~\Downloads\final (1).py

```
"""
Personalised Menstrual Tracking and Prediction Using Machine Learning
This project uses data science techniques to analyze and predict menstrual cycle patterns. It follows the CRISP-DM methodology and applies Exploratory Data Analysis (EDA), statistical analysis, and machine learning (Linear Regression) to forecast the next cycle start date.
"""

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

df = pd.read_excel('/content/drive/MyDrive/google colab files/Menstrual_cycle_tracking.xlsx', na_values=[' ', '', 'NA', 'NaN'])
df

"""Data Cleaning
 Replace empty strings with NaN
 Check missing values in each column
"""

df = df.replace(r'^\s*$', pd.NA, regex=True)
missing_values = df.isnull().sum()
print(missing_values)
df.shape

df.info()

df.describe()

df.duplicated().sum() #no duplicates

columns_to_fill = ['MeanCycleLength', 'MeanMensesLength', 'MeanBleedingIntensity','Height','Weight','Age','BMI']  # add more if needed

df[columns_to_fill] = df.groupby('ClientID')[columns_to_fill].ffill()
df

df['Cycle_Length'] = pd.to_numeric(df['Cycle_Length'], errors='coerce')

df.to_csv("/content/drive/My Drive/google colab files/cleaned_menstrual_data.csv", index=False) #saved cleaned dataset

df.shape
```

```python

df.columns

df['ClientID'].nunique() # Unique Value Counts
df['PMS_intensity'].value_counts()

# Summary Statistics
print(df['Cycle_Length'].mean())
print(df['Cycle_Length'].median())
print(df['Cycle_Length'].std())
print(df['Cycle_Length'].min())
print(df['Cycle_Length'].max())
print(df['Cycle_Length'].mode())
Q1 = df['Cycle_Length'].quantile(0.25)
Q3 = df['Cycle_Length'].quantile(0.75)
IQR = Q3 - Q1
outliers = df[(df['Cycle_Length'] < Q1 - 1.5*IQR) | (df['Cycle_Length'] > Q3 + 1.5*IQR)]
print(f"Number of outliers: {len(outliers)}")

#**correlation and covarience**


df.corr(numeric_only=True)
df.cov(numeric_only=True)

df = pd.read_csv("/content/drive/My Drive/google colab files/cleaned_menstrual_data.csv")
df['Start_Date'] = pd.to_datetime(df['Start_Date'], errors='coerce')
df['start_month'] = df['Start_Date'].dt.month
df['start_weekday'] = df['Start_Date'].dt.dayofweek
df.to_csv("/content/drive/My Drive/google colab files/updated_menstrual_cycle_data.csv", index=False)
df

df['start_month'].value_counts().sort_index()
df['start_weekday'].value_counts()

"""Exploratory Data Analysis (EDA)
We explore key features through visualizations and analyze correlations among them.
"""

sns.set(style="whitegrid", palette="pastel") # style settings for better visuals

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

min_val = int(df['Cycle_Length'].min())
max_val = int(df['Cycle_Length'].max())
bins = np.arange(min_val - 0.5, max_val + 1.5, 1)

sns.histplot(df['Cycle_Length'], bins=bins, kde=True, color='lightcoral', edgecolor='black', linewidth=1.2)
```

```python

mean_val = df['Cycle_Length'].mean()
median_val = df['Cycle_Length'].median()
plt.axvline(mean_val, color='blue', linestyle='--', label=f'Mean: {mean_val:.1f}')
plt.axvline(median_val, color='green', linestyle='-.', label=f'Median: {median_val:.1f}')

plt.title("Distribution of Menstrual Cycle Lengths", fontsize=14)
plt.xlabel("Cycle Length (days)", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.xticks(np.arange(min_val, max_val + 1, 1))
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

"""Top 10 Clients with Longest Average Cycle Length"""

avg_cycle = df.groupby('ClientID')['Cycle_Length'].mean()
avg_cycle_df = avg_cycle.reset_index()
avg_cycle_df.columns = ['ClientID', 'AverageCycleLength']

top10 = avg_cycle_df.sort_values(by='AverageCycleLength', ascending=False).head(10)

plt.figure(figsize=(10, 5))
sns.barplot(data=top10, x='ClientID', y='AverageCycleLength',hue="ClientID",legend=False, palette='Blues_r')
plt.title("Top 10 Clients with Longest Average Cycle Length")
plt.xlabel("Client ID")
plt.ylabel("Average Cycle Length (days)")
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()

client_id = 'nfp8122'
user_data = df[df['ClientID'] == client_id].sort_values('Start_Date')
plt.figure(figsize=(12, 5))
plt.plot(user_data['Start_Date'], user_data['Cycle_Length'], marker='o', linestyle='-', color='purple') # Changed 'start_date' to 'StartOfPeriodDate'
plt.title(f"Cycle Length Changes Over Time for Client {client_id}")
plt.xlabel("Start Date")
plt.ylabel("Cycle Length (days)")
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

"""The top 15 users with the lowest standard deviation in cycle length were identified to evaluate consistency. These users exhibit minimal fluctuations in their menstrual cycle durations, indicating **high regularity**. Such patterns are useful for training initial predictive models, as they represent relatively stable biological cycles."""

```

```python
client_cycle_std = df.groupby('ClientID')['Cycle_Length'].std().dropna().sort_values()
top_clients = client_cycle_std.head(15)
colors = sns.color_palette("hls", len(top_clients))

plt.figure(figsize=(10, 5))
top_clients.plot(kind='bar', color=colors)
plt.title("Top 15 Most Consistent Users (Lowest Std Dev of Cycle Length)")
plt.xlabel("Client ID")
plt.ylabel("Cycle Length Std Dev")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

"""BOX PLOT
This plot shows how the menstrual cycle length changes for the top 10 most active users in your dataset.
Each vertical box is for one user (Client ID).
The line inside each box is the middle value (median) of that user's cycle lengths.
The box shows where most of their cycles fall (the middle 50%).
The lines outside the box show the full range (except for unusual values).
The dots outside the lines are outliers — cycles that were much shorter or longer than usual.
The plot shows that some users have very regular cycle lengths, while others have a lot of variation. A few users had periods that were much shorter or longer than normal.
"""

top_users = df['ClientID'].value_counts().head(10).index # top 10 users with the most cycle records
top_df = df[df['ClientID'].isin(top_users)]

plt.figure(figsize=(12, 6))
sns.boxplot(x='ClientID', y='Cycle_Length', data=top_df, hue='ClientID', palette='Set2', legend=False)
plt.title("Cycle Length Variation Across Top 10 Users")
plt.xlabel("Client ID")
plt.ylabel("Cycle Length (days)")
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

"""
 Box Plot: Most Irregular Users (Highest Std Dev)
This plot shows the top 10 most irregular users based on cycle length variability.
Most of them show wide variation, and many have outliers, indicating inconsistent cycles.
 This suggests that a one-size-fits-all prediction model may not be reliable and personalized predictions could be more effective.
"""

client_cycle_std = df.groupby('ClientID')['Cycle_Length'].std().dropna()

irregular_users = client_cycle_std.sort_values(ascending=False).head(10).index
```

```
irregular_df = df[df['ClientID'].isin(irregular_users)]

plt.figure(figsize=(12, 6)) #box plot
sns.boxplot(x='ClientID', y='Cycle_Length', data=irregular_df, hue='ClientID', palette='Set3', legend=False)
plt.title("Cycle Length Variation Across Top 10 Most Irregular Users")
plt.xlabel("Client ID")
plt.ylabel("Cycle Length (days)")
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

"""HEATMAP
Does stress level increase or decrease cycle length?
Is PMS_intensity linked with bleeding intensity?
Are any features strongly negatively or positively correlated?
