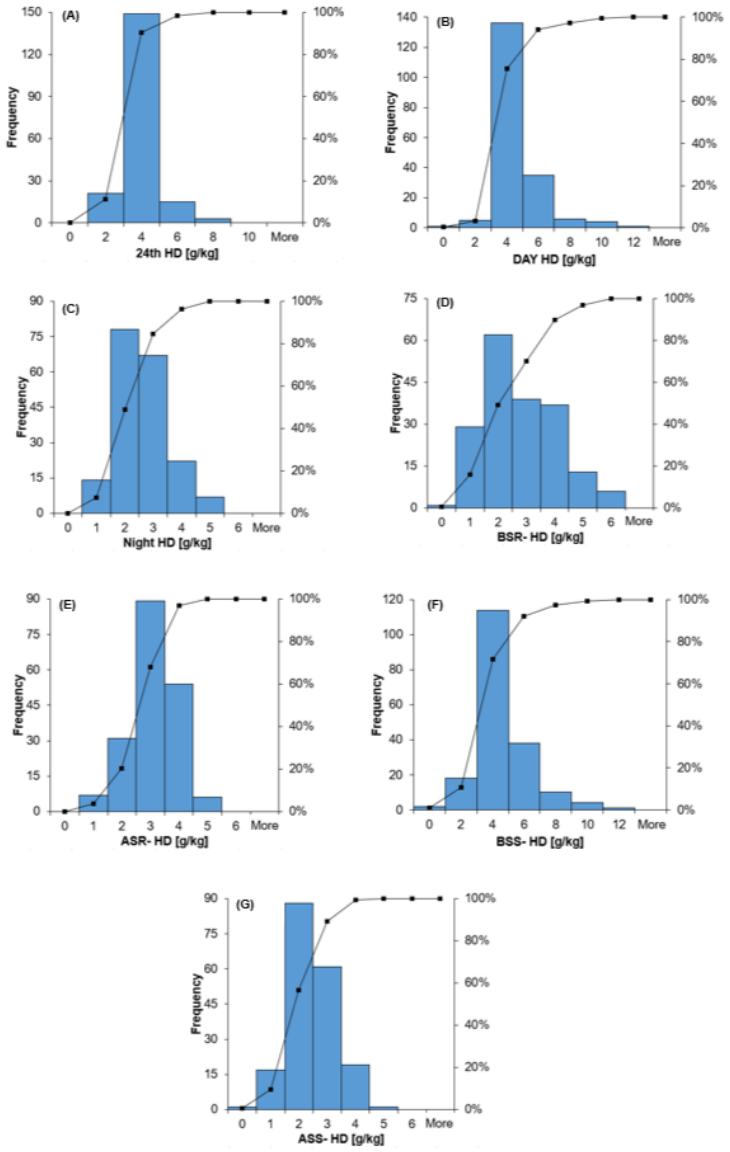


Figure S1. Frequency histograms showing the observed distributions of remaining 7 environmental predictors and one response variable ($P_{n.f.}$). The predictors only include humidity deficit (HD; A–G). Each histogram represents the frequency distribution of the collected observations within the measured range of the corresponding variables.

```
    print("Dataset shape:", df.shape)
    print("\nFirst few rows:")
    print(df.head())

    # Identify target variable (last column)
    target_column = df.columns[-1]
    print(f"\nTarget variable: '{target_column}'")

    # Split chronologically: first 75% for training, last 25% for testing
    train_size = int(len(df) * 0.75)
    print(f"\nTotal rows: {len(df)}")
    print(f"Training rows: first {train_size} rows (75%)")
    print(f"Testing rows: last {len(df) - train_size} rows (25%)")

    # Split the data (maintaining original order)
    train_data = df.iloc[:train_size]  # First 75%
    test_data = df.iloc[train_size:]   # Last 25%

    # Separate features and target for training
    X_train = train_data.drop(columns=[target_column])
    y_train = train_data[target_column]

    # Separate features and target for testing
    X_test = test_data.drop(columns=[target_column])
    y_test = test_data[target_column]

    print(f"\nTraining set: {X_train.shape[0]} samples")
    print(f"Testing set: {X_test.shape[0]} samples")

# Create and train Random Forest model
rf_model = RandomForestRegressor(
    n_estimators=100,
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    random_state=42,
    n_jobs=-1
)

# Train the model
print("\nTraining Random Forest model...")
rf_model.fit(X_train, y_train)
print("Model training completed!")

# Make predictions on both training and testing sets
y_train_pred = rf_model.predict(X_train)
y_test_pred = rf_model.predict(X_test)

# Combine ALL predictions (training + testing) into one DataFrame
all_actual = pd.concat([y_train, y_test], ignore_index=True)
all_predicted = np.concatenate([y_train_pred, y_test_pred])


# Create final DataFrame with 2 columns: Actual and Predicted
final_results = pd.DataFrame({
    'Actual': all_actual,
    'Predicted': all_predicted
})

# Add a column to identify which rows are training and which are testing
final_results['Dataset'] = ['Training'] * len(y_train) + ['Testing'] * len(y_test)

# Save to CSV
final_results.to_csv('final_actual_vs_predicted.csv', index=False)
print("\n        Final CSV saved to 'final_actual_vs_predicted.csv'")
print(f"Total rows: {len(final_results)}")
print(f"  - Training rows: {len(y_train)}")
print(f"  - Testing rows: {len(y_test)}")

# Display first and last few rows
print("\n" + "="*60)
print("FIRST 10 ROWS (Training)")
print("="*60)
print(final_results.head(10))

print("\n" + "="*60)
print("LAST 10 ROWS (Testing)")
print("="*60)
print(final_results.tail(10))

# Model evaluation
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
train_rmse = np.sqrt(mean_squared_error(y_train, y_train_pred))
test_rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))
```

