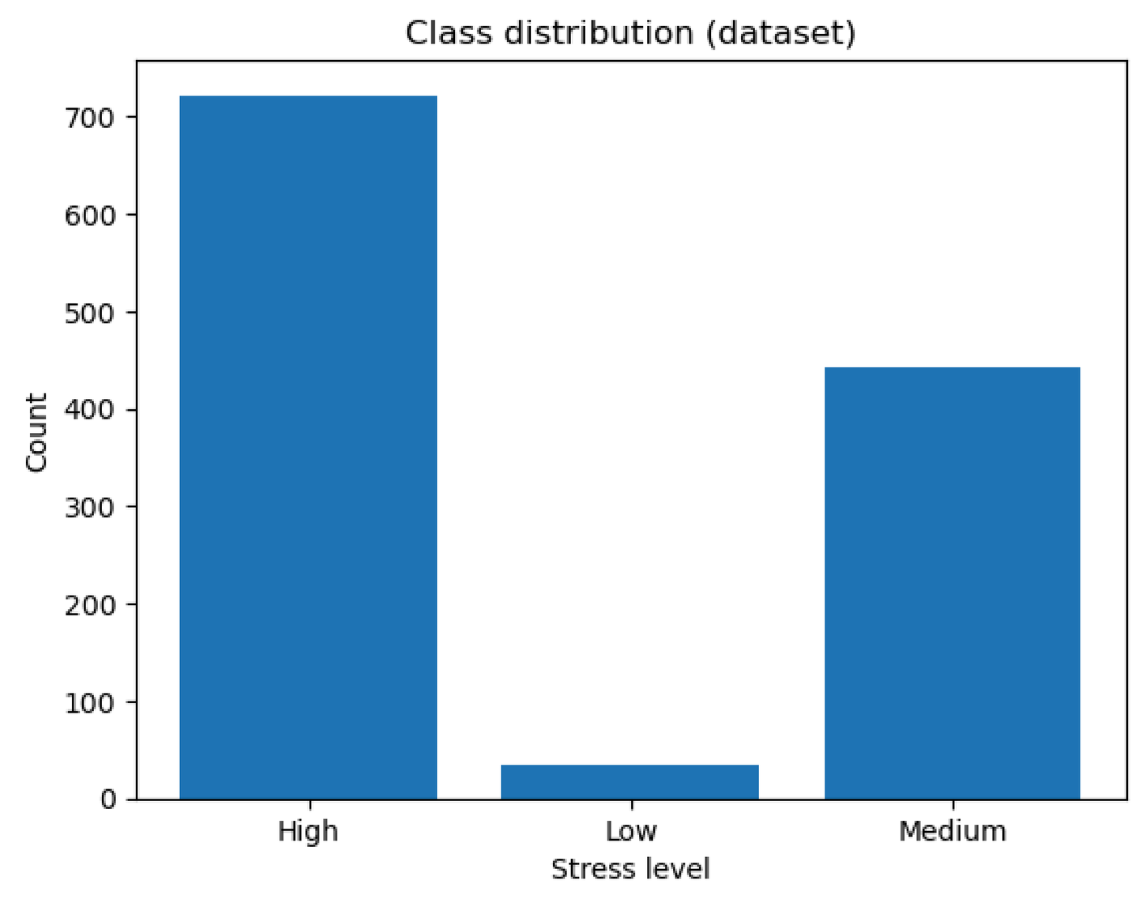
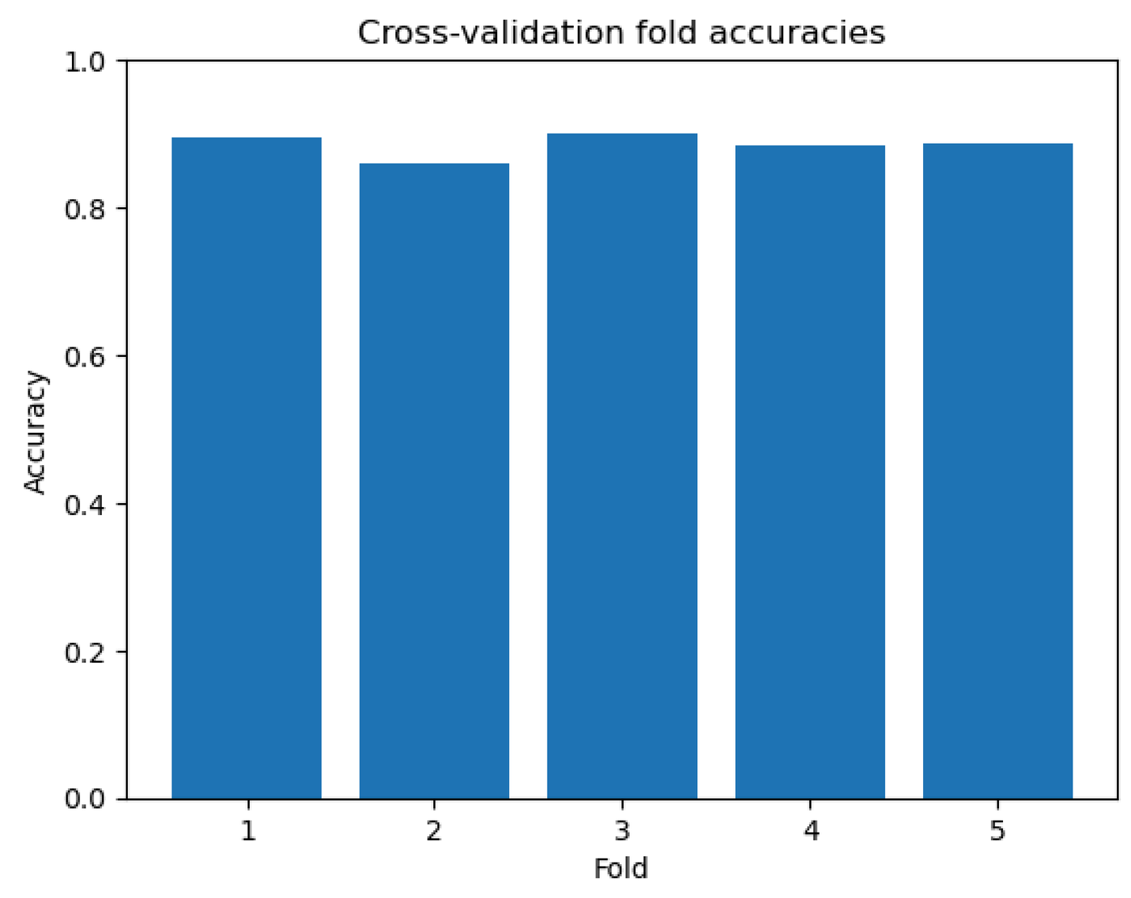
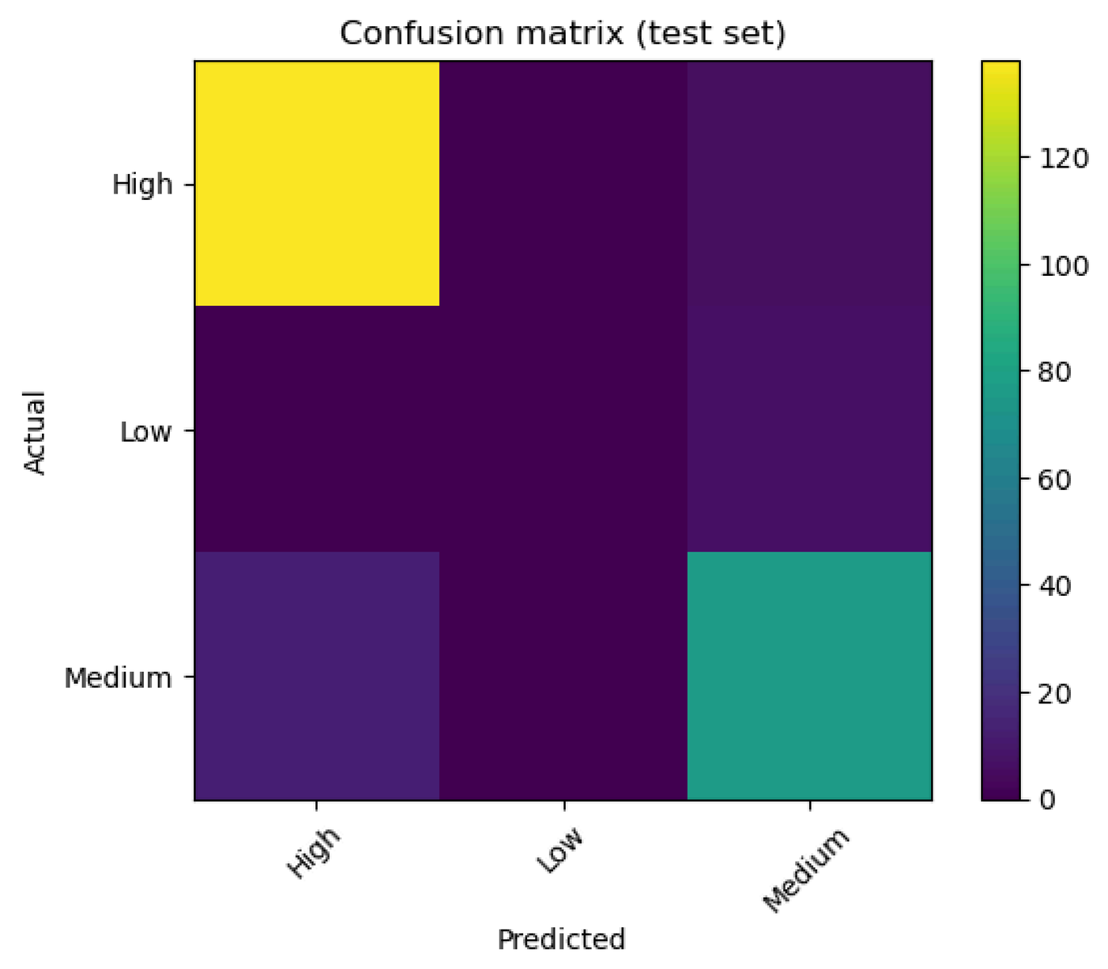
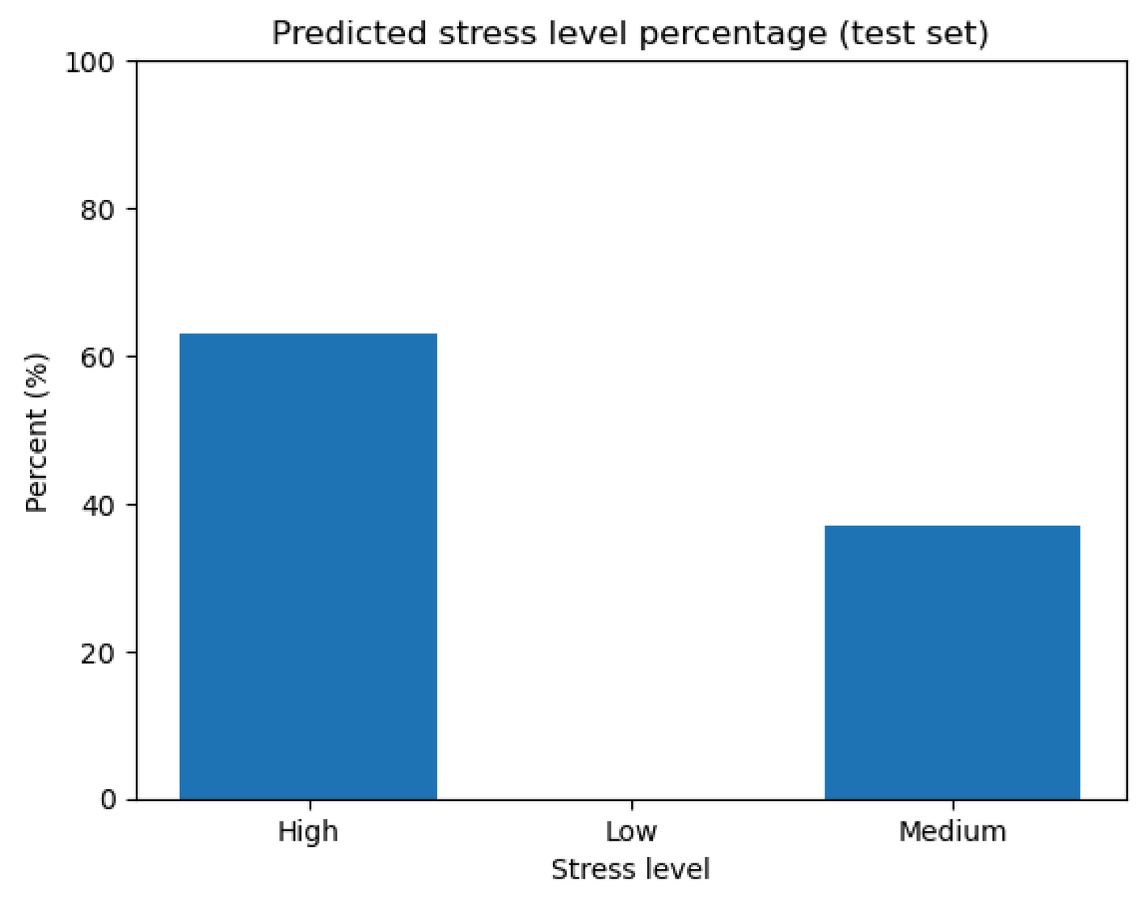
In [3]:
```python
# train_stress_monitor.py
import os, glob
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import joblib
def try_load_csv_dataset(path):
    if not os.path.exists(path):
        return None
    csvs = glob.glob(os.path.join(path, "**", "*.csv"), recursive=True)
    if not csvs:
        return None
    dfs = []
    for f in csvs:
        try:
            dfs.append(pd.read_csv(f))
        except Exception as e:
            print(f"Warning: failed to read {f}: {e}")
    if not dfs:
        return None
