Habit Pattern Predictor — Full Project

This document contains the full project files. Copy each file into your local project folder preserving the structure.

```
habit-pattern-predictor/
|-- src/
|   |-- main.py
|   |-- model.py
|   |-- utils.py
|
|-- data/
|   |-- sample_data.csv
|
|-- requirements.txt
|-- README.md
```

---

src/utils.py

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


def generate_synthetic_data(days=365, seed=42):
    np.random.seed(seed)
    dates = pd.date_range(end=pd.Timestamp.today(), periods=days)
    # base signals
    sleep = np.clip(np.random.normal(7, 1.0, days), 4, 10)  # hours
    screen = np.clip(np.random.normal(180, 60, days), 30, 600)  # minutes
    steps = np.clip(np.random.normal(7000, 2500, days), 0, 25000)
    mood = np.clip(np.random.normal(3.5, 1.0, days), 1, 5)  # 1-5

    # productivity is correlated positively with sleep, steps, mood and
negatively with screen
    prod = (
        0.4 * (sleep - 4) / 6 +
        0.3 * (steps / 15000) +
        0.2 * ((mood - 1) / 4) -
        0.2 * (screen / 600)
    )
    # scale to 0-100
    prod = (prod - prod.min()) / (prod.max() - prod.min()) * 100
    # add noise
    prod = np.clip(prod + np.random.normal(0, 6, days), 0, 100)
```

```
    df = pd.DataFrame({
        'date': dates,
        'sleep_hours': np.round(sleep, 2),
        'screen_minutes': np.round(screen, 0).astype(int),
        'steps': np.round(steps, 0).astype(int),
        'mood': np.round(mood, 2),
        'productivity': np.round(prod, 2)
    })
    df = df.sort_values('date').reset_index(drop=True)
    return df


def load_sample_data(path='data/sample_data.csv'):
    try:
        return pd.read_csv(path, parse_dates=['date'])
    except Exception:
        df = generate_synthetic_data(365)
        df.to_csv(path, index=False)
        return df


def create_features(df):
    df = df.copy()
    df['day_of_week'] = df['date'].dt.dayofweek
    # rolling features
    df['sleep_3d_avg'] = df['sleep_hours'].rolling(3, min_periods=1).mean()
    df['screen_3d_avg'] = df['screen_minutes'].rolling(3,
min_periods=1).mean()
    df['steps_3d_avg'] = df['steps'].rolling(3, min_periods=1).mean()
    df['mood_3d_avg'] = df['mood'].rolling(3, min_periods=1).mean()
    return df


def split_and_scale(df, target='productivity'):
    feature_cols = [
        'sleep_hours', 'screen_minutes', 'steps', 'mood',
        'day_of_week', 'sleep_3d_avg', 'screen_3d_avg', 'steps_3d_avg',
'mood_3d_avg'
    ]
    df = create_features(df)
    X = df[feature_cols].fillna(method='bfill').values
    y = df[target].values

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)
    scaler = StandardScaler()
    X_train_s = scaler.fit_transform(X_train)
    X_test_s = scaler.transform(X_test)
    return X_train_s, X_test_s, y_train, y_test, scaler, feature_cols
```

src/model.py

```
import joblib
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


class HabitModel:
    def __init__(self):
        self.model = LinearRegression()
        self.scaler = None
        self.features = None

    def train(self, X_train, y_train):
        self.model.fit(X_train, y_train)
        return self

    def evaluate(self, X, y):
        preds = self.model.predict(X)
        rmse = mean_squared_error(y, preds, squared=False)
        r2 = r2_score(y, preds)
        return {'rmse': rmse, 'r2': r2}

    def predict(self, X):
        return self.model.predict(X)

    def save(self, path='model.joblib'):
        joblib.dump(self, path)

    @classmethod
    def load(cls, path='model.joblib'):
        return joblib.load(path)
```

src/main.py

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from src.utils import load_sample_data, split_and_scale, create_features
from src.model import HabitModel
from sklearn.linear_model import LinearRegression

st.set_page_config(page_title='Habit Predictor — Gamified',
layout='centered')
```

```
st.title('✨ Habit Pattern Predictor — Gamified')
st.write('Predict your next-day productivity score and earn badges for
healthy habits!')

# Load data
@st.cache_data
def load_data():
    return load_sample_data()

df = load_data()

# Sidebar inputs
st.sidebar.header('Input (or upload your CSV)')
uploaded = st.sidebar.file_uploader('Upload CSV with columns:
date,sleep_hours,screen_minutes,steps,mood,productivity (optional)',
type=['csv'])
if uploaded is not None:
    user_df = pd.read_csv(uploaded, parse_dates=['date'])
else:
    user_df = df.copy()

st.sidebar.markdown('---')
train_button = st.sidebar.button('Train / Retrain Model')

# Show sample of data
st.subheader('Sample Habit Data')
st.dataframe(user_df.tail(7))

# Build features and split
X_train, X_test, y_train, y_test, scaler, feature_cols =
split_and_scale(user_df)

# Train model
model = HabitModel()
model.train(X_train, y_train)
eval_train = model.evaluate(X_train, y_train)
eval_test = model.evaluate(X_test, y_test)

st.subheader('Model Performance')
st.write(f"Train RMSE: {eval_train['rmse']:.2f}, R²: {eval_train['r2']:.2f}")
st.write(f"Test RMSE: {eval_test['rmse']:.2f}, R²: {eval_test['r2']:.2f}")

# Predict next day using latest row
st.subheader('Predict Next Day Productivity')
latest = create_features(user_df).iloc[-1]

next_input = np.array([
    latest['sleep_hours'], latest['screen_minutes'], latest['steps'],
latest['mood'],
    latest['day_of_week'], latest['sleep_3d_avg'], latest['screen_3d_avg'],
latest['steps_3d_avg'], latest['mood_3d_avg']
```

```
]).reshape(1, -1)
next_input_s = scaler.transform(next_input)
next_pred = model.predict(next_input_s)[0]

st.metric('Predicted Productivity (0-100)', f"{next_pred:.1f}")

# Gamified badges
st.subheader('Badges & Tips')
badges = []
if latest['sleep_hours'] >= 7:
    badges.append('🌙 Well-Rested')
if latest['steps'] >= 8000:
    badges.append('🐶 Step Champion')
if latest['screen_minutes'] <= 120:
    badges.append('🧯 Screen Minimalist')
if latest['mood'] >= 4.0:
    badges.append('🦃 Mood Booster')

if not badges:
    st.info('No badges earned yet — small consistent changes help!')
else:
    cols = st.columns(len(badges))
    for c, b in zip(cols, badges):
        c.markdown(f"### {b}")

# Show trend chart
st.subheader('7-Day Productivity Trend')
fig, ax = plt.subplots()
recent = user_df.tail(30)
ax.plot(recent['date'], recent['productivity'], marker='o')
ax.set_xlabel('Date')
ax.set_ylabel('Productivity')
ax.grid(True)
st.pyplot(fig)

# Personalized tip
st.subheader('Personalized Tip')
# simple rule-based tip
if latest['screen_minutes'] > 240:
    st.warning('Your screen time is high — try reducing by 30-60 minutes
before bed.')
elif latest['sleep_hours'] < 6:
    st.warning('Consider increasing sleep to 7-8 hours for better
productivity.')
else:
    st.success('Keep up the good routine — small improvements add up!')

# Export model (optional)
if st.button('Export Model'):
    model.save('model.joblib')
    st.download_button('Download model.joblib', data=open('model.joblib',
```

```
'rb'), file_name='model.joblib')

st.markdown('---')
st.caption('Made with 🦬 — edit src/utils.py to plug your real data')
```

---

data/sample_data.csv (first 10 rows)

```
date,sleep_hours,screen_minutes,steps,mood,productivity
2024-10-27,6.75,198,8300,3.45,61.23
2024-10-28,7.02,150,7200,3.90,64.11
2024-10-29,6.50,220,6000,3.10,55.07
2024-10-30,7.80,120,10000,4.50,78.34
2024-10-31,8.00,90,12000,4.80,85.12
2024-11-01,6.20,300,4000,2.90,48.22
2024-11-02,7.10,160,9000,3.80,70.01
2024-11-03,5.90,400,2000,2.50,30.44
2024-11-04,7.50,110,11000,4.20,80.89
2024-11-05,6.80,180,7500,3.60,62.56
```

---

requirements.txt

```
streamlit
pandas
numpy
scikit-learn
matplotlib
joblib
```

README.md

````
# Habit Pattern Predictor (Gamified)

Predict daily productivity from habit data. Includes synthetic data
generator, Linear Regression model, and a gamified Streamlit interface.

## Run locally
1. Create virtualenv
```bash
python -m venv venv
source venv/bin/activate  # or venv\\Scripts\\activate on Windows
pip install -r requirements.txt
````
````

2. Run Streamlit

```
streamlit run src/main.py
```

Resume bullet

- Built a gamified ML app to predict daily productivity from habit data using Linear Regression and a Streamlit dashboard; included badge-based user feedback and trend visualizations. ```

Next steps

- Copy the files into your local workspace
- Run `pip install -r requirements.txt` and `streamlit run src/main.py`
- When ready I can generate a polished GitHub `README.md` (if you want the one in the repo to be expanded) and a short resume bullet tailored to a specific job description.