"""

selected_cols = ['LengthofCycle', 'LengthofMenses', 'stress_level', 'PMS_intensity','MeanBleedingIntensity', 'TotalNumberofHighDays', 'Age']
filtered_df = df[selected_cols].dropna()
corr = filtered_df.corr()

plt.figure(figsize=(8, 6)) #heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', square=True, linewidths=0.5)
plt.title("Correlation Heatmap: Key Menstrual Features")
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()

"""Hypothesis **Test**
**✅ T-Test:** Does PMS Intensity Affect Cycle Length?
Test whether there's a significant difference in the average cycle length between:
Users with low PMS intensity (e.g., PMS ≤ 5)
Users with high PMS intensity (e.g., PMS > 5)
"""

from scipy.stats import ttest_ind
low_pms = df[df['PMS_intensity'] <= 5]['LengthofCycle'].dropna()
high_pms = df[df['PMS_intensity'] > 5]['LengthofCycle'].dropna()
print("Mean cycle length (Low PMS):", low_pms.mean())
print("Mean cycle length (High PMS):", high_pms.mean())
t_stat, p_value = ttest_ind(low_pms, high_pms, equal_var=False) #ttest
print("T-statistic:", t_stat)
print("P-value:", p_value)
if p_value < 0.05:
    print("✅ Result: Significant difference in cycle length between low and high PMS groups.")
else:
    print("❌ Result: No significant difference in cycle length between the two groups.")
```

```python

"""data suggests that PMS intensity doesn't significantly affect the length of the menstrual cycle
✅ **ANOVA Test**
To determine whether menstrual cycle length significantly varies across age groups.
Test Performed:
One-Way ANOVA
Groups: 20–29, 30–39, 40–49
Variable analyzed: LengthofCycle
"""

bins = [20, 30, 40, 50]
labels = ['20-29', '30-39', '40-49']
df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
group_counts = df.groupby('AgeGroup', observed=False)['LengthofCycle'].count()
valid_groups = group_counts[group_counts >= 2].index
df_valid = df[df['AgeGroup'].isin(valid_groups)]
print("Groups included in ANOVA:", df_valid['AgeGroup'].unique())
anova_groups = [group['LengthofCycle'].dropna() for name, group in df_valid.groupby('AgeGroup', observed=False) if len(group) > 1]
f_stat, p_value = f_oneway(*anova_groups)  # ANOVA

print("F-statistic:", f_stat)  # result
print("P-value:", p_value)
if p_value < 0.05:
    print("✅ Significant difference in cycle length across age groups.")
else:
    print("❌ No significant difference in cycle length across age groups.")

"""📈 Results:
F-statistic: 4.10
P-value: 0.0167
Cycle length is significantly affected by age. Users in different age groups may experience different menstrual patterns.
✅ **Shapiro-Wilk Test:**
To see if the LengthofCycle column is normally distributed, which is helpful before applying regression or other parametric models.
"""

from scipy.stats import shapiro
cycle_lengths = df['LengthofCycle'].dropna()
stat, p_value = shapiro(cycle_lengths)
print("Shapiro-Wilk Test Statistic:", stat)
print("P-value:", p_value)

# Interpretation
if p_value < 0.05:
    print("❌ Data is NOT normally distributed (reject H0).")
else:
    print("✅ Data IS normally distributed (fail to reject H0).")

```

```
"""MODEL
We implemented and compared two regression models — Linear Regression (aligned with syllabus) and Random Forest (real-world extension) — to predict LengthofCycle using lifestyle and health features. After preprocessing and splitting the data, both models were trained and evaluated using R² score, MAE, and RMSE. This step aligns with the CRISP-DM framework's modeling and evaluation phases.