    return pd.concat(dfs, ignore_index=True)

df = try_load_csv_dataset(dataset_dir)
loaded_from = "disk" if df is not None else "synthetic"

if df is None:
    # synthetic fallback dataset
    np.random.seed(42)
    n = 1200
    heart_rate = np.random.normal(75,10,n).clip(40,160)
    hrv = np.random.normal(50,15,n).clip(5,150)
    gsr = np.random.exponential(0.6,n)
    skin_temp = np.random.normal(33,1.2,n).clip(28,38)
    accel = np.abs(np.random.normal(0.5,0.7,n))
    screen_time = np.random.exponential(2.0,n)
    sleep_hours = np.random.normal(6.5,1.2,n).clip(2,10)
    app_usage = np.random.poisson(6,n)
    score = 0.04*(heart_rate-60) - 0.03*(hrv-60) + 0.9*gsr + 0.2*accel - 0.18*(sleep_hours-7) + 0.03*(screen_time-2)
    label = np.where(score > 1.4, "High", np.where(score > 0.2, "Medium", "Low"))
    df = pd.DataFrame({
        "heart_rate": heart_rate,
        "hrv": hrv,
        "gsr": gsr,
        "skin_temp": skin_temp,
        "accel": accel,
        "screen_time": screen_time,
        "sleep_hours": sleep_hours,
        "app_usage": app_usage,
        "label": label
    })
else:
    # harmonize label column
    possible_label_cols = [c for c in df.columns if c.lower() in ("label","stress","stress_level","stresslevel")]
    if not possible_label_cols:
        raise ValueError("Dataset loaded but no label column found. Expected 'label' or 'stress' column.")
