For this assignment I decided to develop an ML system to predict daily temperature extremes (TMAX/TMIN) with confidence intervals to quantify the uncertainty of the model's predictions.

Steps:

- Data Loading
- Initial Data Visualization
  → Hypothesis Testing for Changing Variance (heteroskedasticity)
  → Model Selection Based on Results

- Feature Selection
- Feature Engineering

- Formatting Data for Training
- Model Hyperparameter Tuning
- Model Training

- Evaluating Performance
- Error Analysis

- Formulating Prediction Function
- “Deploying” GUI Application for Dynamic User Input

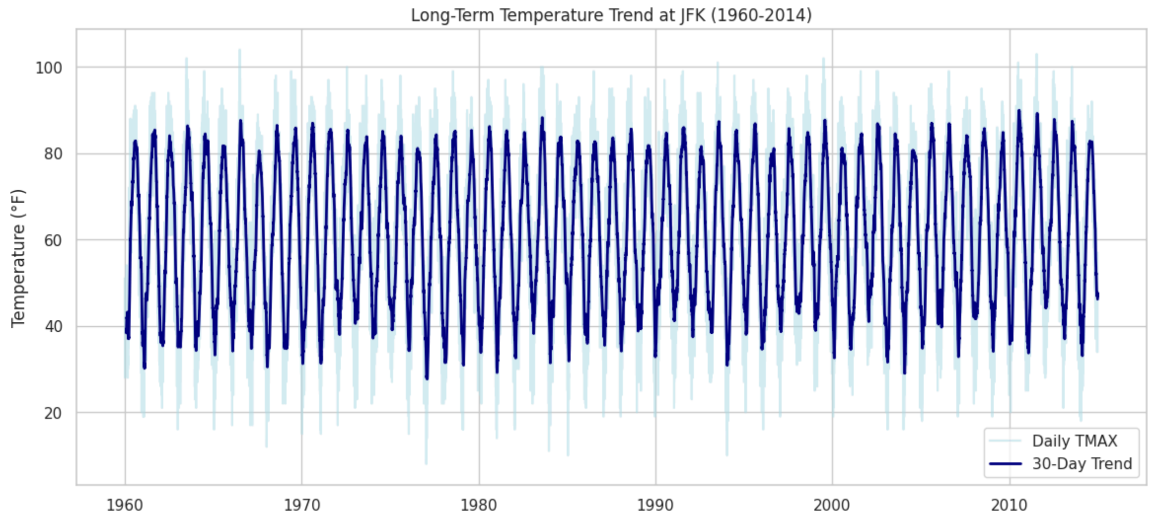
Data Loading and Preparation

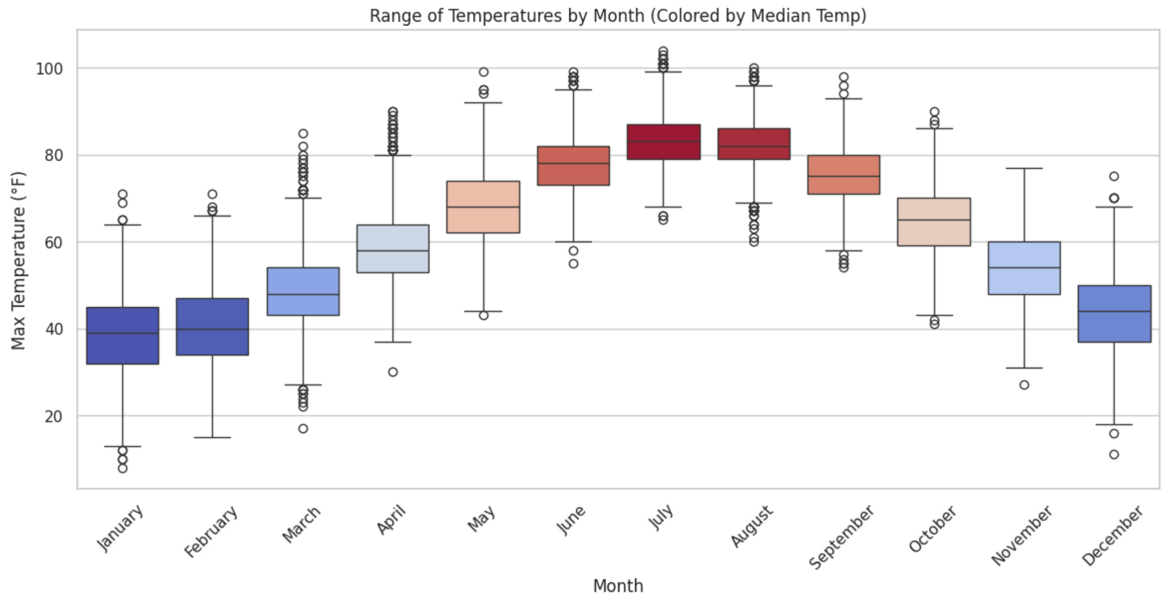
We're using a JFK Airport Weather Data Set, which has been gathered by the NOAA (U.S. National Oceanic and Atmospheric Administration) for decades now.

