NASA CONCEPTUAL

Team Members

- Adam
- Elizabeth
- Khoi
- Manuel

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel as C
from sklearn.metrics import mean_squared_error
```

Load dataset

In [ ]:
```
# Load the dataset
data = pd.read_csv('your_weather_data.csv')

# Display the first few rows
print(data.head())
```

Exploratory Data Analysis

In [ ]:
```
# Check for missing values
print(data.isnull().sum())

# Basic statistics
print(data.describe())

# Example visualization
data['Temperature'].plot(kind='line', title='Temperature Trends', figsize=(10, 5))
plt.show()
```

Data Processing

In [ ]:
```
# Fill missing values
data.fillna(method='ffill', inplace=True)

# Select features and target
X = data[['Feature1', 'Feature2', 'Feature3']]  # Replace with relevant columns
y = data['Target']  # Replace with your target variable (e.g., 'Temperature')

# Normalize features
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random
```

Define and Train the GPR Model

In [ ]:
```
# Define the kernel
kernel = C(1.0, (1e-3, 1e3)) * RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e2))

# Initialize the Gaussian Process Regressor
gpr = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=10, random_state=42

# Train the model
gpr.fit(X_train, y_train)
```

Make predictions and evaluate model

In [ ]:
```
# Predict on the test set
y_pred, sigma = gpr.predict(X_test, return_std=True)

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'Root Mean Squared Error: {rmse}')

# Plot predictions vs actual values
plt.figure(figsize=(10, 5))
plt.plot(y_test.values, label='Actual')
plt.plot(y_pred, label='Predicted')
plt.fill_between(np.arange(len(y_pred)), y_pred - sigma, y_pred + sigma, alpha=0.2, la
plt.legend()
plt.title('GPR Weather Prediction')
plt.show()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: