**1. Importing Data**

**Steps:**

* Load the dataset into Python using pandas.
* Inspect the data using df.head(), df.info(), and df.describe() to understand the structure, data types, and summary statistics.

**Code Example:**

python

Copy code

import pandas as pd

# Load dataset

df = pd.read\_csv('gym\_members\_data.csv')

# Inspect the data

print(df.head())

print(df.info())

print(df.describe())

**2. Data Cleaning**

**Steps:**

* **Handle Missing Values:**
  + Identify missing or null values using df.isnull().sum().
  + Impute or drop missing data (e.g., mean imputation for numerical columns, mode for categorical ones).
* **Fix Inconsistent or Outlier Values:**
  + Use boxplots or z-score to identify outliers in numerical columns like Calories\_Burned, BMI, and Resting\_BPM.
  + For outliers, either cap values (using quantiles) or remove them if justified.
* **Standardize Categorical Values:**
  + Ensure consistency in categorical values (e.g., Workout\_Type).

**Code Example:**

python

Copy code

# Check for missing values

print(df.isnull().sum())

# Fill missing values for numerical columns

df['Calories\_Burned'].fillna(df['Calories\_Burned'].mean(), inplace=True)

# Detect outliers using IQR

Q1 = df['Calories\_Burned'].quantile(0.25)

Q3 = df['Calories\_Burned'].quantile(0.75)

IQR = Q3 - Q1

df = df[(df['Calories\_Burned'] >= Q1 - 1.5 \* IQR) & (df['Calories\_Burned'] <= Q3 + 1.5 \* IQR)]

**3. Data Exploration and Manipulation**

**Steps:**

* Group data to find patterns:
  + Average Calories\_Burned by Workout\_Type and Experience\_Level.
  + Distribution of Workout\_Frequency across genders or age groups.
* Compute additional metrics, if necessary:
  + For example, derive a new feature like "Calories Burned per Minute".

**Code Example:**

python

Copy code

# Average calories burned by workout type and experience level

grouped\_data = df.groupby(['Workout\_Type', 'Experience\_Level'])['Calories\_Burned'].mean()

print(grouped\_data)

# Add a new column for calories burned per minute

df['Calories\_Burned\_Per\_Min'] = df['Calories\_Burned'] / (df['Session\_Duration'] \* 60)

**4. Data Visualization**

**Steps:**

* Use Matplotlib and Seaborn to create:
  + **Bar charts:** Average calories burned by Workout\_Type.
  + **Scatter plots:** Relationship between Session\_Duration and Calories\_Burned.
  + **Heatmaps:** Correlation between numerical features.

**Code Example:**

python

Copy code

import seaborn as sns

import matplotlib.pyplot as plt

# Bar chart for calories burned by workout type

sns.barplot(x='Workout\_Type', y='Calories\_Burned', data=df)

plt.title('Calories Burned by Workout Type')

plt.show()

# Scatter plot for session duration vs calories burned

sns.scatterplot(x='Session\_Duration', y='Calories\_Burned', hue='Experience\_Level', data=df)

plt.title('Session Duration vs Calories Burned')

plt.show()

# Correlation heatmap

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title('Feature Correlations')

plt.show()

**5. Statistical Analysis**

**Steps:**

* Conduct hypothesis testing to verify significant differences:
  + **Example:** Does Workout\_Type significantly affect Calories\_Burned?
  + Use **ANOVA** for multi-category comparisons or a **t-test** for two groups.
* Analyze correlations between features.

**Code Example:**

python

Copy code

from scipy.stats import f\_oneway, ttest\_ind

# ANOVA test for calories burned across workout types

anova\_result = f\_oneway(

df[df['Workout\_Type'] == 'Cardio']['Calories\_Burned'],

df[df['Workout\_Type'] == 'Strength']['Calories\_Burned'],

df[df['Workout\_Type'] == 'Yoga']['Calories\_Burned']

)

print('ANOVA Result:', anova\_result)

# Correlation test between BMI and resting heart rate

correlation = df['BMI'].corr(df['Resting\_BPM'])

print('Correlation between BMI and Resting BPM:', correlation)

**6. Predictive Modeling**

**Steps:**

* **Objective:** Predict Calories\_Burned using relevant features (Session\_Duration, BMI, Avg\_BPM, etc.).
* **Steps:**
  + Split data into training and testing sets.
  + Train models like **Linear Regression**, **Random Forest**, or **Gradient Boosting**.
  + Evaluate using metrics like **R²**, **RMSE**, and **MAE**.

**Code Example:**

python

Copy code

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

# Select features and target

features = ['Session\_Duration', 'BMI', 'Avg\_BPM', 'Workout\_Frequency', 'Experience\_Level']

target = 'Calories\_Burned'

X = df[features]

y = df[target]

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train Random Forest model

model = RandomForestRegressor()

model.fit(X\_train, y\_train)

# Predictions and evaluation

y\_pred = model.predict(X\_test)

print('R²:', r2\_score(y\_test, y\_pred))

print('RMSE:', mean\_squared\_error(y\_test, y\_pred, squared=False))

**7. Insights and Recommendations**

**Steps:**

* Summarize insights from visualizations and statistical tests:
  + Example: "HIIT burns the most calories per minute, making it an efficient workout type for busy individuals."
* Provide actionable recommendations:
  + "Beginners should focus on longer cardio sessions to optimize calorie burn."

**8. Presentation Preparation**

* Use slides to highlight key findings:
  + Visualizations of trends and insights.
  + Model predictions and their implications.
  + Clear, actionable recommendations for gym-goers and trainers.