NAME: RUCHIR MAKARAND ADNAIK
ROLL: 08
SRN: 202200282
DIV: B
BATCH: B1

ASSIGNMENT 09

CODE:

```
# Step 1: Importing Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Step 2: Data Collection
data = pd.read_csv('gym_membership.csv')  # Load your dataset here
data = data.fillna(0)  # Fill missing values with 0 or an appropriate method

# Step 3: Exploratory Data Analysis (EDA)
print(data.head())  # Show the first few rows of the dataset
print(data.info())  # Get info on the data types and missing values
print(data.describe())  # Get statistical summary of the dataset

# Step 4: Feature Engineering
# Convert categorical variables to numerical if needed
data['personal_training'] = data['personal_training'].astype(int)  # Convert
'personal_training' to numeric (0, 1)
data['attend_group_lesson'] = data['attend_group_lesson'].apply(lambda x: 1 if
x == 'TRUE' else 0)  # Convert to binary
data['avg_time_check_in'] =
pd.to_timedelta(data['avg_time_check_in']).dt.total_seconds() / 60  # Convert
time to minutes
data['avg_time_check_out'] =
pd.to_timedelta(data['avg_time_check_out']).dt.total_seconds() / 60  # Convert
time to minutes
data['avg_time_in_gym'] =
pd.to_timedelta(data['avg_time_in_gym']).dt.total_seconds() / 60  # Convert
time to minutes

# We will drop non-numerical columns that won't be used in prediction
data = data.drop(columns=['id', 'first_name', 'gender', 'abonoment_type',
'fav_group_lesson', 'visit_per_week', 'days_per_week'])
```

```
# Target variable: Let's assume 'personal_training' indicates whether they are
likely to show up
y = data['personal_training']
X = data.drop('personal_training', axis=1)  # All other columns as features

# Step 5: Splitting the dataset into training and testing sets (Train-Test
Split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)  # 70% training, 30% testing

# Step 6: Model Training
model = GaussianNB()  # Create an instance of the Gaussian Naive Bayes model
model.fit(X_train, y_train)  # Fit the model on training data

# After the data processing steps and before the prediction

# Step 7: Model Evaluation
y_pred = model.predict(X_test)  # Make predictions on the test set
accuracy = accuracy_score(y_test, y_pred)  # Calculate accuracy
conf_matrix = confusion_matrix(y_test, y_pred)  # Get confusion matrix
class_report = classification_report(y_test, y_pred)  # Get classification
report

# Print evaluation metrics
print(f'Accuracy: {accuracy:.2f}')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)

# Check the columns of X
print("Columns in feature set (X):", X.columns)
print("Number of columns in feature set (X):", len(X.columns))

# Visualization of the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Did not show up', 'Showed up'],
            yticklabels=['Did not show up', 'Showed up'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Step 8: Single User Input Testing
# Create a function for predicting a single user input
def predict_user_input(input_data):
    # Ensure input data matches the expected format (DataFrame)
```

```
    input_df = pd.DataFrame([input_data], columns=X.columns)
    prediction = model.predict(input_df)  # Predict using the trained model
    return "Showed Up" if prediction[0] == 1 else "Did Not Show Up"

# Example of user input
# Adjust the user input to match the number of features in X
user_input = [30, 130, 160, 75, 1, 0]  # Example input; change as needed
result = predict_user_input(user_input)
print(f'Prediction for user input {user_input}: {result}')
```

OUTPUT:

```
PS C:\Users\ruchi\OneDrive\Desktop\3rd year\DWDM\assignment9> python -u "c:\Users\ruchi\OneDrive\Desktop\3rd year\DWDM\assignment9\assign9.py"
    id first_name  gender   Age  ... avg_time_check_in  avg_time_check_out  avg_time_in_gym personal_training
0  1.0        Fey  Female  27.0  ...          19:31:00            21:42:00           131.0             False
1  2.0    Doralin  Female  47.0  ...          19:31:00            21:43:00           132.0              True
2  3.0       Linc    Male  41.0  ...          08:29:00            09:55:00            86.0              True
3  4.0     Darren    Male  44.0  ...          09:54:00            12:15:00           141.0              True
4  5.0       Petr    Male  44.0  ...          08:29:00            10:39:00           130.0              True

[5 rows x 14 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  1000 non-null   float64
 1   first_name          1000 non-null   object
 2   gender              1000 non-null   object
 3   Age                 1000 non-null   float64
 4   abonoment_type      1000 non-null   object
 5   Unnamed: 5          1000 non-null   float64
 6   visit_per_week      1000 non-null   float64
 7   days_per_week       1000 non-null   object
 8   attend_group_lesson 1000 non-null   object
 9   fav_group_lesson    1000 non-null   object
 10  avg_time_check_in   1000 non-null   object
 11  avg_time_check_out  1000 non-null   object
 12  avg_time_in_gym     1000 non-null   float64
 13  personal_training   1000 non-null   object
dtypes: float64(5), object(9)
memory usage: 109.5+ KB
None
              id          Age  Unnamed: 5  visit_per_week  avg_time_in_gym
count  1000.000000  1000.000000      1000.0     1000.000000      1000.000000
mean      1.275000     1.591000         0.0        0.126000         4.902000
std       6.429675     7.320472         0.0        0.613596        23.072033
min       0.000000     0.000000         0.0        0.000000         0.000000
25%       0.000000     0.000000         0.0        0.000000         0.000000
50%       0.000000     0.000000         0.0        0.000000         0.000000
75%       0.000000     0.000000         0.0        0.000000         0.000000
max      50.000000    48.000000         0.0        5.000000       175.000000
Accuracy: 0.98
[[285   6]
 [  0   9]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       291
           1       0.60      1.00      0.75         9

    accuracy                           0.98       300
   macro avg       0.80      0.99      0.87       300
weighted avg       0.99      0.98      0.98       300

Columns in feature set (X): Index(['Age', 'Unnamed: 5', 'attend_group_lesson', 'avg_time_check_in',
       'avg_time_check_out', 'avg_time_in_gym'],
      dtype='object')
Number of columns in feature set (X): 6
Prediction for user input [30, 130, 160, 75, 1, 0]: Showed Up
PS C:\Users\ruchi\OneDrive\Desktop\3rd year\DWDM\assignment9>
```