NYC Taxi Demand Forecasting

Technical Analysis & Model Evaluation

DATASET SPECIFICATIONS

Temporal Coverage:
• Start: 2014-07-01 00:00
• End: 2015-01-31 23:30
• Duration: 214 days
• Total Observations: 10,320
• Sampling Frequency: 30-minute intervals
• Missing Values: 0 (0.00%)

Statistical Properties:
• Mean: 15137.57 trips/30min
• Median: 16778.00 trips/30min
• Standard Deviation: 6939.50
• Coefficient of Variation: 0.458
• Skewness: -0.452
• Kurtosis: -0.780

TARGET MODELS TECHNICAL OVERVIEW

□ Naive Forecasting
Algorithm: $y_{t+1} = y_t$
Complexity: O(1)
Memory: O(1)
Parameters: 0

□ SARIMA (Seasonal ARIMA)
Algorithm: $(1-\phi L)(1-\Phi L^s)(1-L)^d(1-L^s)^D y_t = (1+\theta L)(1+\Theta L^s)\varepsilon_t$
Complexity: O(n)
Memory: O(max(p,q,P,Q))
Parameters: p+d+q+P+D+Q = 6

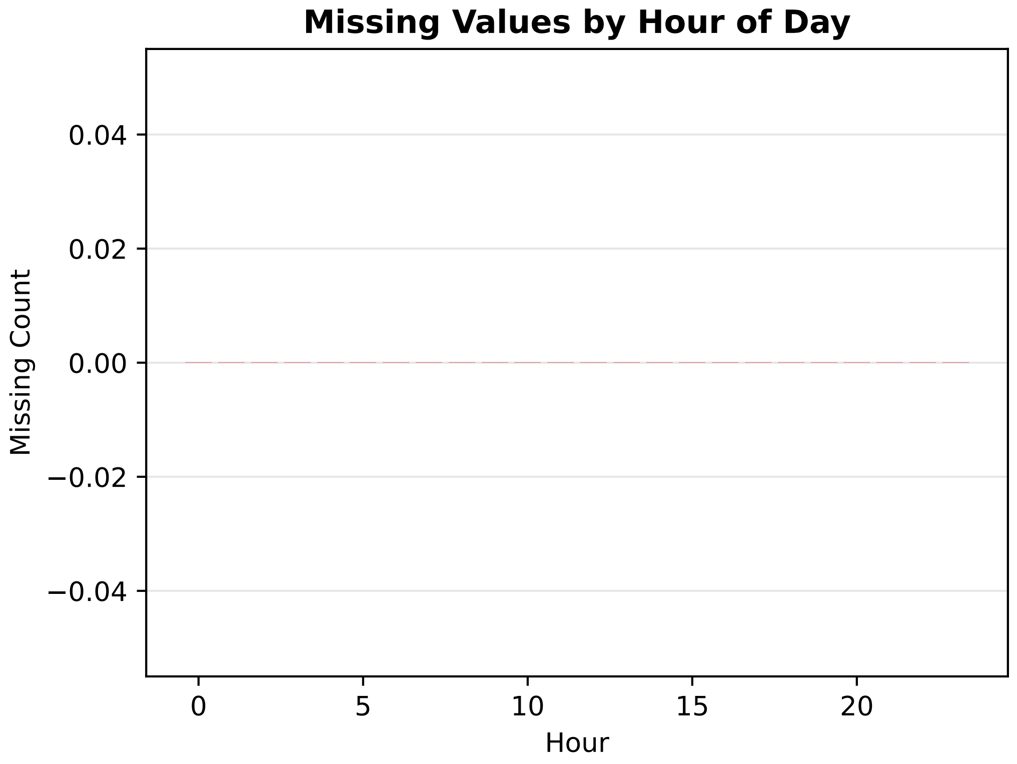
□ Random Forest
Algorithm: Ensemble of Decision Trees with Bootstrap Aggregating
Complexity: O(n_trees × n_features × log(n_samples))
Memory: O(n_trees × tree_depth)
Parameters: ~100-500 per tree

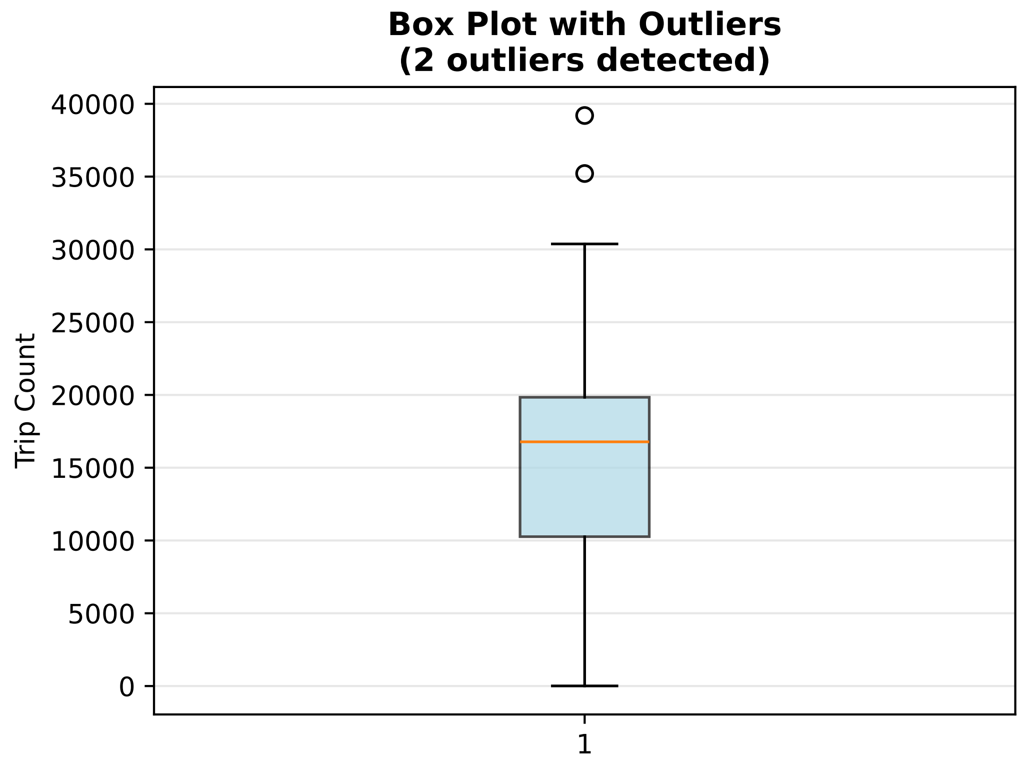
□ LSTM Neural Network
Algorithm: $\sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ → Cell State Updates
Complexity: O(sequence_length × hidden_units^2)
Memory: O(hidden_units × layers)
Parameters: 4×(hidden_units^2 + hidden_units×input_dim)