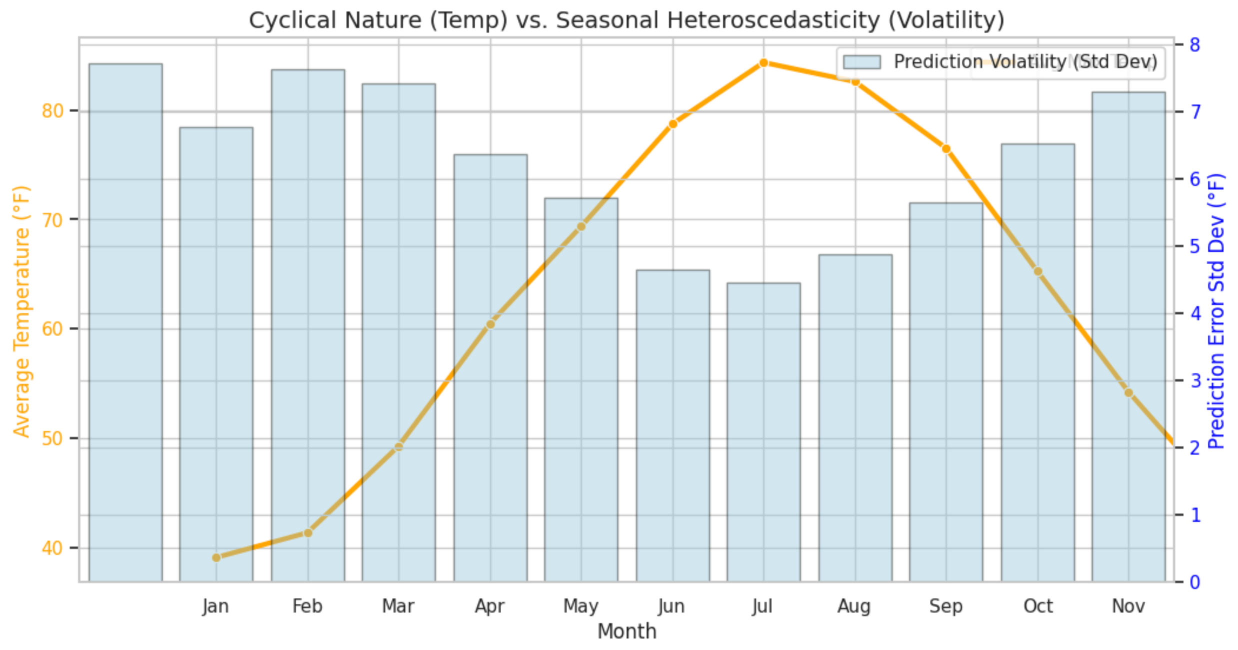
Volume: Even though the dataset is of reasonable size for statistical learning, it's a bit small for the state-of-the-art Deep Learning models that have been released recently for weather forecasting.

Sparsity: Several key columns (Wind, Sunshine) are missing in ~50% of the historical record. Specific weather event flags (specified as WTxx columns) are extremely sparse, and also require feature engineering.

Initial Visualization







Over the years weather events may have become more volatile and this would affect the model performance and give unrealistic confidence in our predictions. To test if this is the case let's use the Breusch-Pagan test for heteroskedasticity.

Hypothesis:
The variance is changing over time in the dataset.

Test (Breusch-Pagan test for heteroskedasticity):
This test can be run by looking at the residuals of a linear model (OLS) versus input features.

Result:
Lagrange Multiplier statistic: 360.5026 P-value: 9.5860e-76
—> Heteroscedasticity detected! (Variance is changing)
Note:
There is a note in the implementation saying that this test can be biased towards claiming heteroskedasticity when the dataset is sufficiently large, and since this corresponds with our visual overview it most likely picked up on local (seasonal) heteroskedasticity rather than a global trend.

