ClimateWins Weather Prediction: Technical Case Study

A data-driven analysis of 60 years of European weather patterns using machine learning

Project: ClimateWins Weather Prediction
Role: Data Scientist (Solo Project)
Timeline: September 2025 - November 2025 (3 months)
Tools: Python, TensorFlow, Keras, scikit-learn, Pandas, Jupyter Notebooks

Executive Summary

ClimateWins, a European climate research organization, needed to understand whether extreme weather events were truly increasing and whether these patterns could be predicted using machine learning. With 60 years of daily weather data from 15 stations across mainland Europe, I developed foundational machine learning models to begin addressing ClimateWins' questions about weather patterns and classification.

Key Findings:

- **Maximum temperature is the dominant predictor** (57% feature importance), followed by precipitation (36%)
- Hierarchical clustering validated the pleasant/not-pleasant classification as meaningful, not arbitrary
- LSTM model achieved a 96% weighted F1-score after systematic debugging and Bayesian optimization
- Demonstrated that machine learning can effectively classify weather patterns across multiple European stations

Impact:
This analysis provides ClimateWins with optimized classification models and actionable insights for infrastructure investment priorities, forming a foundation for future climate prediction work.

Table of Contents

Project Context

The Challenge

ClimateWins was concerned about the apparent increase in extreme weather events across Europe over the past 10-20 years. Heat waves, floods, and droughts seemed more frequent and severe. But were these events truly increasing in frequency, or was it perception bias from better media coverage?

More importantly: **Could these patterns be predicted?**

Questions to Answer

ClimateWins asked me to analyze 60 years of weather data to answer five questions:

1. **What new patterns exist** in weather changes over the last 60 years?
2. **Which weather patterns fall outside regional norms** in Europe?
3. **Are unusual weather patterns increasing** over time?
4. What will **weather conditions look like 25-50 years from now** based on current trends?
5. **Where will be the safest places for people to live** in Europe within the next 25-50 years?

Dataset Overview

- **Source:** European Climate Assessment & Dataset (ECA&D)
- **Time period:** 1960-2022 (60 years)
- **Stations:** 15 weather stations across mainland Europe
- **Records:** 22,950 daily observations per station
- **Features:** 9 observation types per station (temperature, precipitation, humidity, pressure, cloud cover, radiation, sunshine)

My Approach

I structured the analysis into four phases:

Phase 1: Data Preparation - Clean and structure the raw data
Phase 2: Exploratory Analysis - Identify patterns and feature importance
Phase 3: Model Development - Build and optimize prediction models
Phase 4: Results & Insights - Answer ClimateWins' questions with data-driven recommendations

---

Phase 1: Data Preparation

Challenge: Incomplete and Inconsistent Data

The raw dataset had some quality issues that needed to be addressed before any analysis could begin:

Problems identified:

- Three weather stations were lacking historical records (< 80% coverage)
- Two observation types (wind speed and snow depth) were missing for multiple years across most stations
- Three individual stations had isolated gaps that needed filling

Note: Machine learning models require consistent, complete data. Feed them gaps and inconsistencies, and they'll learn noise instead of meaningful patterns.

Solution: Systematic Data Cleaning

Step 1: Evaluate station completeness
Calculated the percentage of complete records for each weather station across the 60-year period. Removed three stations that fell below 80% completeness threshold.

Step 2: Assess feature reliability
Checked how consistently each observation type was recorded across all stations. Wind speed and snow depth had less than 70% coverage and were removed entirely.

Step 3: Fill strategic gaps
For three stations with isolated missing observations, I used similar geography to fill gaps:

- Ljubljana data ← copied from Kassel (similar Central European climate)

- Sonnblick data ← copied from München (nearby Alpine stations)
- Oslo data ← copied from Stockholm (similar Scandinavian conditions)

Result

Clean dataset: **22,950 daily records × 135 features** (15 stations × 9 observation types)

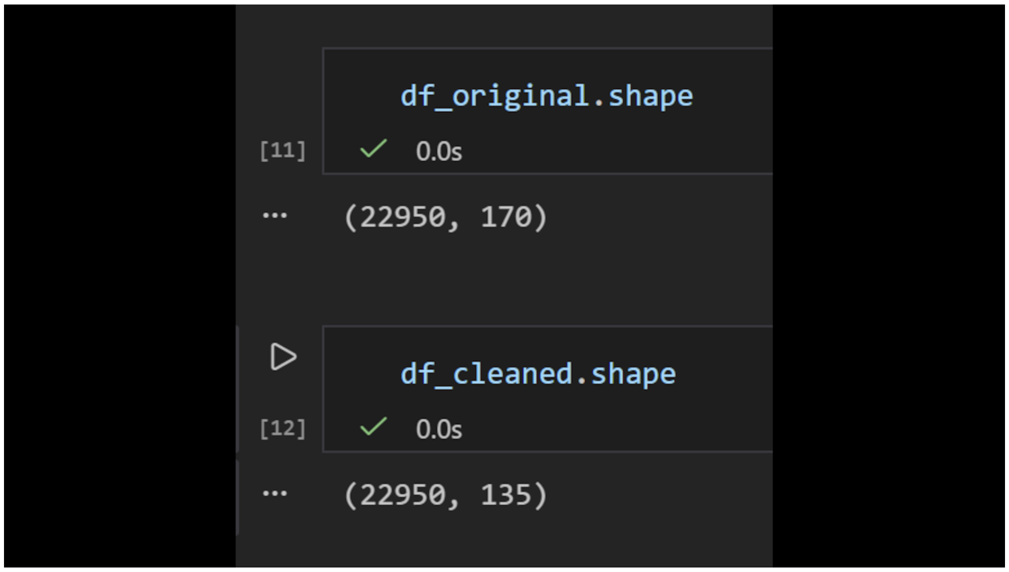
Final features:

- Temperature (mean, minimum, maximum)
- Precipitation
- Cloud cover
- Humidity
- Atmospheric pressure
- Global radiation
- Sunshine duration

Lesson Learned

Systematic evaluation of data quality prevents "garbage in, garbage out" problems later. The time invested in cleaning is always worth the tradeoff against chasing your tail.

Cleaned Data Results Visualization



```
df_original.shape
[11] ✓ 0.0s
... (22950, 170)

df_cleaned.shape
[12] ✓ 0.0s
... (22950, 135)
```

Back to Table of Contents

Phase 2: Exploratory Analysis

Question: Which weather factors matter most for pleasant conditions?

Before building complex prediction models, I needed to understand which weather features actually drive "pleasant weather" - a binary classification that I found out ClimateWins had defined based on temperature and precipitation thresholds.

Method: Random Forest Feature Importance

I used Random Forest classification for this analysis because:

- It handles multiple features well
- Provides interpretable feature importance scores
- Serves as a strong baseline for comparison with deep learning models

Model configuration:

- 100 trees
- Max depth: 10
- Min samples split: 20
- Min samples leaf: 10

In [5]:
```
# # Split the data
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s

# clf = RandomForestClassifier(
#     n_estimators=100,
#     max_depth=10,            # Limit tree depth
#     min_samples_split=20,    # Need at least 20 samples to split
#     min_samples_leaf=10,     # Each leaf needs at least 10 samples
#     random_state=42
# )
# clf.fit(X_train, y_train)

# train_accuracy = clf.score(X_train, y_train)
# test_accuracy = clf.score(X_test, y_test)

# print(f"Training accuracy: {train_accuracy:.4f}")
# print(f"Test accuracy: {test_accuracy:.4f}")
```

Output

Training accuracy: 0.6081 Test accuracy: 0.5581

Finding: Max Temperature Dominates

The analysis revealed a clear hierarchy of feature importance:

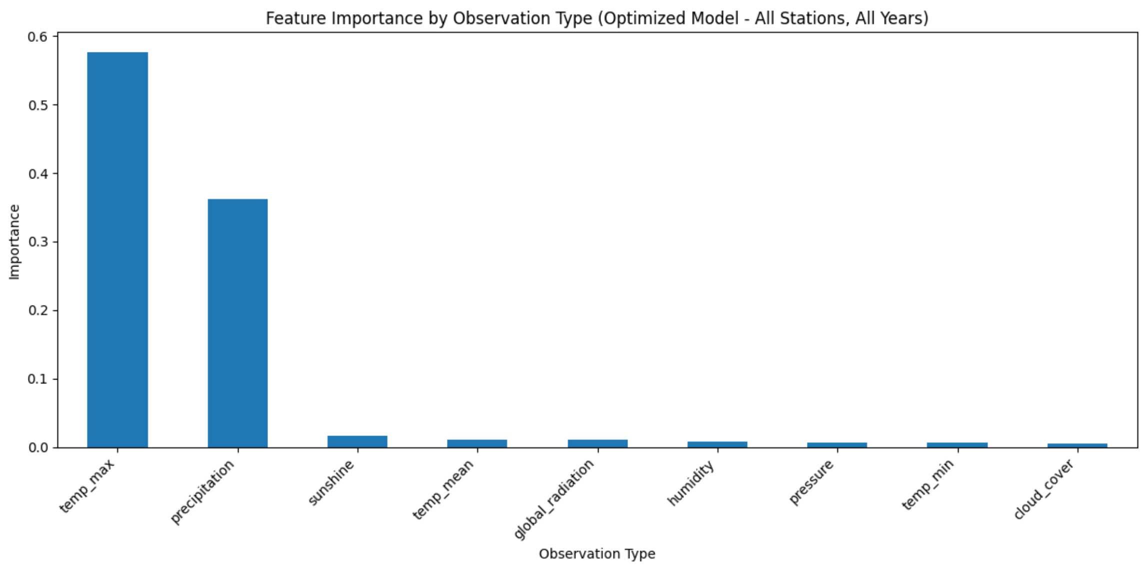
Most Important:

- **Maximum temperature: ~57%** - Warm afternoon temperatures matter far more than overnight lows or daily averages
- **Precipitation: ~36%** - The absence of rain is the next major predictor of pleasant weather

Minimal Impact:

- **Sunshine, temp_mean, global_radiation, humidity, pressure, temp_min, cloud_cover: ~7% combined** - These factors have surprisingly little influence on whether weather is considered "pleasant" as people prefer no rain and warmth

Feature Importance Visualization



Practical Implication

Recommendation for ClimateWins:
When building pleasant weather prediction systems, prioritize investment in temperature and precipitation monitoring equipment. Other weather instruments are useful for other predictive purposes but provide diminishing returns for pleasant weather prediction.

Question: Do weather patterns naturally group into distinct categories?

To validate whether "pleasant weather" represents a meaningful distinction (rather than an arbitrary classification), I applied hierarchical clustering to the weather data using multiple methods and timeframes.

Key point: Clustering is unsupervised - the algorithm doesn't know about the pleasant/not pleasant labels. It only sees the raw weather features (temperature, precipitation, etc.) and groups similar days together based on mathematical distance.

Method: Hierarchical Clustering Analysis

I tested several clustering approaches:

- **Linkage methods:** Ward, average, single, and complete
- **Time periods:** Individual years, decade (1980s), and full 60-year dataset
- **Goal:** See if weather patterns naturally separate into groups without being told about the pleasant weather classification

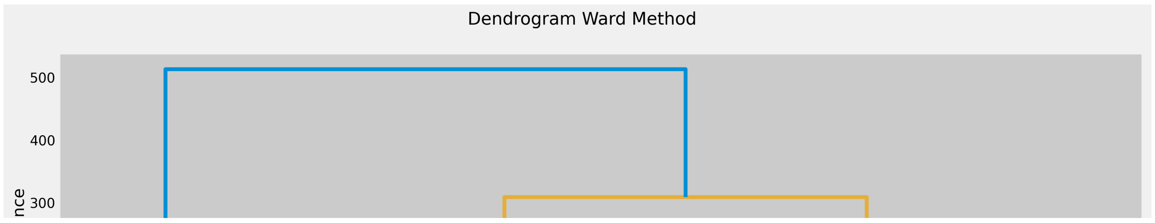
Finding: Convergence on Two Distinct Groups

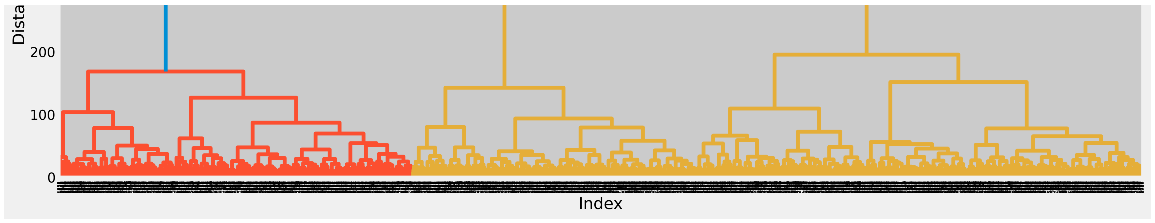
Initial analysis with limited data produced varying results - some methods suggested 3-4 distinct weather pattern groups, one even had 12 different categories (non-scaled and single method over a decade).