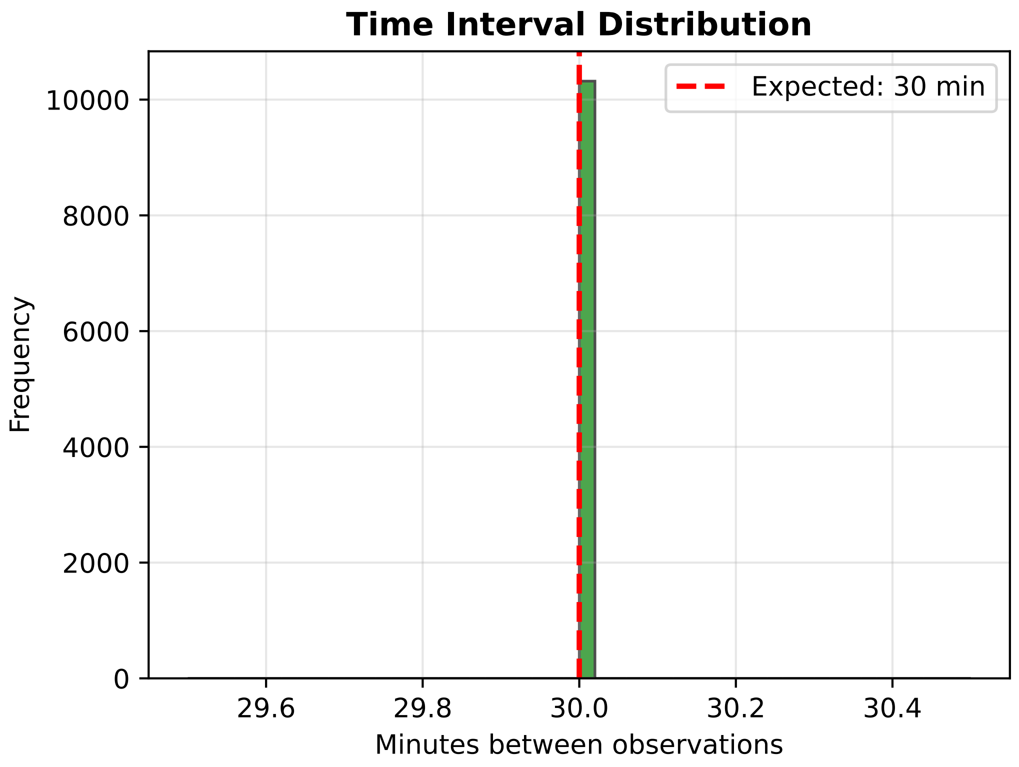
EVALUATION METHODOLOGY

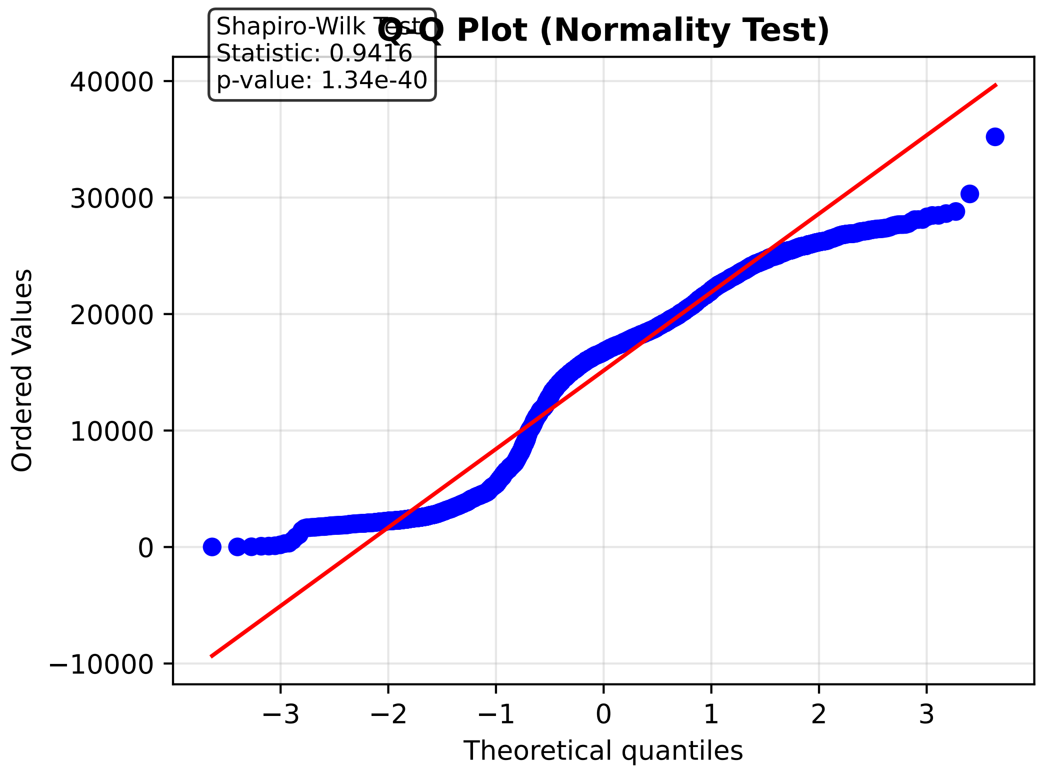
Statistical Metrics:
• Mean Absolute Error (MAE)
• Root Mean Square Error (RMSE)
• Mean Absolute Percentage Error (MAPE)
• R-squared (R^2)
• Directional Accuracy

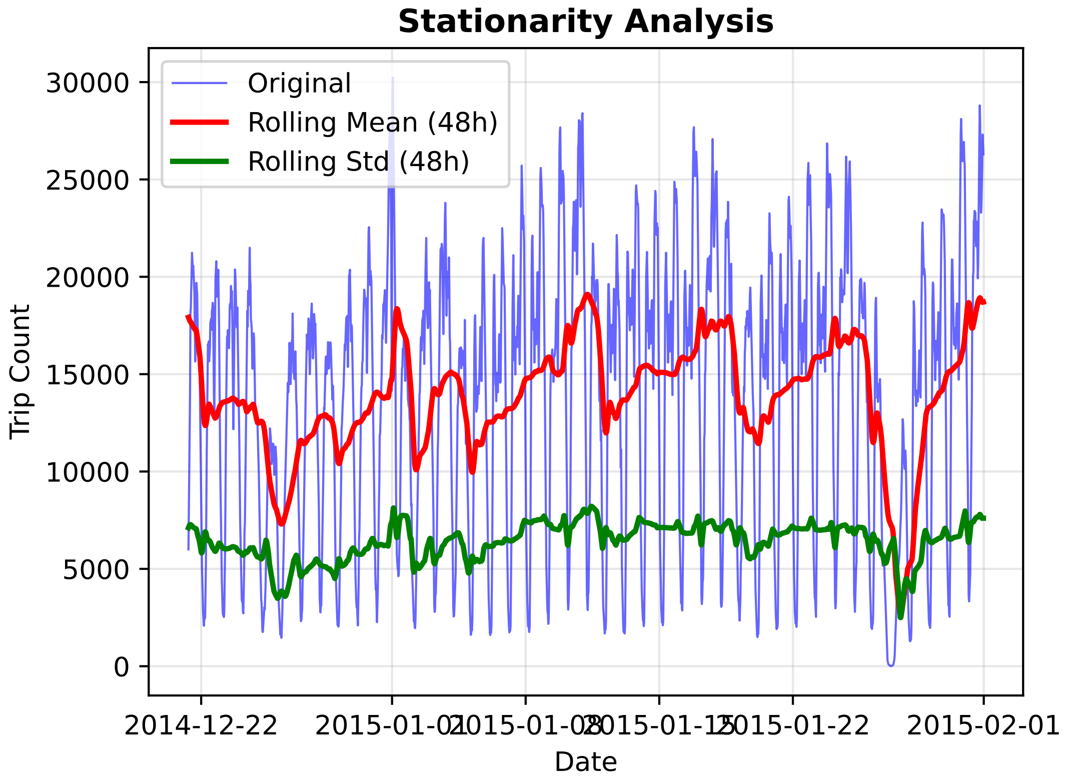
Cross-Validation:
• Time Series Split Validation
• Walk-Forward Validation
• Blocked Cross-Validation
• Rolling Window Validation


Missing Values by Hour of Day
Missing Count
0.04
0.02
0.00
−0.02
−0.04
0
5
10
15
20
Hour


Box Plot with Outliers
(2 outliers detected)
Trip Count
40000
35000
30000
25000
20000
15000
10000
5000
0
1

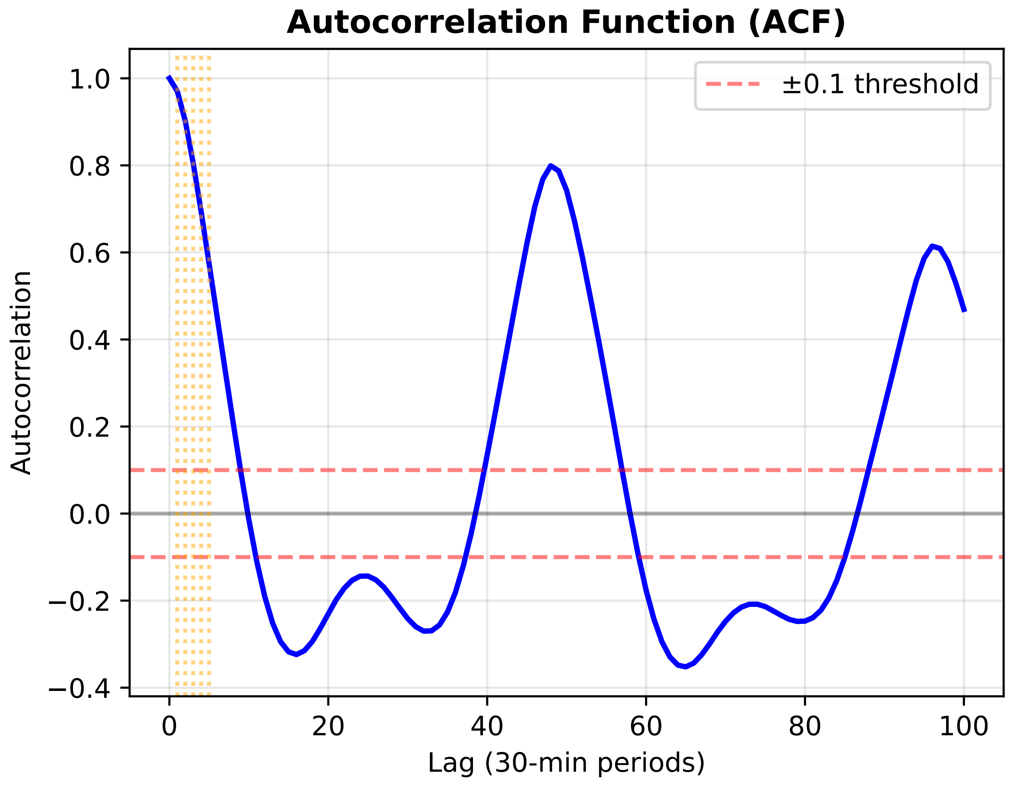

Time Interval Distribution
Expected: 30 min
Frequency
10000
8000
6000
4000
2000
0
29.6
29.8
30.0
30.2
30.4
Minutes between observations

