Solar Irradiance Forecasting Project: Step-by-Step Guide

Phase 1: Project Setup & Literature Review (Week 1-2)

Task 1.1: Environment Setup

- Set up Python environment with required libraries:
  - Data manipulation: `pandas`, `numpy`
  - Visualization: `matplotlib`, `seaborn`, `plotly`
  - Time series: `statsmodels`, `pmdarima`
  - Deep learning: `tensorflow/keras`, `pytorch`
  - Prophet: `prophet` (formerly `fbprophet`)
  - Metrics: `scikit-learn`
- Create project structure:
- ```
  solar_forecasting/├── data/
                    ├── notebooks/
                    ├── src/
                    ├── models/
                    ├── results/
                    └── reports/
  ```

Task 1.2: Literature Review

Focus Areas:

- Time series forecasting for meteorological data
- Solar irradiance prediction methods
- Multivariate time series models
- Deep learning applications in weather forecasting
- Geographical considerations in meteorological modeling

Key Topics to Research:

- ARIMA/SARIMAX for meteorological data
- LSTM/GRU applications in solar forecasting
- Prophet for seasonal meteorological patterns
- Ensemble methods for weather prediction
- Transfer learning across geographical locations
- Handling missing data in meteorological time series

Deliverable: 3-4 page literature review with 15-20 relevant papers

Phase 2: Data Exploration & Preprocessing

Task 2.1: Initial Data Assessment
  ```

- Load and examine dataset structure
- Identify data types, missing values, and anomalies
- Understand temporal resolution (24 readings/day = hourly data)
- Map geographical locations and their characteristics
- Calculate basic statistics for each variable and location

Task 2.2: Exploratory Data Analysis (EDA)

- **Temporal Analysis:**
  - Daily, monthly, seasonal patterns
  - Trend analysis across years (2005-2020)
  - Autocorrelation and partial autocorrelation plots
- **Spatial Analysis:**
  - Compare patterns across different towns
  - Identify geographical clusters or similarities
- **Variable Relationships:**
  - Correlation matrices between meteorological variables
  - Lag correlations (how past weather affects future irradiance)
  - Principal Component Analysis (PCA) if needed

Task 2.3: Data Preprocessing

- **Missing Data Handling:**
  - Identify patterns of missingness
  - Apply appropriate imputation methods (forward fill, interpolation, or advanced methods)
- **Outlier Detection and Treatment:**
  - Use statistical methods (IQR, Z-score) or domain knowledge
  - Decide on removal vs. transformation
- **Feature Engineering:**
  - Create time-based features (hour, day of week, month, season)
  - Calculate moving averages and rolling statistics
  - Create lag features for predictors
  - Generate clear sky index or other solar-specific features
- **Data Standardization:**
  - Scale features appropriately for different models
  - Handle different units across variables

Deliverable: Clean dataset with comprehensive EDA report and visualizations

Phase 3: Baseline Model Implementation

Task 3.1: Simple Baselines

- **Persistence Model:** Tomorrow's irradiance = today's irradiance
- **Seasonal Naive:** Use same hour from previous day/week
- **Moving Average:** Simple and weighted moving averages
- **Linear Regression:** Basic multivariate linear model

Task 3.2: Model Validation Framework

- **Time Series Cross-Validation:**
  - Use walk-forward validation
  - Maintain temporal order (no future data leakage)
- **Train/Validation/Test Split:**
  - Training: 2005-2017 (70%)
  - Validation: 2018-2019 (20%)
  - Test: 2020 (10%)
- **Evaluation Metrics:**
  - RMSE, MAE, MAPE
  - R^2 and adjusted R^2
  - Directional accuracy
  - Skill scores relative to persistence

Deliverable: Baseline results and validation framework

Phase 4: Advanced Model Implementation

Task 4.1: ARIMA/SARIMAX Implementation

- **Univariate ARIMA:**
  - Use auto-ARIMA for parameter selection
  - Test on individual locations first
- **SARIMAX (Multivariate):**
  - Include exogenous variables (temperature, humidity, wind speed)
  - Test different seasonal components
  - Handle multiple locations (separate models vs. pooled)

Task 4.2: Deep Learning Models

- **LSTM Implementation:**
  - Design architecture (single vs. multi-layer)
  - Experiment with sequence lengths
  - Implement attention mechanisms if beneficial
- **GRU Alternative:**
  - Compare performance with LSTM
  - Test computational efficiency
- **Considerations:**
  - Handle multiple locations (separate models vs. shared layers)
  - Implement proper regularization (dropout, early stopping)
  - Use appropriate loss functions

Task 4.3: Prophet Implementation

- **Basic Prophet:**
  - Handle seasonality (daily, weekly, yearly)
  - Include holiday effects if relevant
- **Prophet with Regressors:**

    - Add meteorological variables as additional regressors
    - Test different approaches for multiple locations

Deliverable: Three working models with initial performance metrics

Phase 5: Model Optimization & Comparison

Task 5.1: Hyperparameter Tuning

- **SARIMAX:** Grid search for (p,d,q) and seasonal parameters
- **LSTM/GRU:**
    - Architecture optimization (layers, units, dropout)
    - Learning rate scheduling
    - Batch size and sequence length tuning
- **Prophet:** Seasonality parameters and regressor coefficients

Task 5.2: Comprehensive Evaluation

- **Performance Metrics:**
    - Calculate all metrics across different time horizons (1-hour, 6-hour, 24-hour ahead)
    - Evaluate performance by season and location
- **Statistical Testing:**
    - Diebold-Mariano test for forecast accuracy comparison
    - Residual analysis and diagnostic tests
- **Visualization:**
    - Forecast plots with confidence intervals
    - Error distribution analysis
    - Feature importance plots (where applicable)

Deliverable: Comprehensive model comparison report

Phase 6: Hybrid/Improved Model Development

Task 6.1: Identify Improvement Opportunities

- Analyze where each model performs best/worst
- Identify patterns in residuals
- Consider ensemble approaches

Task 6.2: Develop Hybrid Model

Potential Approaches:

- **Ensemble Methods:**
    - Simple averaging or weighted combinations
    - Stacking with meta-learner
- **Hierarchical Models:**

- Global model with location-specific adjustments
    - Multi-task learning for shared patterns
- **Hybrid Architectures:**
    - LSTM with ARIMA residual modeling
    - Prophet trend + LSTM for short-term patterns

Task 6.3: Model Justification

- Provide theoretical reasoning for hybrid approach
- Demonstrate improvement over individual models
- Analyze computational trade-offs

Deliverable: Novel hybrid model with performance justification

Phase 7: Geographic Analysis & Insights

Task 7.1: Location-Specific Analysis

- Compare model performance across different towns
- Identify geographical patterns in forecast accuracy
- Analyze climate zone effects on model performance

Task 7.2: Transfer Learning Investigation

- Test models trained on one location and applied to others
- Investigate domain adaptation techniques
- Analyze what makes some locations harder to predict

Task 7.3: Practical Considerations

- Discuss data availability challenges in different regions
- Consider computational requirements for deployment
- Address scalability for additional locations

Deliverable: Geographic insights and practical deployment considerations

Phase 8: Final Report & Documentation

Task 8.1: Results Compilation

- Create comprehensive results tables and visualizations
- Prepare executive summary of findings
- Document all code and methodologies

Task 8.2: Final Report Structure

1. **Executive Summary**
2. **Introduction & Literature Review**

3. **Data Description & Preprocessing**
4. **Methodology**
5. **Results & Analysis**
6. **Proposed Hybrid Model**
7. **Geographic Insights**
8. **Conclusions & Future Work**
9. **Appendices**

Task 8.3: Code Documentation

- Clean and comment all code
- Create reproducible notebooks
- Write README with setup instructions
- Prepare requirements.txt

Deliverable: Complete project report and reproducible codebase

Success Metrics & Quality Checkpoints

- Literature review complete
- EDA insights documented
- Baseline models running
- All three main models implemented
- Performance comparison complete
- Hybrid model developed
- Geographic analysis complete
- Final report ready

Quality Standards:

- All code must be reproducible and well-documented
- Models must be properly validated using time series techniques
- Results must include statistical significance tests
- Visualizations must be publication-ready
- Report must be suitable for academic or industry submission

Tools & Resources Recommendations

Essential Libraries:

```
# Data manipulation
import pandas as pd
import numpy as np

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go

# Time series
from statsmodels.tsa.arima.model import ARIMA
```

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
import pmdarima as pm

# Deep learning
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, GRU, Dense, Dropout

# Prophet
from prophet import Prophet

# Metrics and evaluation
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

Recommended Computing Resources:

- Minimum 8GB RAM for deep learning models
- GPU recommended for LSTM/GRU training
- Cloud platforms (Google Colab, AWS) for computational flexibility