Figure S2. Script of RF model generated by Python praogramming tool for environmental factors versus Paprika fruit production ($P_{n.f.}$).

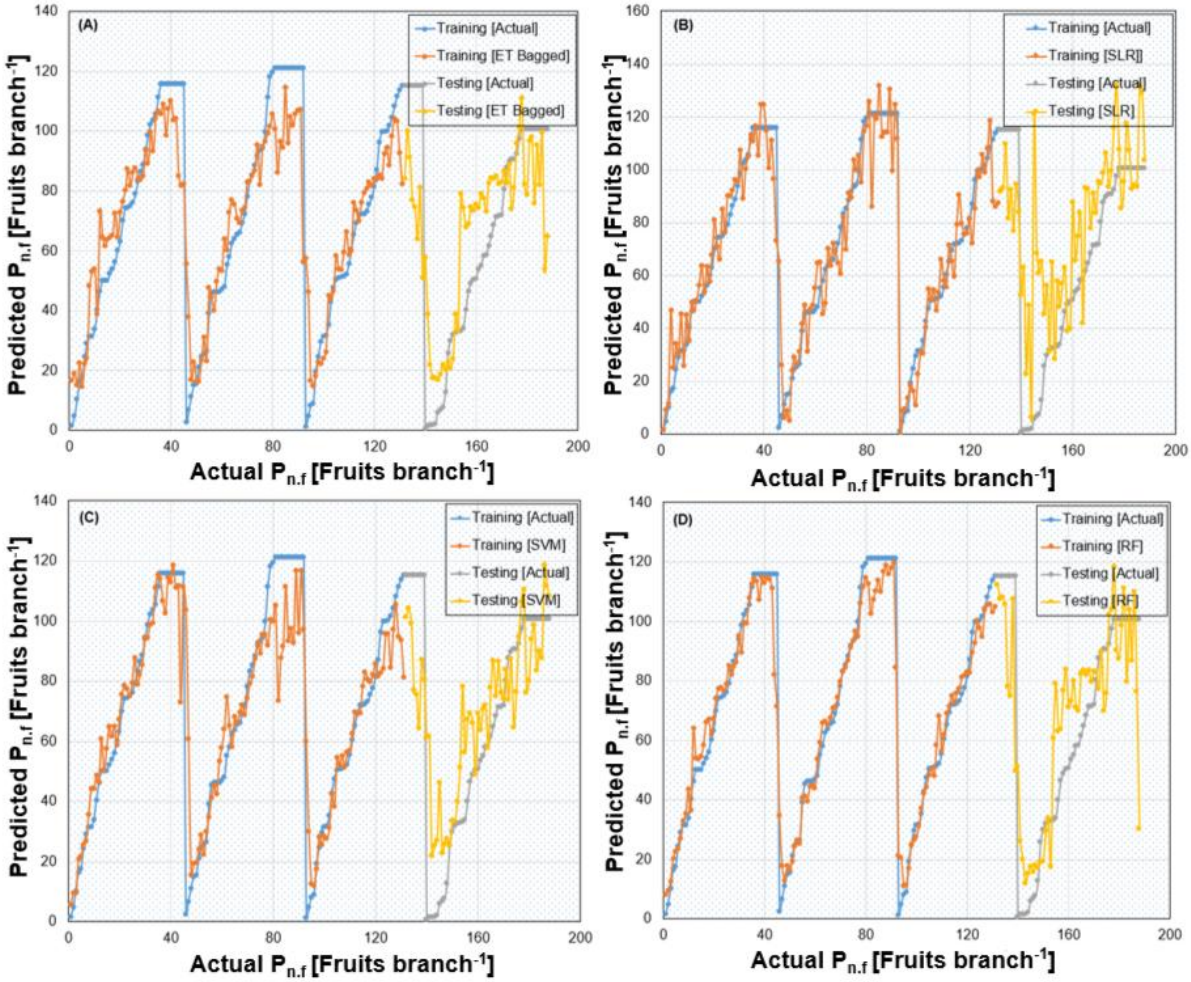


Figure S3. Comparison of the predictive competences of ET Bagged, MLR, SVM and RF models by using training and testing dataset: (A) ET Bagged model; (B) MLR; (C) SVM; (D) RF