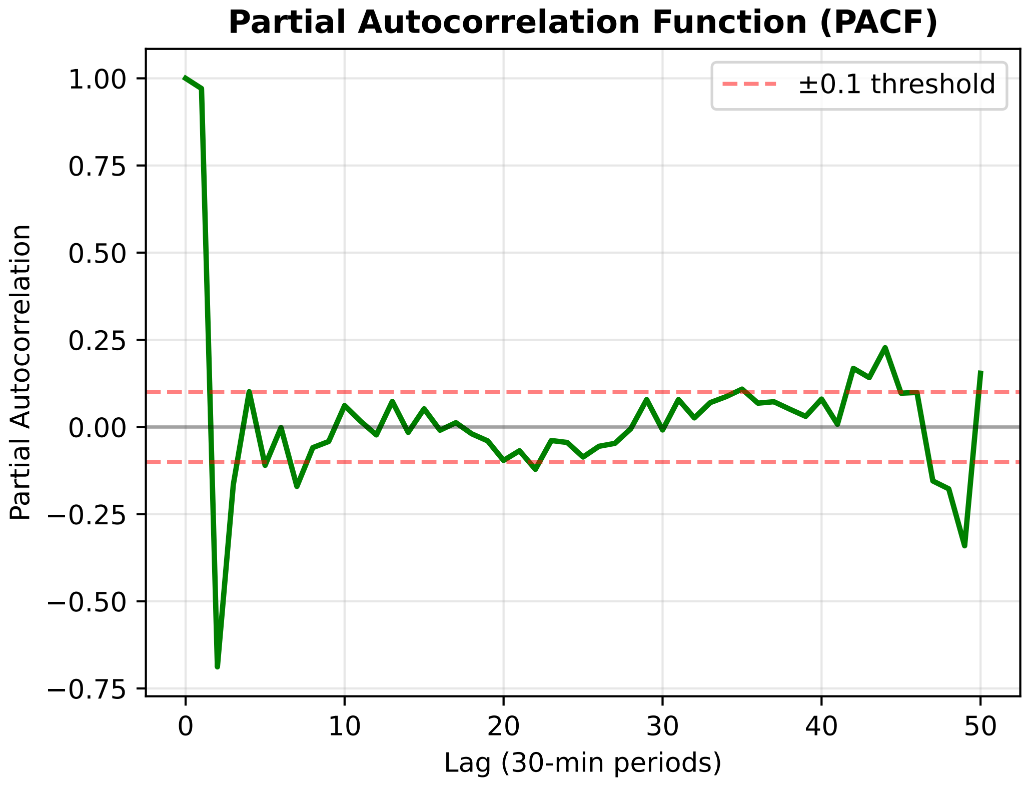

Q-Q Plot (Normality Test)
Shapiro-Wilk Test
Statistic: 0.9416
p-value: 1.34e-40
Ordered Values
40000
30000
20000
10000
0
−10000
−3
−2
−1
0
1
2
3
Theoretical quantiles