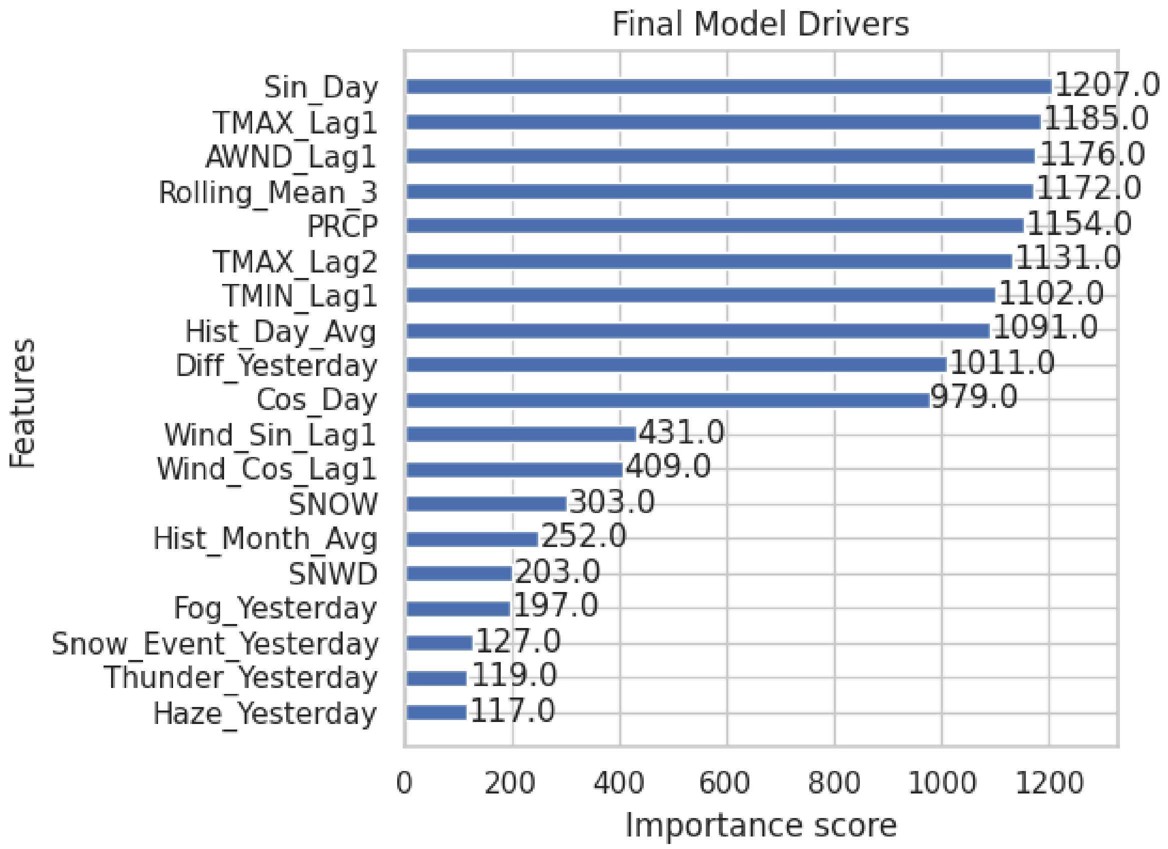
Conclusion:
Based on the volatility findings, a standard regression model would fail to capture the changing risk profile across seasons. A great model choice to model this uncertainty is a quantile regression, more specifically **XGBoost Regression with quantiles (5th, median, 95th)**. This way we can share information about our confidence in our predictions.

Feature Selection

The features that carry the most information (in the context of TMIN/TMAX) are the following:

```
useful_cols = [
    'DATE', 'TMAX', 'TMIN', 'PRCP', 'SNOW', 'SNWD', # Core
    'AWND', 'WDF2',                                  # Wind
    'WT01', 'WT03', 'WT08', 'WT18'                   # Fog, Thunder, Haze, Snow
]
```

These columns were found by looking at the model feature importance plot.



Feature Engineering:

Wind Data:
Wind data is a great predictor of extreme temperatures in New York as they often experience warm southern winds that push up their maximum temperatures on a given day or during a heatwave, conversely strong northern, north-western winds often cool down the climate.

However, this data was not tracked from the beginning of the data set and is only available from the 2000's.

To solve this we can impute this data with **historical backfill**, however we have to note that with this we are creating some impurities in our data (I tested the model and the performance was similar, a tiny bit worse without this data augmentation).

Cyclical Data:
Many features have a cyclical nature, and should be engineered to communicate this information to our model. Most obviously dates and angles; a common way to alter these features is through sinusoidal transformations.

Rolling Means:
Incorporating rolling means are great in smoothing our sharply changing data. I have added the 3-day-rolling-mean for the temperatures as features to the data.

Averages:
Historical averages give us the power to formulate a prediction function that is "forgiving to the user". As well as being a (very) strong correlate for actual values.

Formatting Data for Training

Lags in Time-Series Data:
Making predictions for today with today's data would not be too difficult, but it wouldn't be too rewarding either. Therefore, to make proper predictions we need to shift the rows by one, so our model learns to predict tomorrow's weather based on past weather.

Splitting Data:
We have to be careful when splitting time series data for training and testing as the data should not be shuffled in contrast with non temporal data. I opted for an 80/20 training/test split.

Model Hyperparameter Tuning

In order to get the most out of our xgboost regression model we should tune its hyperparameters. A great library for this is Optuna, a bayesian optimization framework for (hyper)parameter selection.

```
param = {
    'n_estimators': trial.suggest_int('n_estimators', 400, 1000),
    'max_depth': trial.suggest_int('max_depth', 3, 8),
    'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.15, log=True),
    'subsample': trial.suggest_float('subsample', 0.6, 1.0),
    'colsample_bytree': trial.suggest_float('colsample_bytree', 0.6, 1.0),
    'reg_alpha': trial.suggest_float('reg_alpha', 0, 10),
    'reg_lambda': trial.suggest_float('reg_lambda', 0, 10),
    'objective': 'reg:absoluteerror',
    'n_jobs': -1,
    'random_state': 42
 }
```

Since we're developing a separate model for TMIN and TMAX we have to run this twice. The optimal
hyperparameters found for TMIN and TMAX respectively are the following:

best_params_tmin = {'n_estimators': 838, 'max_depth': 5, 'learning_rate': 0.02782095539000955, 'subsample': 0.6006492450379086, 'colsample_bytree': 0.8567563295032506, 'reg_alpha': 6.196478079243171, 'reg_lambda': 3.249283925555016}

best_params_tmax = {'n_estimators': 1000, 'max_depth': 4, 'learning_rate': 0.03184079463622287, 'subsample': 0.7093676333469952, 'colsample_bytree': 0.615069702494021, 'reg_alpha': 8.497485797807963, 'reg_lambda': 8.969796234392428}

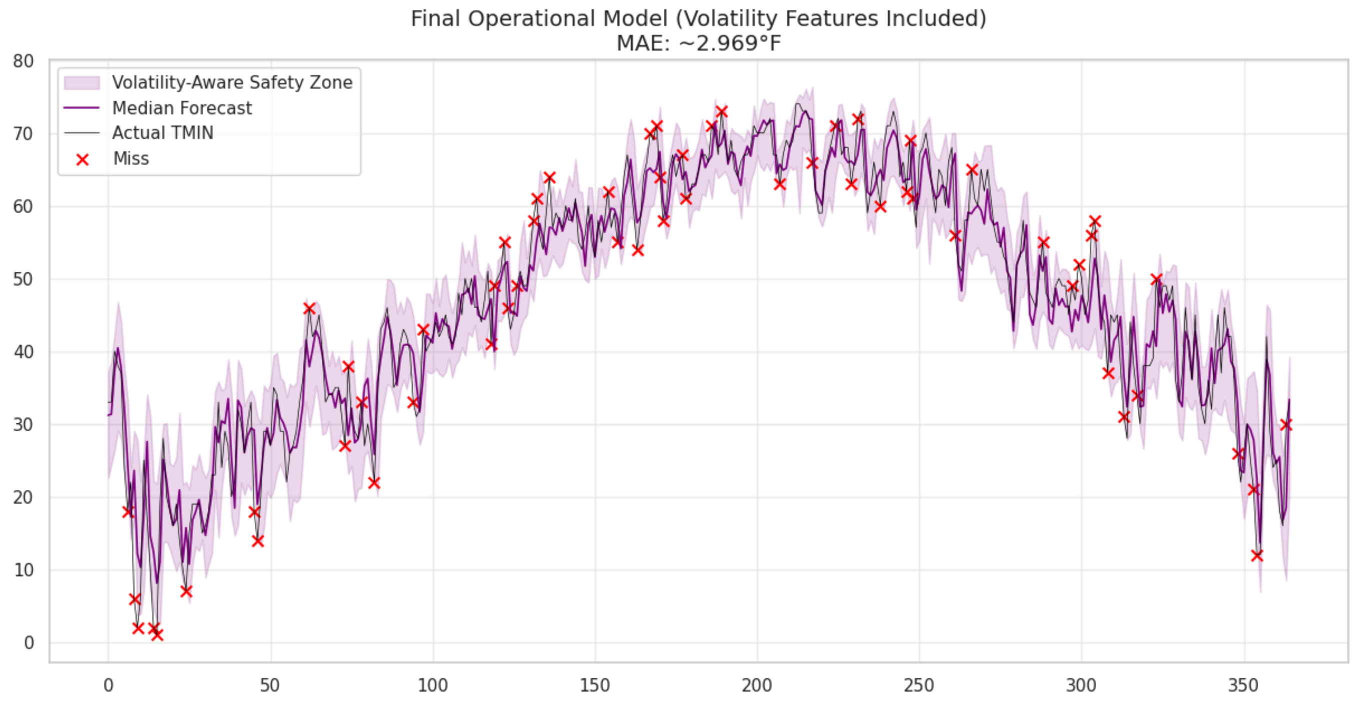
Model Training

To get the actual quantiles we have to train 3 models for each task (TMIN/TMAX) and each (chosen) quantile.

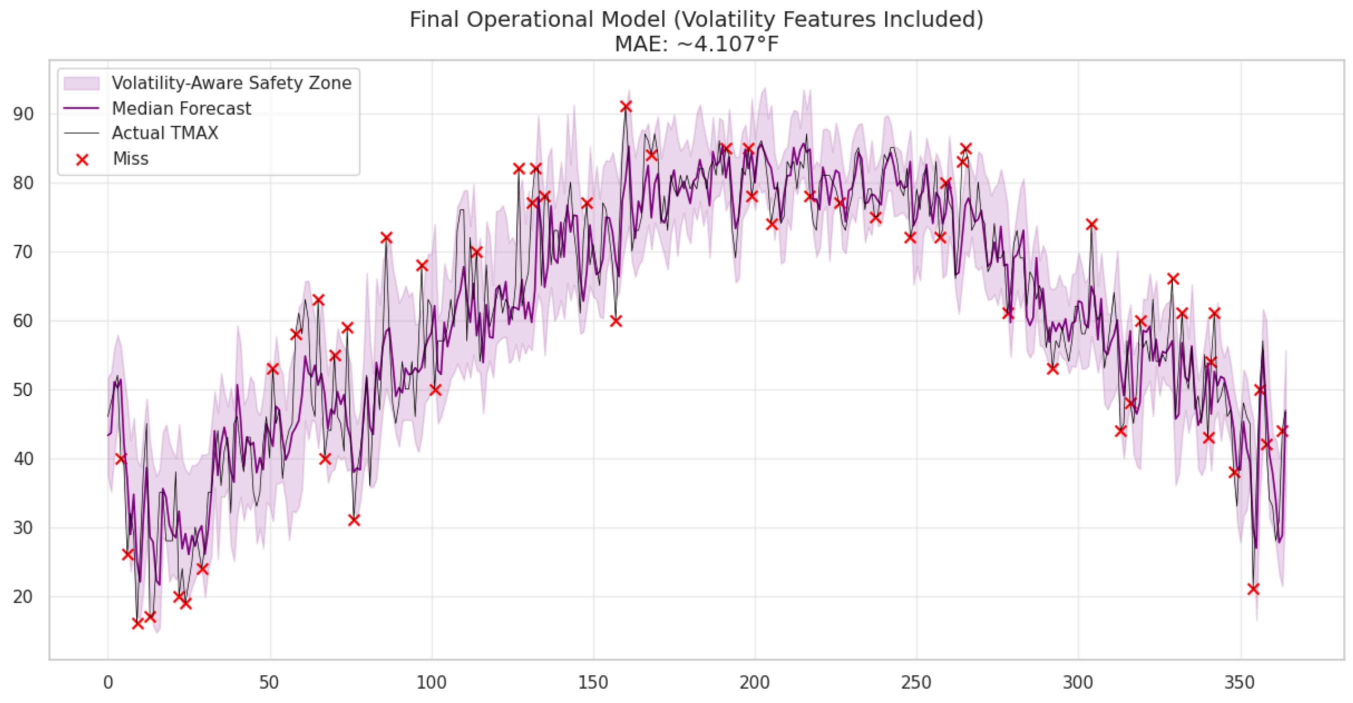
```
m_low_min  = xgb.XGBRegressor(**q_params, quantile_alpha=0.05).fit(X_train_min,y_train_min)
m_mid_min  = xgb.XGBRegressor(**q_params, quantile_alpha=0.50).fit(X_train_min,y_train_min)
m_high_min = xgb.XGBRegressor(**q_params, quantile_alpha=0.95).fit(X_train_min,y_train_min)
```

```
m_low_max  = xgb.XGBRegressor(**q_params, quantile_alpha=0.05).fit(X_train_max,y_train_max)
m_mid_max  = xgb.XGBRegressor(**q_params, quantile_alpha=0.50).fit(X_train_max,y_train_max)
m_high_max = xgb.XGBRegressor(**q_params, quantile_alpha=0.95).fit(X_train_max,y_train_max)
```