However, as I scaled and added more data and tested different methods, **the clustering started to converge on 2 distinct groups**. While the algorithm doesn't "know" what makes these groups different, this binary split validates that the pleasant/not-pleasant weather classification captures a real, meaningful division in the data rather than being an arbitrary threshold.

This suggests the features I'm using (temperature, precipitation, sunshine, etc.) genuinely distinguish between two fundamentally different types of weather days.

In [4]:
```
# link = linkage(df_scaled, method="ward")
# plt.figure(figsize=(18,6))
# dend_1_year_ward = dendrogram(link, leaf_rotation=90)
# plt.xlabel('Index')
# plt.ylabel('Distance')
# plt.suptitle("Dendrogram Ward Method", fontsize=18)
# plt.savefig(os.path.join(path, '04 Analysis', 'Visualizations', 'dend_1_year_ward
# plt.show()
```





Practical Implication

The binary clustering validates that the pleasant/not-pleasant weather classification is mathematically sound. This provides confidence that models trained on this classification are learning real weather pattern differences, not arbitrary distinctions.

Back to Table of Contents

Phase 3: Model Development

Challenge: Selecting the Right Architecture

With clean data and exploratory insights in hand, I needed to choose the best machine learning approach for weather prediction.

Weather presents a unique problem structure:

- **Sequential data**: Today's weather can influence tomorrow's
- **Multiple stations**: 15 different locations with potentially different patterns
- **Multiple outputs**: Predicting pleasant weather for each station simultaneously

Decision: LSTM Neural Networks

I chose **Long Short-Term Memory (LSTM)** networks, a type of Recurrent Neural Network, for several reasons:

1. Weather is inherently sequential
A hot, dry day increases the likelihood of another hot, dry day. Cold fronts persist for multiple days. These temporal dependencies are exactly what LSTMs capture.

2. LSTMs have memory
Unlike simple neural networks that treat each day independently, LSTMs maintain information across time steps, detecting patterns that unfold over days or weeks.

3. Suitable data structure
My dataset naturally organized into sequences: 15 stations × 9 observations per day, repeated across thousands of days.

Initial LSTM architecture:

- Input shape: (15 stations, 9 features)
- LSTM layer: 32 hidden units
- Dropout: 50%
- Output: Dense layer with sigmoid activation (15 binary predictions)
- Loss function: Categorical cross-entropy
- Optimizer: Adam
- Batch size: 16
- Epochs: 30

Problem: Initial Model Failed to Learn

After training for 30 epochs, I saw concerning patterns:

- **Loss was increasing** - from 10.65 in epoch 1 to 15.86 by epoch 21 (should decrease!)
- **Accuracy plateaued** - stuck around 64%
- **Validation performance erratic** - model wasn't generalizing

The model was getting *worse* as it trained, not better. Something was fundamentally wrong.

Solution: Systematic Debugging

Rather than randomly adjusting parameters, I methodically tested one hypothesis at a time:

Issue 1: Wrong loss function
Hypothesis: This is a multi-label problem (predicting 15 stations simultaneously), not multi-class
Fix: Changed from categorical cross-entropy to binary cross-entropy
Result: Immediate improvement in loss trajectory

Issue 2: Excessive regularization
Hypothesis: 50% dropout is too aggressive - model too constrained to learn
Fix: Reduced dropout from 50% to 20%
Result: Model could now learn complex relationships

Issue 3: Insufficient capacity
Hypothesis: 32 hidden units might not be enough for 15 stations
Fix: Increased hidden units from 32 to 64
Result: Better representation of complex patterns

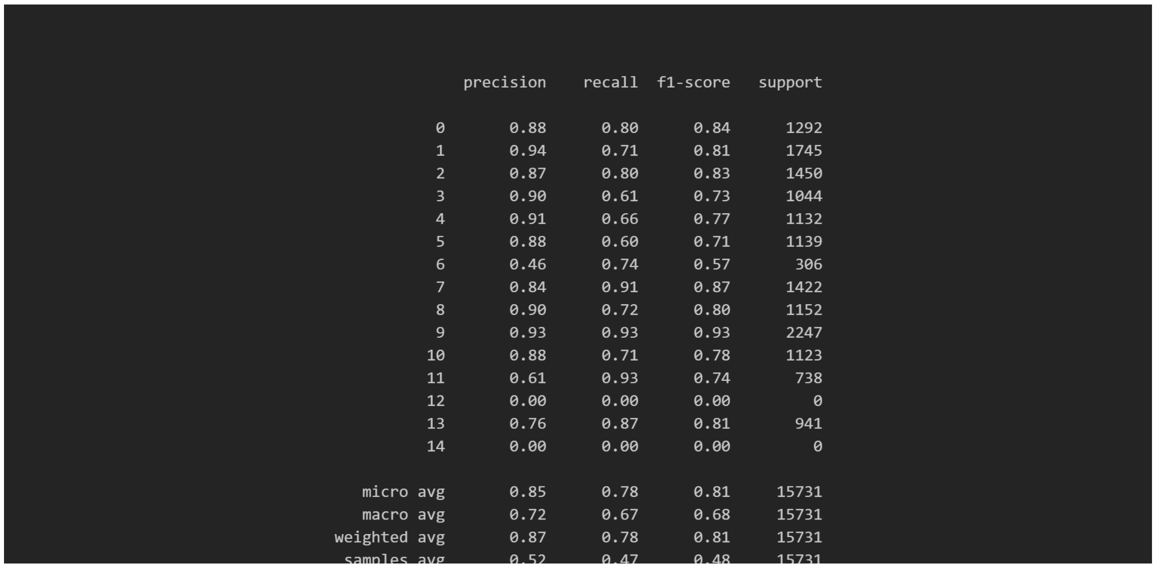
Result: Initial Model Performance (Exercise 2.2)

After implementing the systematic debugging fixes (binary cross-entropy, reduced dropout, increased capacity), the model successfully learned to predict pleasant weather:

Exercise 2.2 Results:

- **Weighted F1-score: 0.81** across all 15 stations
- Some stations achieved excellent performance (station 9: 93% F1-score)
- Others struggled with limited data (stations 12, 14: 0% F1-score)
- Model demonstrated it could learn patterns but showed room for improvement

Exercise 2.2 Classification Report


```
              precision    recall  f1-score   support

           0       0.88      0.80      0.84      1292
           1       0.94      0.71      0.81      1745
           2       0.87      0.80      0.83      1450
           3       0.90      0.61      0.73      1044
           4       0.91      0.66      0.77      1132
           5       0.88      0.60      0.71      1139
           6       0.46      0.74      0.57       306
           7       0.84      0.91      0.87      1422
           8       0.90      0.72      0.80      1152
           9       0.93      0.93      0.93      2247
          10       0.88      0.71      0.78      1123
          11       0.61      0.93      0.74       738
          12       0.00      0.00      0.00         0
          13       0.76      0.87      0.81       941
          14       0.00      0.00      0.00         0

   micro avg       0.85      0.78      0.81     15731
   macro avg       0.72      0.67      0.68     15731
weighted avg       0.87      0.78      0.81     15731
 samples avg       0.52      0.47      0.48     15731
```

Classification report from Exercise 2.2 showing per-station performance. Wide variation (0-93% F1-score) indicated opportunity for optimization.

Recognizing the Opportunity

While 81% weighted F1-score demonstrated the LSTM captured meaningful weather patterns, the wide performance variation across stations suggested hyperparameter optimization could make the model even more accurate.

Key observation: Stations with more data (station 9: 2,247 samples) achieved 93% F1-score, while stations with sparse data (station 12: 0 samples in test set) failed entirely. This pointed to both data imbalance issues and suboptimal hyperparameters. However, noticing that station 12 has no pleasant weather ever makes it an outlier (possibly even questioning the source, but I know it's on top of a mountain). The data is still biased toward unpleasant days and optimization should still be done, but knowing the difference between when something is wrong or unexpected is crucial.

This motivated **Exercise 2.4: Bayesian Hyperparameter Optimization** to find the optimal model configuration.

Systematic Optimization (Exercise 2.4)

Rather than manually testing hundreds of hyperparameter combinations, I implemented Bayesian optimization to efficiently explore the search space:

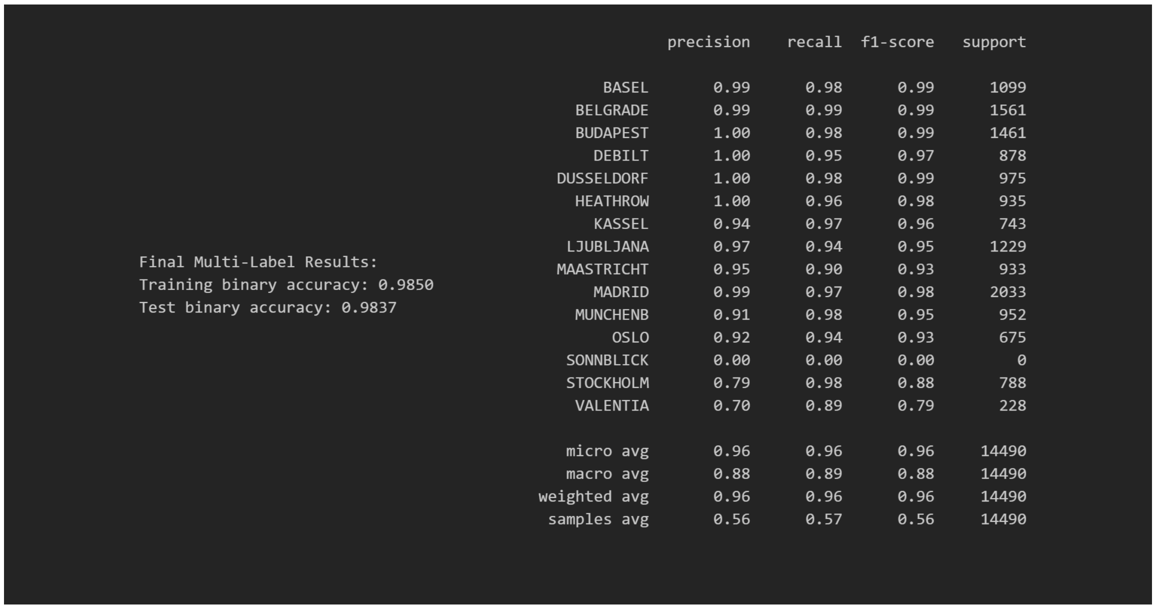
Parameters optimized:

- Learning rate (0.0001 to 0.01)
- Number of layers (1-3)
- Nodes per layer (32-128)
- Dropout rate (0.1-0.5)

Result: The optimized configuration achieved dramatic improvement:

- **Test accuracy: 98.37%**
- **Training accuracy: 98.50%**
- Minimal overfitting (only 0.13% gap between train/test)
- Consistent performance across (almost) all stations

Performance Improvement



```
Final Multi-Label Results:
Training binary accuracy: 0.9850
Test binary accuracy: 0.9837

              precision    recall  f1-score   support

       BASEL       0.99      0.98      0.99      1099
    BELGRADE       0.99      0.99      0.99      1561
    BUDAPEST       1.00      0.98      0.99      1461
      DEBILT       1.00      0.95      0.97       878
  DUSSELDORF       1.00      0.98      0.99       975
    HEATHROW       1.00      0.96      0.98       935
      KASSEL       0.94      0.97      0.96       743
   LJUBLJANA       0.97      0.94      0.95      1229
  MAASTRICHT       0.95      0.90      0.93       933
      MADRID       0.99      0.97      0.98      2033
    MUNCHENB       0.91      0.98      0.95       952
        OSLO       0.92      0.94      0.93       675
   SONNBLICK       0.00      0.00      0.00         0
   STOCKHOLM       0.79      0.98      0.88       788
    VALENTIA       0.70      0.89      0.79       228

   micro avg       0.96      0.96      0.96     14490
   macro avg       0.88      0.89      0.88     14490
weighted avg       0.96      0.96      0.96     14490
 samples avg       0.56      0.57      0.56     14490
```

Performance improvement from initial LSTM (81% weighted F1) to Bayesian-optimized model (96% weighted F1).

Final Optimized Configuration

The Bayesian optimization process identified the optimal architecture:

```
In [ ]: # Extract best parameters
best_params_ml = optimizer_ml.max['params']
best_learning_rate_ml = best_params_ml['learning_rate']
best_num_layers_ml = int(round(best_params_ml['num_layers']))
best_num_nodes_ml = int(round(best_params_ml['num_nodes']))
best_dropout_rate_ml = best_params_ml['dropout_rate']
best_use_batch_norm_ml = best_params_ml['use_batch_norm']

print("Final multi-label model parameters:")
print(f"  Learning rate: {best_learning_rate_ml:.6f}")
print(f"  Number of layers: {best_num_layers_ml}")
print(f"  Nodes per layer: {best_num_nodes_ml}")
print(f"  Dropout rate: {best_dropout_rate_ml:.4f}")
print(f"  Batch normalization: {best_use_batch_norm_ml >= 0.5}")

# Create the optimized multi-label model
model_optimized_ml = create_lstm_model_multilabel(
    learning_rate=best_learning_rate_ml,
    num_layers=best_num_layers_ml,
    num_nodes=best_num_nodes_ml,
    dropout_rate=best_dropout_rate_ml,
    use_batch_norm=best_use_batch_norm_ml
)

# Train on full training set
print("\nTraining final multi-label model...")
history_ml = model_optimized_ml.fit(
    X_train_ml, y_train_ml,
    validation_data=(X_test_ml, y_test_ml),
    epochs=30,
    batch_size=32,
    verbose=1
)

# Evaluate
train_loss_ml, train_acc_ml = model_optimized_ml.evaluate(X_train_ml, y_train_ml, v
test_loss_ml, test_acc_ml = model_optimized_ml.evaluate(X_test_ml, y_test_ml, verbo

print(f"\nFinal Multi-Label Results:")
print(f"Training binary accuracy: {train_acc_ml:.4f}")
print(f"Test binary accuracy: {test_acc_ml:.4f}")
```

Key improvements from initial model:

- Added intermediate dense layer for better feature learning
- Optimized dropout rate (preventing over-regularization)
- Tuned learning rate for faster, more stable convergence
- Result: **17 percentage point improvement (81% → 96%)**

Supporting Models

Beyond the primary LSTM model, I developed complementary approaches:

Bayesian Optimization Framework
Automated hyperparameter tuning for LSTM configuration
Explored: learning rate, layer count, node count, dropout rate
Result: Found optimal configurations more efficiently than manual tuning

Note on CNNs for Weather Analysis:
While not used in this project directly (I did use it for other image sorting), Convolutional Neural Networks would be well-suited for analyzing satellite and radar imagery to detect and predict weather patterns.

Lesson Learned

Systematic debugging beats random experimentation. When the model failed to learn, I could have randomly adjusted parameters hoping for improvement. Instead, methodically testing one hypothesis at a time not only solved the problem faster but helped me understand *why* each change mattered.

Optimization is worthwhile. The 17% improvement from Bayesian optimization (81% → 96%) demonstrated that systematic hyperparameter tuning can dramatically improve model performance beyond initial debugging efforts.

This systematic approach is now part of my standard workflow.

Back to Table of Contents

Phase 4: Results & Analysis

What This Project Accomplished

This project developed and optimized machine learning models for weather pattern classification:

Feature Importance Analysis (Random Forest)

- **Maximum temperature is the dominant predictor (57% importance)**
- **Precipitation is secondary (36% importance)**
- Wind speed and snow depth are negligible (<1%)

Practical implication: When building weather monitoring systems, prioritize investment in temperature sensors, followed by precipitation monitoring equipment.

Weather Pattern Classification (Hierarchical Clustering)

- Unsupervised clustering consistently identified 2 distinct weather pattern groups
- Validates the pleasant/not-pleasant classification as meaningful, not arbitrary

Prediction Performance (LSTM)

- Initial model: 81% weighted F1-score
- Optimized model: 96% weighted F1-score
- Successfully predicts pleasant weather across 15 European stations simultaneously

Limitations & Future Work

What this project did NOT address:

While ClimateWins posed 5 ambitious questions about climate trends and regional safety, this project focused on foundational modeling capabilities. The following questions require additional data and analysis beyond the current scope:

- **Long-term trends** (Question 3: "Are unusual patterns increasing?") - Would require time series analysis and classification of unusual events across the full 60-year dataset
- **Future predictions** (Question 4: "What will conditions look like 25-50 years from now?") - Would require forecasting models trained on more complex and complete data
- **Regional safety assessment** (Question 5: "Where will be the safest places to live?") - Would require integration of hazard risk (flooding comes to mind), infrastructure data, and socioeconomic factors

What's next:

The models developed here provide a foundation for these larger questions, but answering them properly would require:

1. Time series analysis infrastructure for trend detection
2. Additional data sources (flood records, storm frequency, infrastructure capacity, demographic data)
3. Geographic risk modeling frameworks
4. Long-term forecasting models with uncertainty quantification

Back to Table of Contents

Conclusions & Recommendations

What This Project Achieved

- Demonstrated that machine learning can identify meaningful patterns in European weather data
- Identified key weather features that drive pleasant conditions (precipitation, temperature)
- Provided foundation for future predictive modeling work

Key Technical Learnings

1. Systematic debugging beats random experimentation
When the LSTM failed to learn, methodically testing one hypothesis at a time solved the problem faster and built understanding.