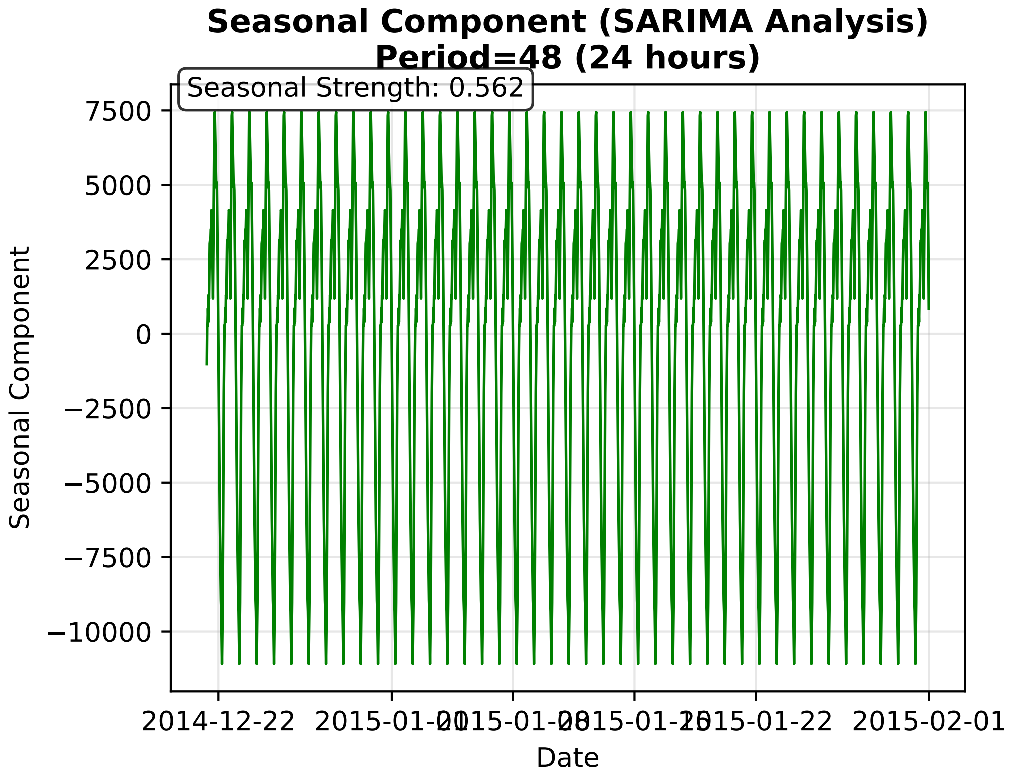


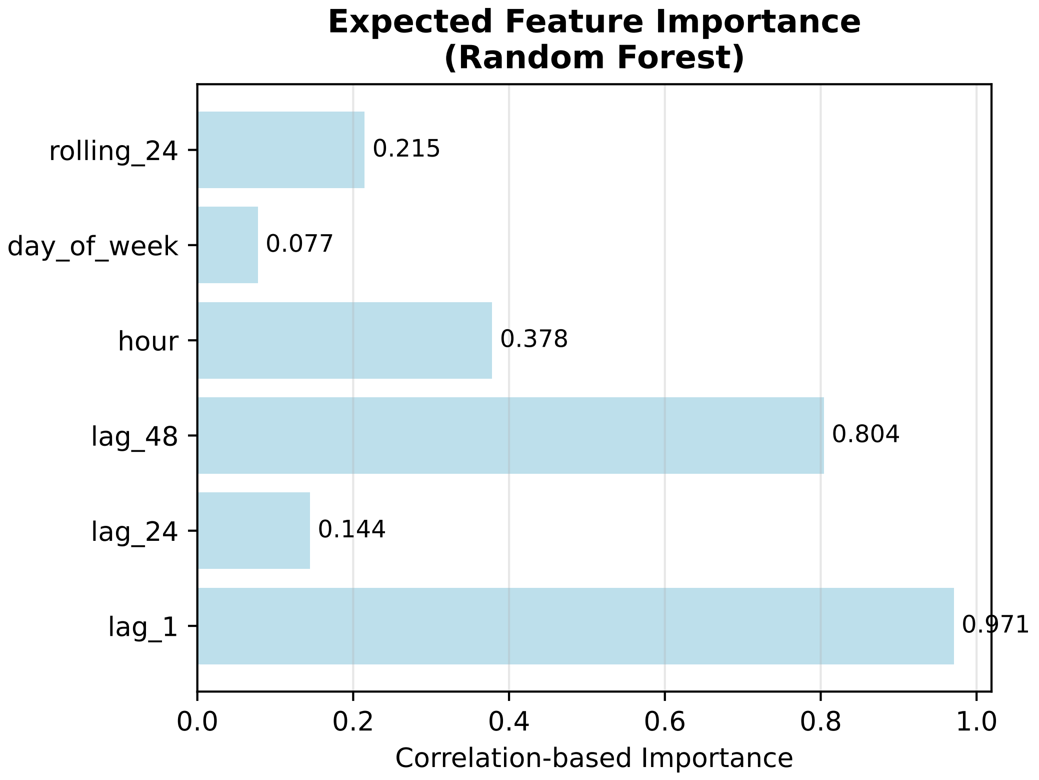
Stationarity Test Results

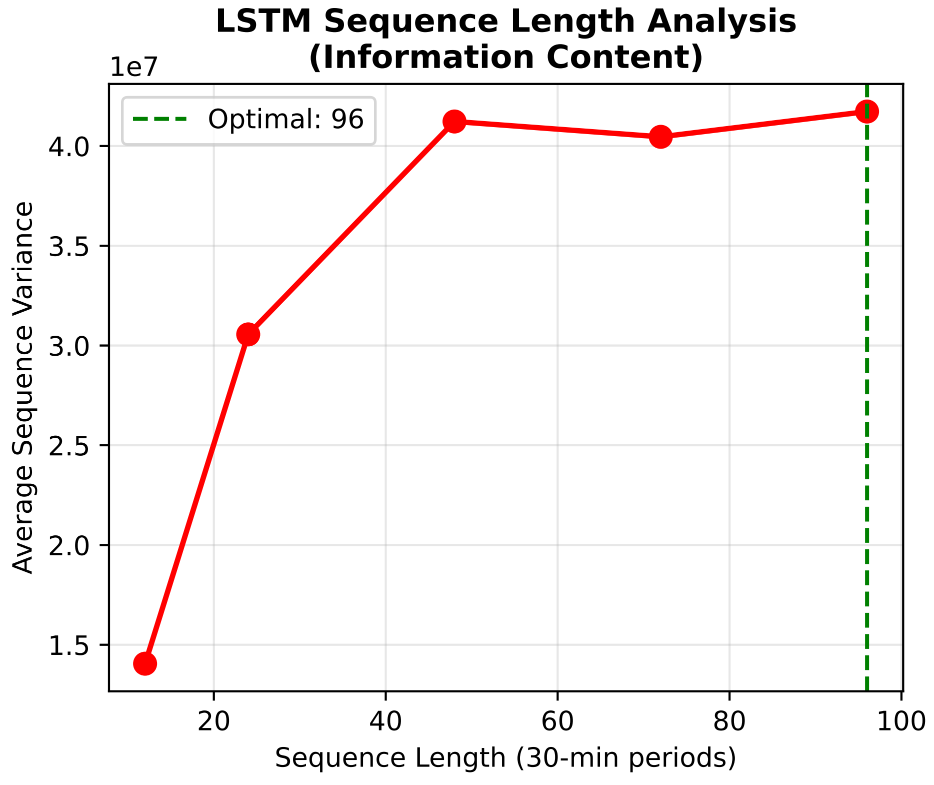
```
ADF Stationarity Test:
Test Statistic: -7.5666
p-value: 0.0000
Critical Values:
  1%: -3.4310
  5%: -2.8618
  10%: -2.5669

Interpretation:
Stationary
SARIMA d parameter: 0
```

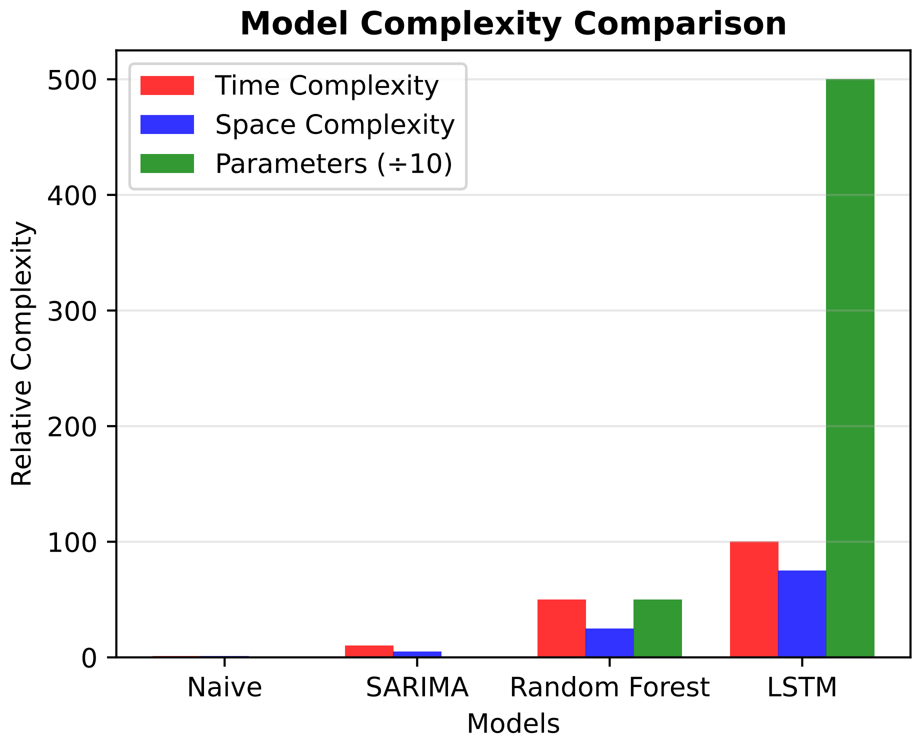


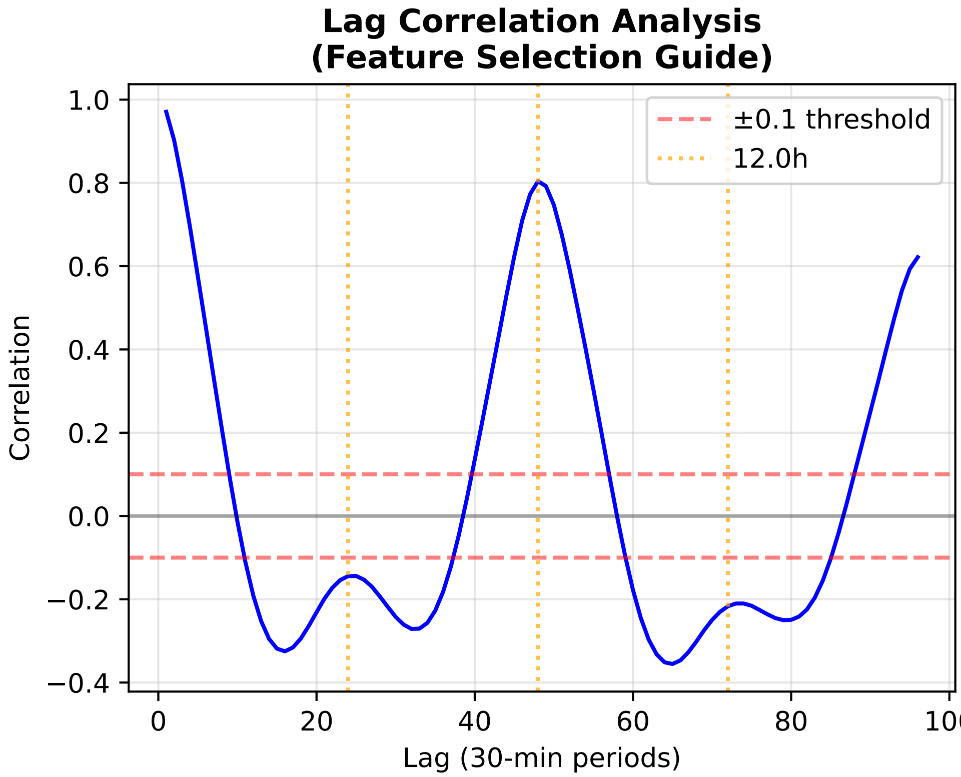


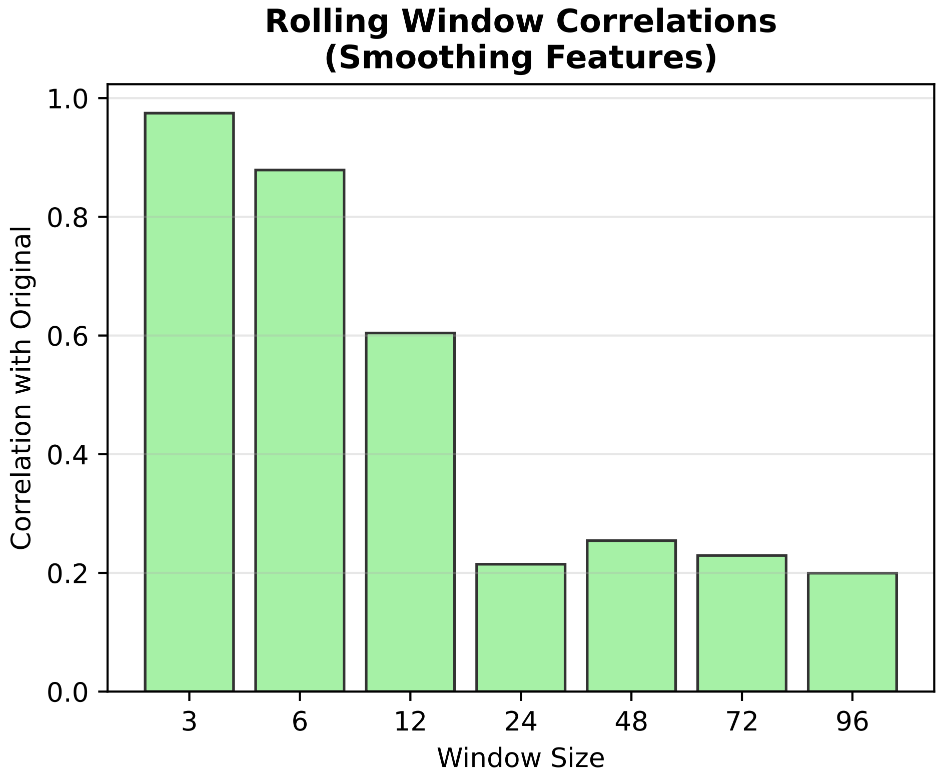

Seasonal Component (SARIMA Analysis)
Period=48 (24 hours)
Seasonal Strength: 0.562
Seasonal Component
7500
5000
2500
0
−2500
−5000
−7500
−10000
2014-12-22
2015-01-01
2015-01-08
2015-01-15
2015-01-22
2015-02-01
Date