    df = df.rename(columns={possible_label_cols[0]: "label"})

print(f"Dataset source: {loaded_from}, shape: {df.shape}")

# select features
required_features = ["heart_rate","hrv","gsr","skin_temp","accel","screen_time","sleep_hours","app_usage"]
features = [f for f in required_features if f in df.columns]
if len(features) < 4:
    numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
    features = [c for c in numeric_cols if c != "label"]
    print("Using numeric columns as features:", features)

X = df[features].copy()
y = df["label"].astype(str).copy()

le = LabelEncoder()
y_enc = le.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_enc, test_size=0.2, random_state=42, stratify=y_enc)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = RandomForestClassifier(n_estimators=150, random_state=42)
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)

acc = accuracy_score(y_test, y_pred)
print("\nTest accuracy: {:.3f}".format(acc))
print("\nClassification report:\n")
print(classification_report(y_test, y_pred, target_names=le.classes_))

cm = confusion_matrix(y_test, y_pred)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, scaler.transform(X), y_enc, cv=cv)
print("Cross-validation accuracies:", np.round(cv_scores,3))
print("CV mean accuracy: {:.3f}".format(cv_scores.mean()))

# Plot 1: class distribution
counts = pd.Series(y).value_counts().reindex(le.classes_).fillna(0)
plt.figure()
plt.bar(counts.index, counts.values)
plt.title("Class distribution (dataset)")
plt.xlabel("Stress level")
plt.ylabel("Count")
plt.show()

# Plot 2: CV fold accuracies
plt.figure()
plt.bar(range(1, len(cv_scores)+1), cv_scores)
plt.title("Cross-validation fold accuracies")
plt.xlabel("Fold")
plt.ylabel("Accuracy")
plt.ylim(0, 1)
plt.show()

# Plot 3: Confusion matrix
plt.figure()
plt.imshow(cm, interpolation='nearest')
plt.title("Confusion matrix (test set)")
plt.colorbar()
plt.xticks(range(len(le.classes_)), le.classes_, rotation=45)
plt.yticks(range(len(le.classes_)), le.classes_)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Plot 4: Predicted percentages on the test set
pred_counts = pd.Series([le.classes_[p] for p in y_pred]).value_counts().reindex(le.classes_).fillna(0)
pred_perc = 100 * pred_counts / pred_counts.sum()
plt.figure()
plt.bar(pred_perc.index, pred_perc.values)
plt.title("Predicted stress level percentage (test set)")
plt.xlabel("Stress level")
plt.ylabel("Percent (%)")
plt.ylim(0, 100)
plt.show()

# Save model and scaler
joblib.dump(model, "rf_stress_model.joblib")
joblib.dump(scaler, "scaler.joblib")
print("Saved rf_stress_model.joblib and scaler.joblib")
```

```
Dataset source: synthetic, shape: (1200, 9)

Test accuracy: 0.892

Classification report:

              precision    recall  f1-score   support

        High       0.91      0.96      0.94       144
         Low       0.00      0.00      0.00         7
      Medium       0.85      0.85      0.85        89

    accuracy                           0.89       240
   macro avg       0.59      0.60      0.60       240
weighted avg       0.87      0.89      0.88       240

C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Cross-validation accuracies: [0.896 0.858 0.9   0.883 0.888]
CV mean accuracy: 0.885
```










```
Saved rf_stress_model.joblib and scaler.joblib
```

In [ ]:

In [ ]: