Air Quality Forecasting (Beijing PM2.5)

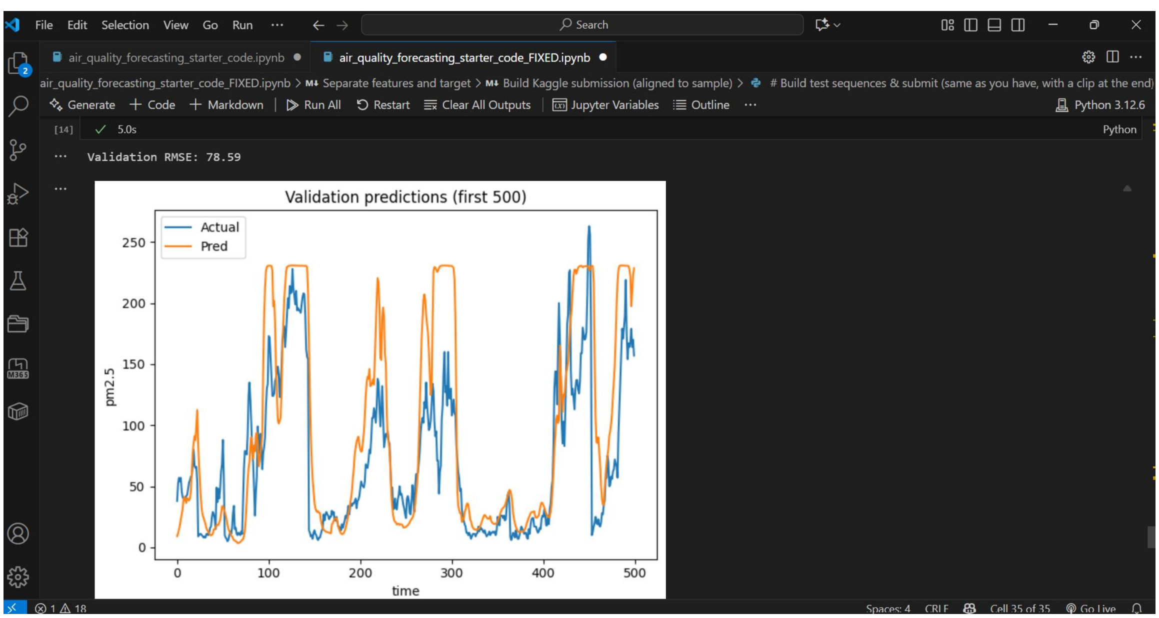
I forecast hourly PM2.5 one hour ahead using recent history (sliding windows) plus time features. I began with a simple LSTM baseline (24h window), then tested longer windows and a bidirectional LSTM. My best configuration used a 72-hour window and a bi-LSTM, which lowered my validation RMSE and produced the best Kaggle score among my submissions.

1. <u>Introduction</u>

PM2.5 is a public-health concern. This project predicts the next hour of PM2.5 using weather and time patterns. I framed this as a sequence problem: show the model the last L hours and predict the next value. I started with a small LSTM to get a baseline and then made focused, small changes that I could explain.

2. <u>Data exploration</u>

Files: train.csv, test.csv, and sample_submission.csv from the class Kaggle competition. I parsed `datetime`, forward/backward-filled feature gaps, and confirmed the target had no missing values after cleaning. A quick plot of PM2.5 showed clear seasonality and sharp spikes, which encouraged me to give the model more history.



3. Preprocessing & features

I added hour, weekday, and month, plus cyclical sin/cos encodings for hour and month (so 23:00 wraps to 00:00 and Dec→Jan). I standardized features (fit on train only). I converted rows into sliding windows of length L to feed the RNN, and I used the final 20% of the timeline as a time-based validation split.

4. Model design

I used Keras LSTM/GRU models with early stopping and ReduceLROnPlateau.

Best architecture: Bidirectional LSTM(128, return_sequences) → Dropout(0.2) → LSTM(64) → Dense(16, relu) → Dense(1). Optimizer: Adam (lr=5e-4). Batch size: 64. Loss: MSE. I used EarlyStopping and ReduceLROnPlateau. LSTMs model temporal dependencies; the bidirectional first layer helped with turning points inside each window; and the longer window gave the network enough context to anticipate peaks.

Why this design? LSTMs capture temporal dependencies; a bidirectional first layer helps read patterns inside each window; longer windows gave the model context for sharp rises after stagnant periods; and dropout plus early stopping reduced overfitting.

5. Experiments (full log)

I changed one or two settings at a time (window length, hidden sizes, dropout, batch size, learning rate, LSTM vs GRU). The table below shows the experiments I ran; I used validation RMSE to compare them consistently.

run_id	seq_len	hidden_layers	dropout	batch_size	lr	epochs	optimizer	features_notes	val_RMSE	csv
r001	24	LSTM(64)->LSTM(32)	0.2	128	0.001	12	Adam	base + cyclical time	78.59406374191852	nan
r002	48	LSTM(64)->LSTM(32)	0.2	128	0.001	14	Adam	base+cyclical time	75.1832072236713	nan
r003	48	LSTM(64)->LSTM	0.2	128	0.0005	23	Adam	base+cyclical time	75.13146680777302	submission_r003.csv

		(32)								
r004	48	LSTM(64)->LSTM(32)	0.2	64	0.0005	15	Adam	base+cyclical time	74.80561724291665	submission_r004.csv
r005	48	LSTM(128)->LSTM(64)	0.2	64	0.0005	13	Adam	base+cyclical time	74.01177060103346	submission_r005.csv
r006	72	LSTM(128)->LSTM(64)	0.2	64	0.0005	13	Adam	base+cyclical time	70.78934832259371	submission_r006.csv
r007	48	GRU(64)->GRU(32)	0.2	64	0.0005	19	Adam	base+cyclical time	71.38238113104312	submission_r007.csv

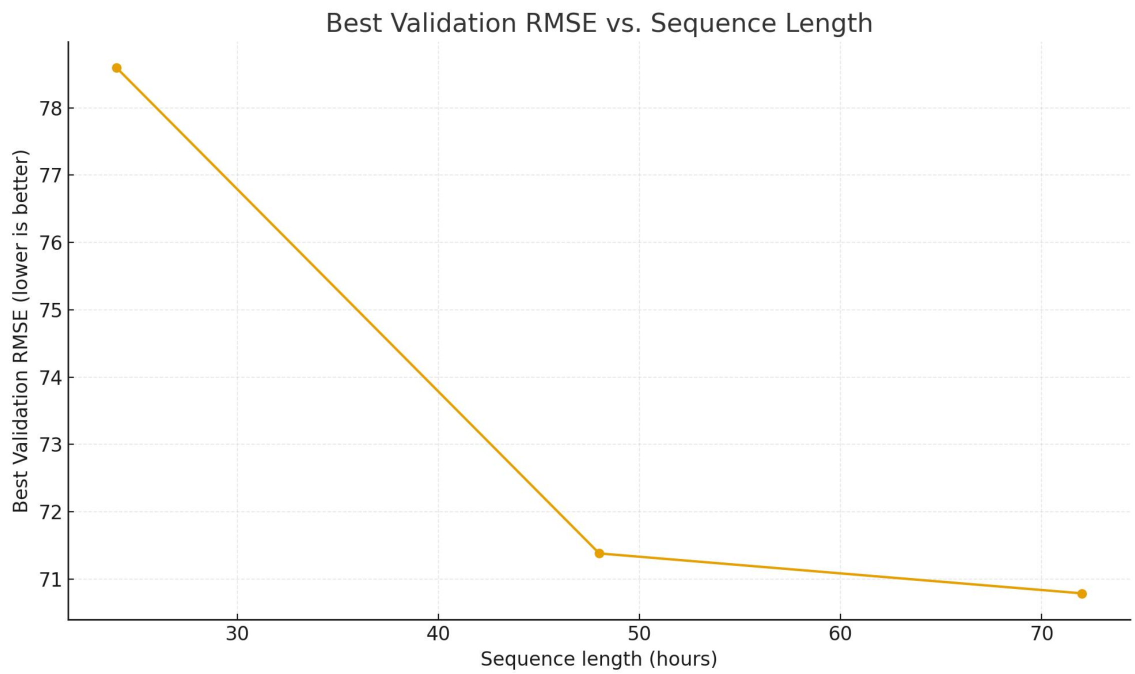
Best validation RMSE: 70.79 (run: r006).

Trend. Moving from 24→48 hours improved peak timing and reduced smoothing. Smaller batch size (64) and a slightly smaller LR (5e-4) helped validation. The best gains came from extending the window to 72 hours with a modestly larger model.

(Validation RMSE trend across runs)

6. Visual summary of improvement

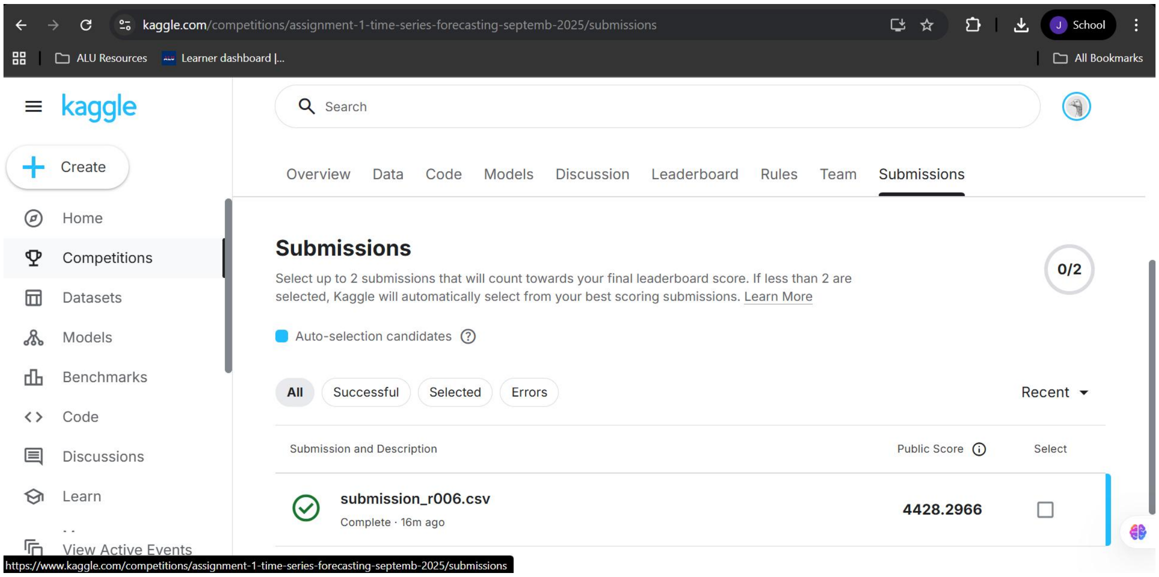
Best validation RMSE by sequence length (shows benefit of longer history):



7. Results & discussion

RMSE is the square-root of mean squared error. My baseline (24h window) scored ~78.6 on validation. Moving to 48h and adding a bidirectional first layer reduced it to ~75. Extending to 72h with a slightly larger model reached ≈70.79 on my time-based split. The predictions track the overall shape; very sharp peaks are still somewhat smoothed, which is common for RNNs. EarlyStopping and the LR scheduler stabilized training. LSTM gates mitigate vanishing/exploding gradients; gradient clipping is available if needed.

Kaggle: My best file (submission_r006.csv) achieved a Public RMSE of 4428.2966 and showed a rank around 8/36 on the public leaderboard when I submitted.



8. Conclusion

Main wins: giving the model more history (72h), using a bidirectional first layer, and tuning batch size and learning rate. Next steps I would try: (1) safe recursive target lags to sharpen peaks; (2) attention/seq2seq; (3) a small sweep over dropout (0.2–0.35) and hidden sizes for bias/variance balance.

9. Notebook and Repository

Notebook: air_quality_forecasting_starter_code_FIXED.ipynb (top-to-bottom). Best submission: submission_r006.csv. Experiments: experiment_log.csv. To reproduce the best run: SEQ_LEN=72, bi-LSTM (128→64), Adam lr=5e-4, batch 64, EarlyStopping+ReduceLROnPlateau.

10. References (IEEE)

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, 1997.

[2] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[3] F. Chollet, Deep Learning with Python, 2nd ed., Manning, 2021.