_xy, p_value_xy, std_err_xy = stats.linregress(x, y)

estimate_yx = slope_yx * 61 + intercept_yx

# Output results

print("slope_yx:", slope_yx)

print("intercept_yx:", intercept_yx)
```

```
print("slope_xy:", slope_xy)

print("intercept_xy:", intercept_xy)

print("estimate_yx:", estimate_yx)

print("KSMSCIT039 Mitali ")
```

Output:-

```
slope_yx: 0.8217665615141956
intercept_yx: 2.7851735015772903
slope_xy: 0.807001239157373
intercept_xy: 19.646530359355637
estimate_yx: 52.91293375394322
Mitali KSMSCIT 039
```

Interpretation: -

The resulting line minimizes the total squared distance of each data point from the line. This best-fit line provides an estimate for predicting y values based on new x values, making it useful for trend analysis and forecasting in a linear relationship.