Calorie Burned Prediction Web App

Overview

The **Calorie Burned Prediction Web App** is an interactive tool designed to estimate the number of kilocalories burned based on user-specific parameters such as age, BMI, exercise duration, heart rate, and body temperature. This app leverages machine learning models to provide personalized calorie burn predictions and offers insights into how a user's parameters compare with a broader dataset.

Features

1. **User Input Parameters:**
   - Users can input their parameters via sliders and radio buttons, including:
     - Age
     - BMI (Body Mass Index)
     - Duration of Exercise (in minutes)
     - Heart Rate (beats per minute)
     - Body Temperature (in Celsius)
     - Gender (Male/Female)
2. **Prediction:**
   - The app uses a pre-trained Random Forest Regressor model to predict the number of kilocalories burned based on the provided inputs.
3. **Visualization:**
   - Displays the predicted calorie burn.
   - Shows a sample of similar results from a dataset, providing context to the prediction.
   - Offers general comparisons to indicate how the user's parameters stack up against others.
4. **Data Insights:**
   - Compares user parameters with those of others in the dataset, including:
     - Age
     - Exercise duration
     - Heart rate
     - Body temperature

Components

1. Importing Libraries

```python
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
import time

from matplotlib import style
style.use("seaborn")
import plotly.express as px

import warnings
warnings.filterwarnings('ignore')
```

- **Streamlit**: For building the interactive web interface.
- **NumPy**: For numerical operations.
- **Matplotlib & Seaborn**: For data visualization.
- **Pandas**: For data manipulation.
- **Scikit-Learn**: For machine learning models and metrics.
- **Plotly**: For interactive plots.

2. Sidebar for User Input

```
st.sidebar.header("User Input Parameters : ")

def user_input_features():
    global age, bmi, duration, heart_rate, body_temp
    age = st.sidebar.slider("Age : " , 10 , 100 , 30)
    bmi = st.sidebar.slider("BMI : " , 15 , 40 , 20)
    duration = st.sidebar.slider("Duration (min) : " , 0 , 35 , 15)
    heart_rate = st.sidebar.slider("Heart Rate : " , 60 , 130 , 80)
    body_temp = st.sidebar.slider("Body Temperature (C) : " , 36 ,
42 , 38)
    gender_button = st.sidebar.radio("Gender : ", ("Male" , "Female"))

    if gender_button == "Male":
        gender = 1
    else:
        gender = 0

    data = {
        "Age": age,
        "BMI": bmi,
        "Duration": duration,
        "Heart_Rate": heart_rate,
        "Body_Temp": body_temp,
        "Gender": ["1" if gender_button == "Male" else "0"]
    }
    data_model = {
        "Age": age,
        "BMI": bmi,
        "Duration": duration,
        "Heart_Rate": heart_rate,
        "Body_Temp": body_temp,
        "Gender_male": gender
    }

    features = pd.DataFrame(data_model, index=[0])
    data = pd.DataFrame(data, index=[0])
    return features, data
```

- Provides sliders and radio buttons for users to input their parameters.
- Converts input values into a format suitable for the machine learning model.

3. Displaying the App Header and Image

```
st.write("## Calories Burned Prediction")
from PIL import Image

# Opening and displaying the image
import os
image = Image.open('calories-burned-prediction-main/gym_04.jpg')
image = image.resize((500, 300))
st.image(image, caption='Workout')
st.sidebar.markdown('''
    <a href="https://www.freepik.com/free-vector/young-man-exercising-fitness-gym-room-with-sport-equipment-workouts-guy-training-lifting-dumbbell-sitting-bench_24023173.htm#query=gym%20cartoon&position=0&from_view=keyword&track=ais">Image by studio4rt
    </a> on Freepik''',
    unsafe_allow_html=True
)
```

- Displays the title of the app and a relevant image.

4. Data Preparation and Model Loading

```
df, data = user_input_features()
st.write("---")
st.header("Your Parameters : ")
latest_iteration = st.empty()
bar = st.progress(0)
for i in range(100):
    bar.progress(i + 1)
    time.sleep(0.01)
st.write(data)

calories = pd.read_csv("calories-burned-prediction-main/calories.csv")
exercise = pd.read_csv("calories-burned-prediction-main/exercise.csv")
```

```
exercise_df = exercise.merge(calories, on="User_ID")
exercise_df.drop(columns="User_ID", inplace=True)

import joblib
random_reg =
joblib.load("calories-burned-prediction-main/random_forest.joblib")
prediction = random_reg.predict(df)
```

- Loads and merges datasets for exercise and calorie data.
- Loads the pre-trained Random Forest model to make predictions.

5. Displaying Prediction and Similar Results

```
st.write("---")
st.header("Prediction : ")
latest_iteration = st.empty()
bar = st.progress(0)
for i in range(100):
    bar.progress(i + 1)
    time.sleep(0.01)

st.write(round(prediction[0], 2), "   **kilocalories**")

st.write("---")
st.header("Similar Results : ")
latest_iteration = st.empty()
bar = st.progress(0)
for i in range(100):
    bar.progress(i + 1)
    time.sleep(0.01)

range = [prediction[0] - 10, prediction[0] + 10]
ds = exercise_df[(exercise_df["Calories"] >= range[0]) &
(exercise_df["Calories"] <= range[-1])]
st.write(ds.sample(5))
```

- Displays the predicted calories burned.
- Shows a sample of similar results from the dataset within a range of the predicted value.

6. Providing General Information

```
st.write("---")
st.header("General Information : ")

boolean_age = (exercise_df["Age"] < age).tolist()
boolean_duration = (exercise_df["Duration"] < duration).tolist()
boolean_body_temp = (exercise_df["Body_Temp"] < body_temp).tolist()
boolean_heart_rate = (exercise_df["Heart_Rate"] < heart_rate).tolist()

st.write("You are older than %" , round(sum(boolean_age) /
len(boolean_age) , 2) * 100 , "of other people.")
st.write("Your had higher exercise duration than %" ,
round(sum(boolean_duration) / len(boolean_duration) , 2) * 100 , "of
other people.")
st.write("You had more heart rate than %" ,
round(sum(boolean_heart_rate) / len(boolean_heart_rate) , 2) * 100 ,
"of other people during exercise.")
st.write("You had higher body temperature than %" ,
round(sum(boolean_body_temp) / len(boolean_body_temp) , 2) * 100 , "of
other people during exercise.")
```

- Compares user parameters with the dataset to provide contextual insights.

Conclusion

The **Calorie Burned Prediction Web App** provides users with an easy-to-use interface for estimating calories burned based on various personal metrics. By leveraging machine learning models and interactive visualizations, users can gain insights into their exercise data and compare their metrics with a broader population. ```

This description provides a clear and comprehensive overview of the app’s purpose, features, components, and functionality. It helps users understand the app’s capabilities and how to interact with it.