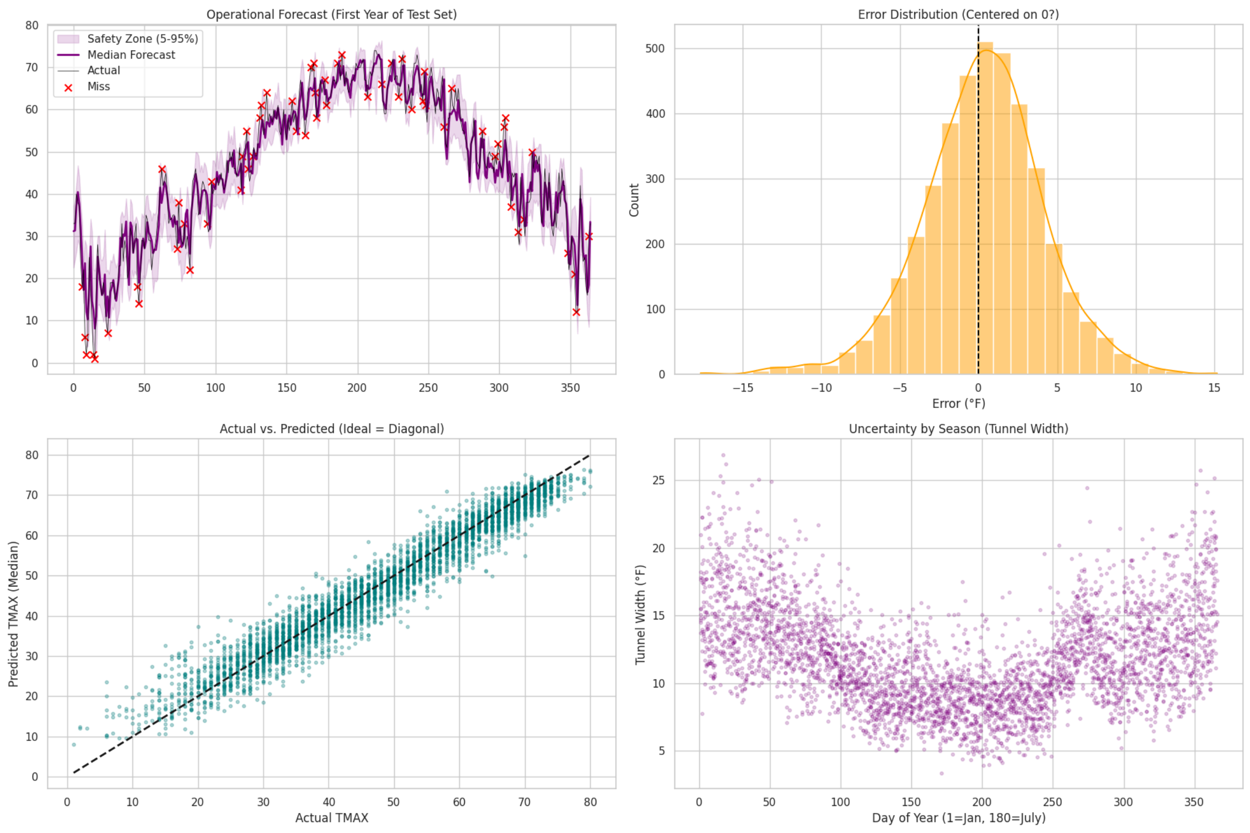
Evaluating Performance



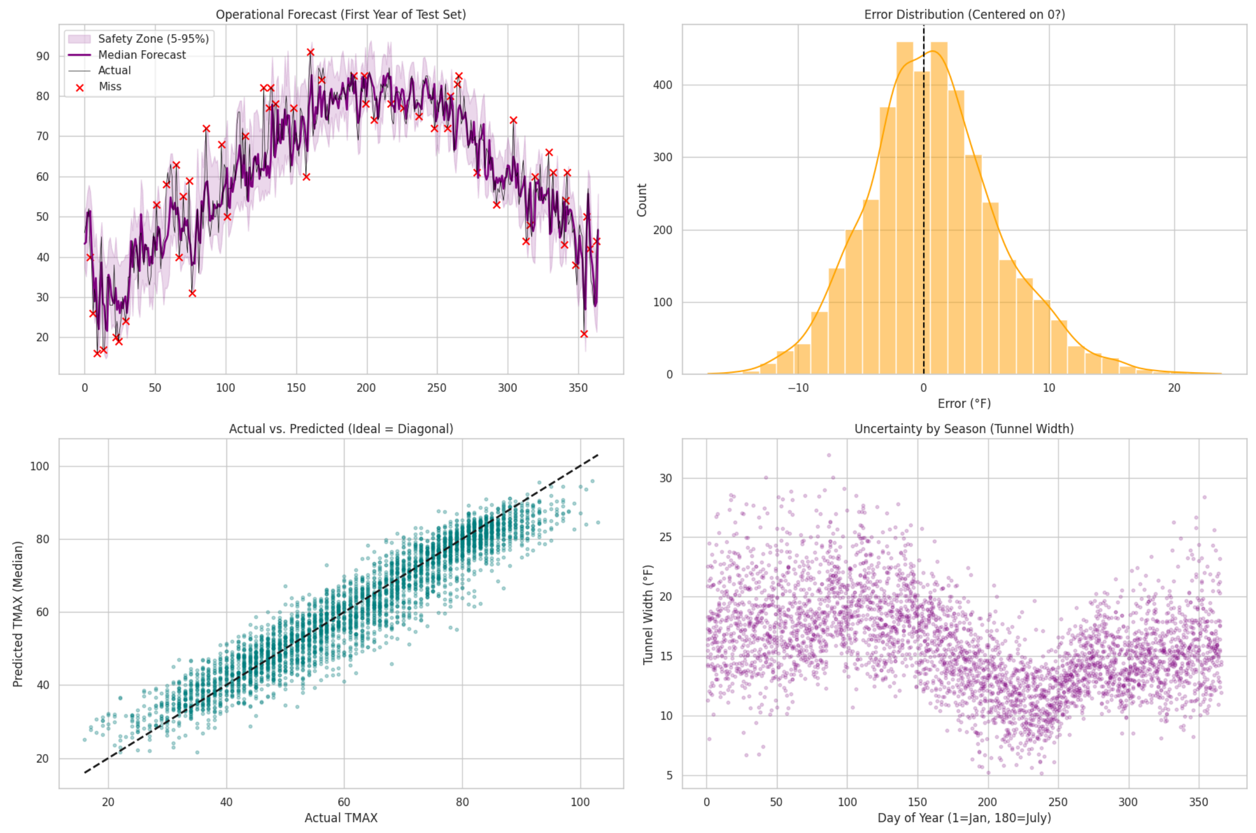
Model(TMIN) predictions visualized over the first 365 days in the test data.



Model(TMAX) predictions visualized over the first 365 days in the test data.



```
==================================================
FINAL MODEL REPORT CARD
==================================================
Median MAE:       2.9928 °F  (Typical Error)
Root Mean Sq:     3.9064 °F  (Penalizes large errors)
R² Score:         0.9439      (Variance explained)
--------------------------------------------------
Tunnel Coverage:  85.64%     (Target: 90.00%)
Avg Tunnel Width: 11.78 °F (Avg Uncertainty)
==================================================
```



```
==================================================
FINAL MODEL REPORT CARD
==================================================
Median MAE:       4.1188 °F  (Typical Error)
Root Mean Sq:     5.3438 °F  (Penalizes large errors)
R² Score:         0.9064      (Variance explained)
--------------------------------------------------
Tunnel Coverage:  85.59%     (Target: 90.00%)
Avg Tunnel Width: 15.91 °F (Avg Uncertainty)
==================================================
```

Interpretation
The model captures the data reasonably well, however there is still room for improvement. Probably more in depth feature engineering could push down the mean absolute error a couple tenths of a degree for each model respectively.