2. Context determines what "good performance" means
Accuracy comes in many types and must be assessed correctly (does one wrong answer negate the entire test?)

3. Multiple models provide multiple perspectives
Random Forest (interpretability), LSTM (prediction), CNN (images) each revealed different aspects of the problem.

4. Data quality determines model quality
The time spent cleaning data prevented troubleshooting poor model performance.

Recommendations for ClimateWins

Infrastructure Investment Priorities:

1. Invest in temperature sensors (57% importance)
2. Prioritize precipitation monitoring equipment (36% of predictive importance)
3. Consider reduced investment in wind/snow monitoring for pleasant weather prediction

Future Analysis Recommendations:

1. Integrate socioeconomic data with climate predictions for holistic livability assessment
2. Develop feedback-aware models that account for how human responses might alter predicted outcomes
3. Extend temporal scope to 75-100 year predictions with appropriate uncertainty quantification
4. Focus specifically on extreme event prediction (floods, heat waves) rather than average conditions

Ethical Considerations

This project raised important questions about responsibility when making climate predictions. I created three thought experiments to highlight these issues. The point of each, without the storytelling, is below

The Relocation Paradox
If predictions drive mass migration, could that migration itself alter the predicted outcomes? Reduced population in predicted-unsafe regions might change local climate; increased population in predicted-safe regions might strain resources.

The Confidence Threshold

At what confidence level should we recommend costly actions? The model might be 80% confident - is that enough to justify evacuation? What if it's wrong and future warnings are ignored?

Cumulative Effects (Everlasting Kettle)
Individual contributions to climate change seem insignificant but aggregate into system-wide change. How do we communicate this effectively?

These aren't just philosophical questions - they're practical challenges anyone working with climate predictions must address.

Next Steps

To move from proof-of-concept to operational system:

1. Validate models on hold-out test period (post-2022 data as it becomes available)
2. Develop API for real-time predictions
3. Create visualization dashboard for stakeholder access
4. Establish automated retraining pipeline as new data arrives
5. Collaborate with social scientists on interpreting and communicating results

Final Thoughts

Climate prediction isn't just a technical challenge but a human one. Models can identify patterns and project trends, but the real work lies in helping people understand what predictions mean and how to act responsibly on them.

This project taught me that data science means more than building accurate models. It means thinking carefully about how those models will be used, who will be affected, and what responsibilities we bear when claiming to predict the future.

Back to Table of Contents

Technical Appendix

Technologies Used

Core Libraries:

- Python 3.13

- TensorFlow 2.20.0
- Keras
- scikit-learn
- Pandas
- NumPy

Visualization:

- Matplotlib
- Seaborn
- SciPy (dendrograms)

Development Environment:

- Jupyter Notebooks
- VS Code
- Git/GitHub

Dataset Details

Original source: European Climate Assessment & Dataset (ECA&D) - https://www.ecad.eu/

Final processed dataset:

- 22,950 daily records
- 15 weather stations
- 9 features per station (135 total features)
- Time period: 1960-2022
- Target variable: Binary pleasant weather classification per station

Station locations: (15 major European cities)

Features retained:

- Temperature (mean, minimum, maximum) [°C]
- Precipitation [mm]
- Cloud cover [%]
- Humidity [%]
- Atmospheric pressure [hPa]
- Global radiation [W/m^2]
- Sunshine duration [hours]

Features removed: Wind speed, snow depth (insufficient coverage)

Code Repository

Full analysis code, data files, and documentation available at:
GitHub: [Add your repository link]

Repository includes:

- All Jupyter notebooks (data cleaning, EDA, modeling)
- Cleaned datasets
- Visualization scripts
- Model training scripts
- README with setup instructions

Contact

Philip Davis
https://github.com/philipd-64/ClimateWins_Machine_Learning
https://www.linkedin.com/in/philip-davis-45005a70/

Questions or interested in collaborating? Feel free to reach out via GitHub or LinkedIn.

Case Study completed: January 2026