"""

from sklearn.model_selection import GroupShuffleSplit
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

df = pd.read_csv("/content/drive/MyDrive/google colab files/updated_menstrual_cycle_data.csv")
df['Start_Date'] = pd.to_datetime(df['Start_Date'], errors='coerce')
df = df.sort_values(by=['ClientID', 'Start_Date'])
df['Next_Start_Date'] = df.groupby('ClientID')['Start_Date'].shift(-1)
df['Target_DaysToNextCycle'] = (df['Next_Start_Date'] - df['Start_Date']).dt.days
df = df.dropna(subset=['Target_DaysToNextCycle'])
mapping_dicts = {
    'exercise': {'Cardio': 0, 'Strength': 1, 'Yoga': 2},
    'diet_quality': {'Poor': 0, 'Average': 1, 'Good': 2},
    'spotting': {'No': 0, 'Yes': 1},
    'sexual_activity': {'No': 0, 'Yes': 1},
    'sleep_quality': {'Poor': 0, 'Average': 1, 'Good': 2}
}
for col, mapping in mapping_dicts.items():
    df[col] = df[col].map(mapping).fillna(-1).astype(int)
features = [
    'LengthofCycle', 'LengthofMenses',
    'MeanCycleLength', 'MeanMensesLength', 'BMI',
    'PMS_intensity', 'EstimatedDayofOvulation',
    'MeanBleedingIntensity', 'TotalDaysofFertility',
    'TotalMensesScore', 'Age', 'sleep_quality',
    'stress_level', 'exercise', 'diet_quality',
    'spotting', 'sexual_activity'
]
df = df.dropna(subset=features + ['Target_DaysToNextCycle'])
X = df[features]
y = df['Target_DaysToNextCycle']
groups = df['ClientID']

splitter = GroupShuffleSplit(test_size=0.2, n_splits=1, random_state=42)
train_idx, test_idx = next(splitter.split(X, y, groups=groups))

X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

# ✅ Train model
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
```

```python
y_pred_rf = rf_model.predict(X_test)

# ✅ Evaluate model
mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print("📊 Random Forest Evaluation:")
print(f"Mean Squared Error (MSE): {mse_rf:.2f}")
print(f"Mean Absolute Error (MAE): {mae_rf:.2f}")
print(f"R² Score: {r2_rf:.2f}")

"""# ✅ Visualize
Actual vs Predicted Days (Random Forest
"""

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred_rf, color='darkorange', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r', label='Perfect Prediction Line')
plt.xlabel("Actual Days to Next Cycle")
plt.ylabel("Predicted Days")
plt.title("Actual vs Predicted Days (Random Forest)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

"""✅ Predict next cycle for latest entry"""

target_client = 'nfp8237'
client_data = df[df['ClientID'] == target_client].sort_values('Start_Date')
latest_entry = client_data.iloc[[-1]]
last_start_date = latest_entry['Start_Date'].values[0]
latest_features = latest_entry[features]

predicted_days = rf_model.predict(latest_features)[0]


predicted_next_date = pd.to_datetime(last_start_date) + pd.Timedelta(days=int(predicted_days))

# 📢 Output
print("🧬 Prediction for Client:", target_client)
print("📅 Last Known Cycle Start Date:", pd.to_datetime(last_start_date).date())
print(f"🔮 Predicted Days to Next Cycle: {int(predicted_days)} days")
print("📅 Predicted Next Cycle Start Date:", predicted_next_date.date())

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

```

```python

# y_test: actual values, y_pred_rf: predicted values
mae = mean_absolute_error(y_test, y_pred_rf)
mse = mean_squared_error(y_test, y_pred_rf)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred_rf)

# Custom accuracy: within ±2 days
tolerance = 2
within_tolerance = np.abs(y_test - y_pred_rf) <= tolerance
custom_accuracy = within_tolerance.mean() * 100

print("📊 Evaluation Metrics:")
print(f"MAE(Mean Absolute Error): {mae:.2f} days")
print(f"MSE(Mean Squared Error): {mse:.2f}")
print(f"RMSE(Root Mean Squared Error): {rmse:.2f} days")
print(f"R² Score: {r2:.2f}")
print(f"Custom Accuracy (±{tolerance} days): {custom_accuracy:.2f}%")
```