As expected the predictions for minimum temperatures are more accurate than the ones for maximals as the low temperatures are more bounded than highs as sudden warmups can drastically change the daily max, while keeping the daily min relatively stable.

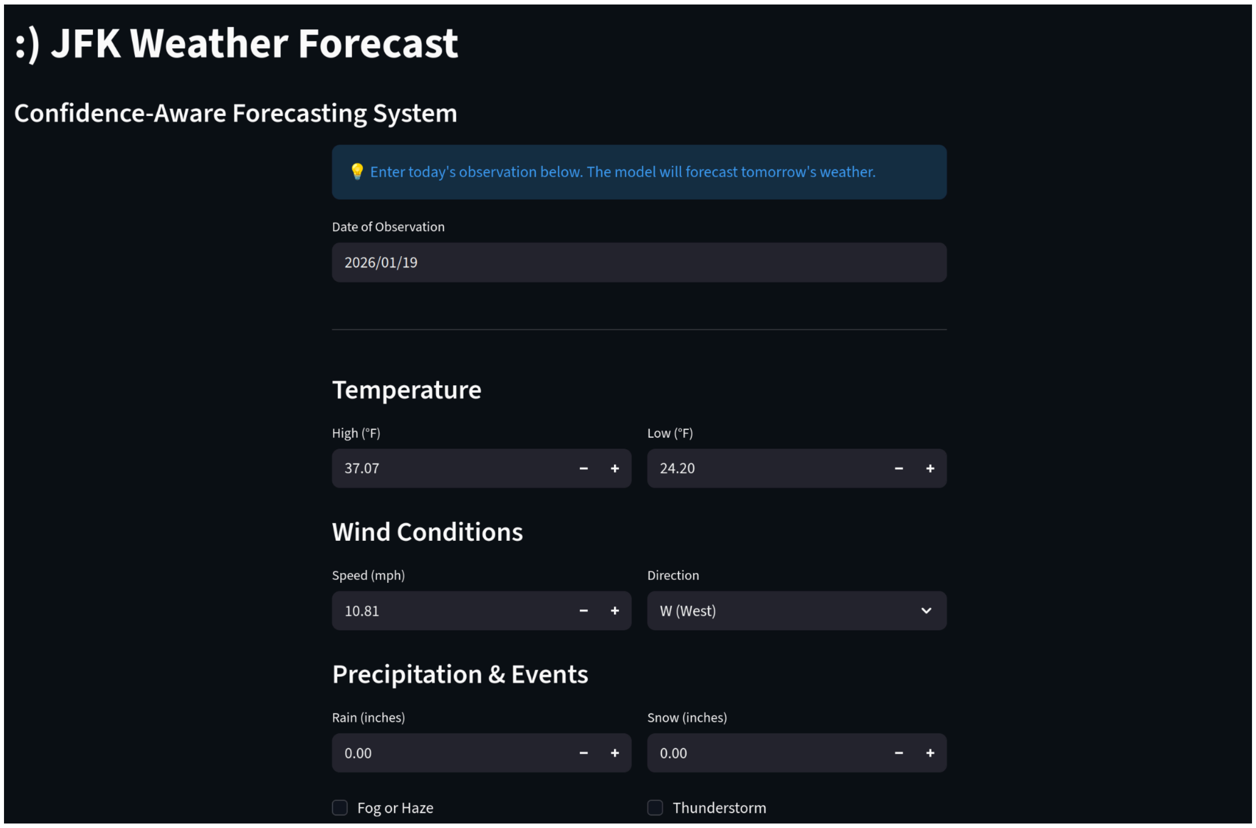
The model biases are very nicely centered around 0, so we're not systematically making errors in either direction.

Formulating Prediction Function

Putting the models together in order to form predictions based on some data is the last step. To make this more visually appealing I opted for a simple streamlit GUI that is connected to our models and our historical New York weather data.

I adapted our data parsing process to allow for users to input partial information and the rest will be augmented by the historical averages for the missing features.

It can be run by following the instructions specified in the github:
https://github.com/A-lamo/EUrail-application/tree/main



:) JFK Weather Forecast

Confidence-Aware Forecasting System

💡 Enter today's observation below. The model will forecast tomorrow's weather.

Date of Observation: 2026/01/19

Temperature

High (°F): 37.07

Low (°F): 24.20

Wind Conditions

Speed (mph): 10.81

Direction: W (West)

Precipitation & Events

Rain (inches): 0.00

Snow (inches): 0.00

☐ Fog or Haze

☐ Thunderstorm