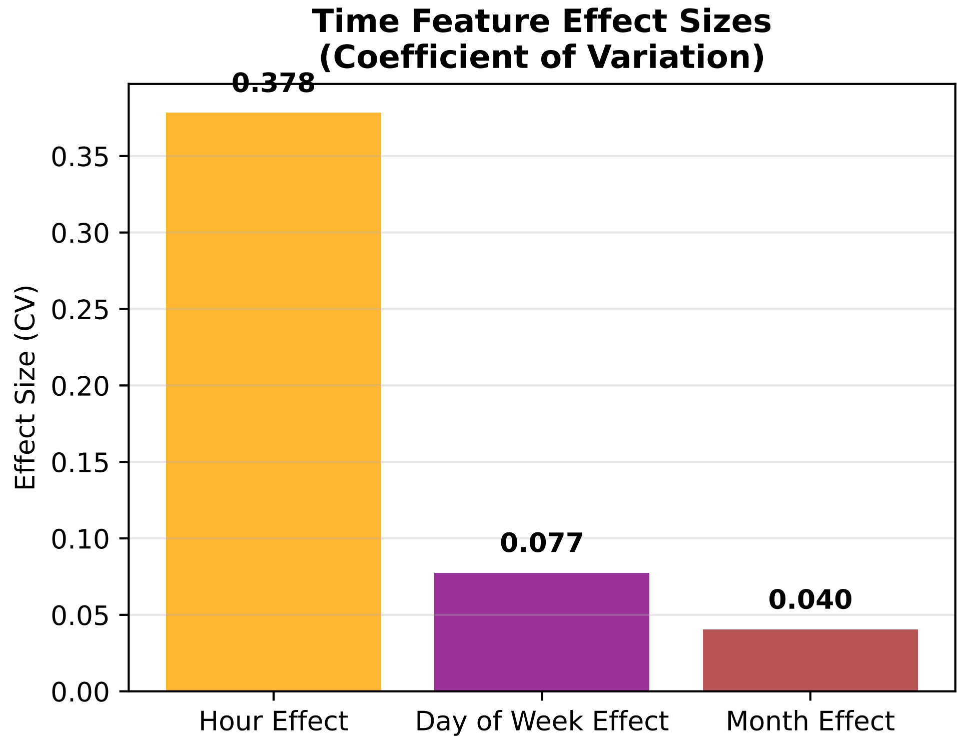

Expected Feature Importance
(Random Forest)
rolling_24
day_of_week
hour
lag_48
lag_24
lag_1
0.215
0.077
0.378
0.804
0.144
0.971
0.0
0.2
0.4
0.6
0.8
1.0
Correlation-based Importance

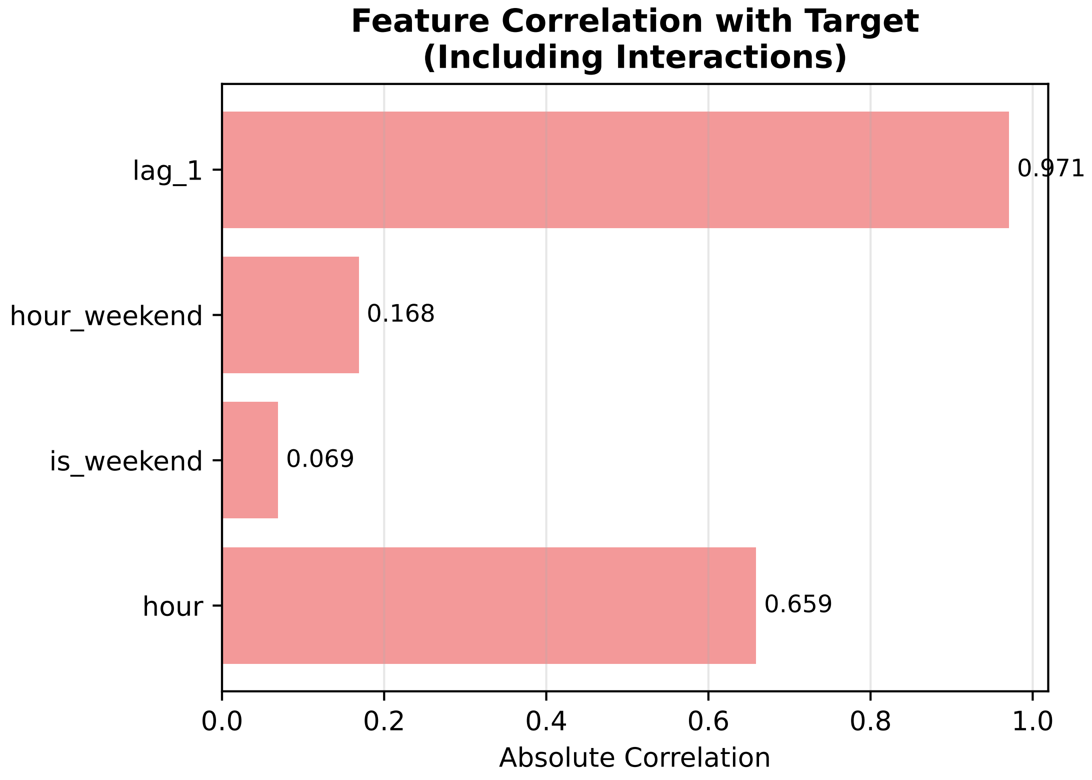

LSTM Sequence Length Analysis
(Information Content)
1e7
Optimal: 96
Average Sequence Variance
1.5
2.0
2.5
3.0
3.5
4.0
20
40
60
80
100
Sequence Length (30-min periods)

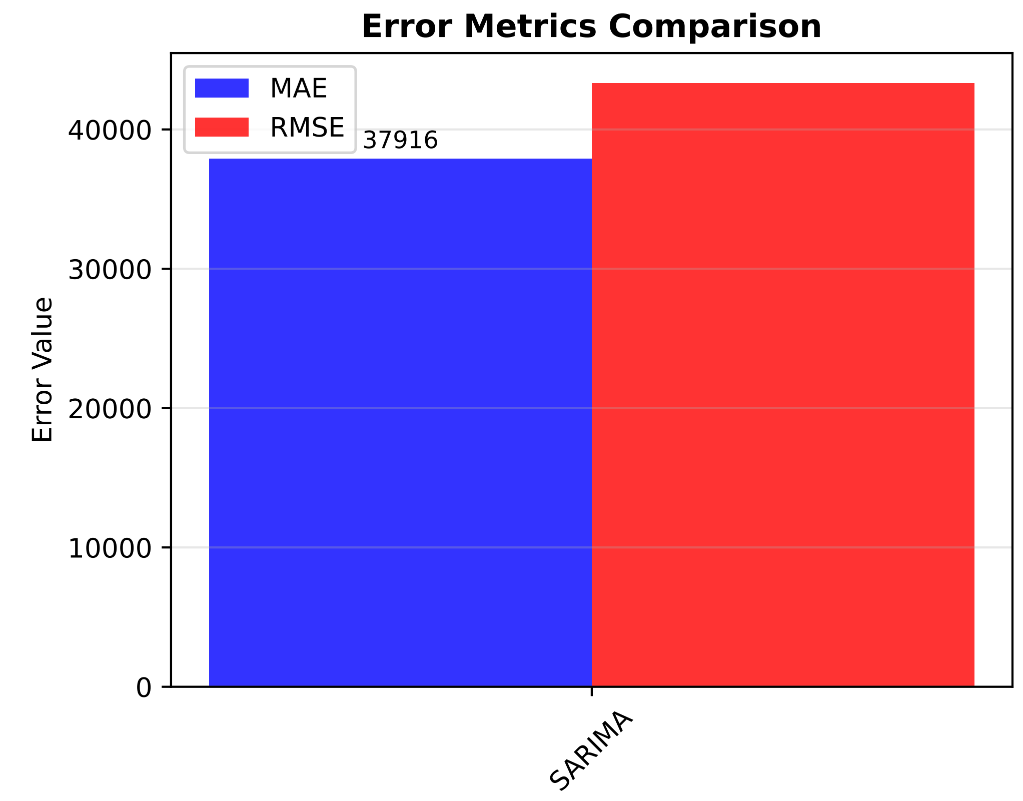

Model Complexity Comparison
Time Complexity
Space Complexity
Parameters (÷10)
Relative Complexity
0
100
200
300
400
500
Naive
SARIMA
Random Forest
LSTM
Models

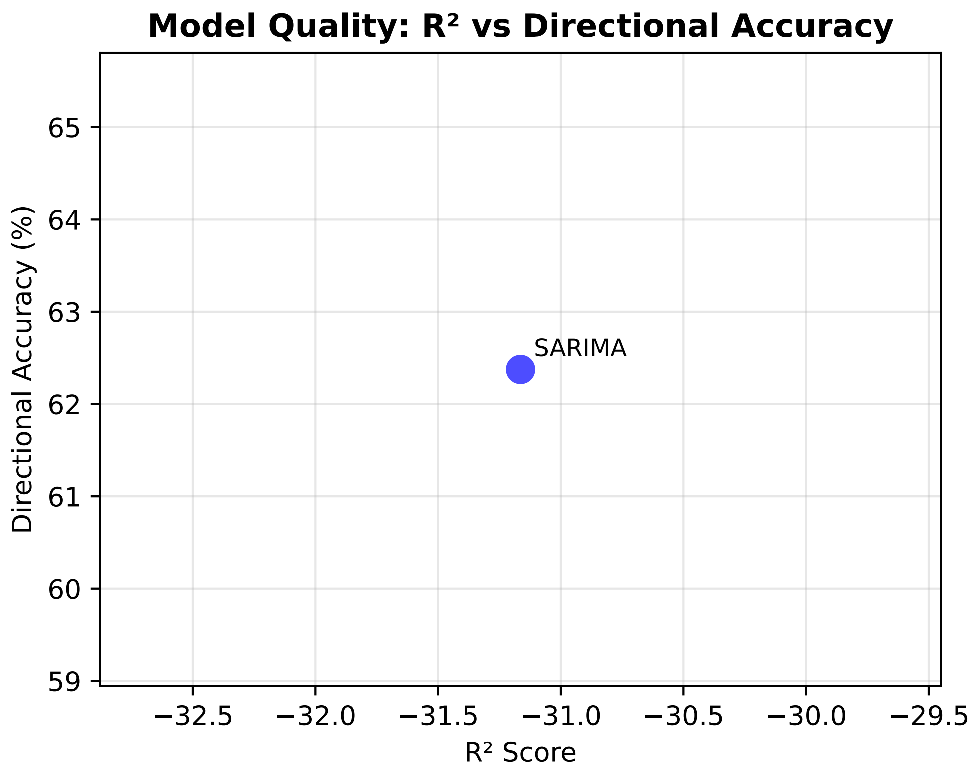

Lag Correlation Analysis
(Feature Selection Guide)
±0.1 threshold
12.0h
Correlation
Lag (30-min periods)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
20
40
60
80
100

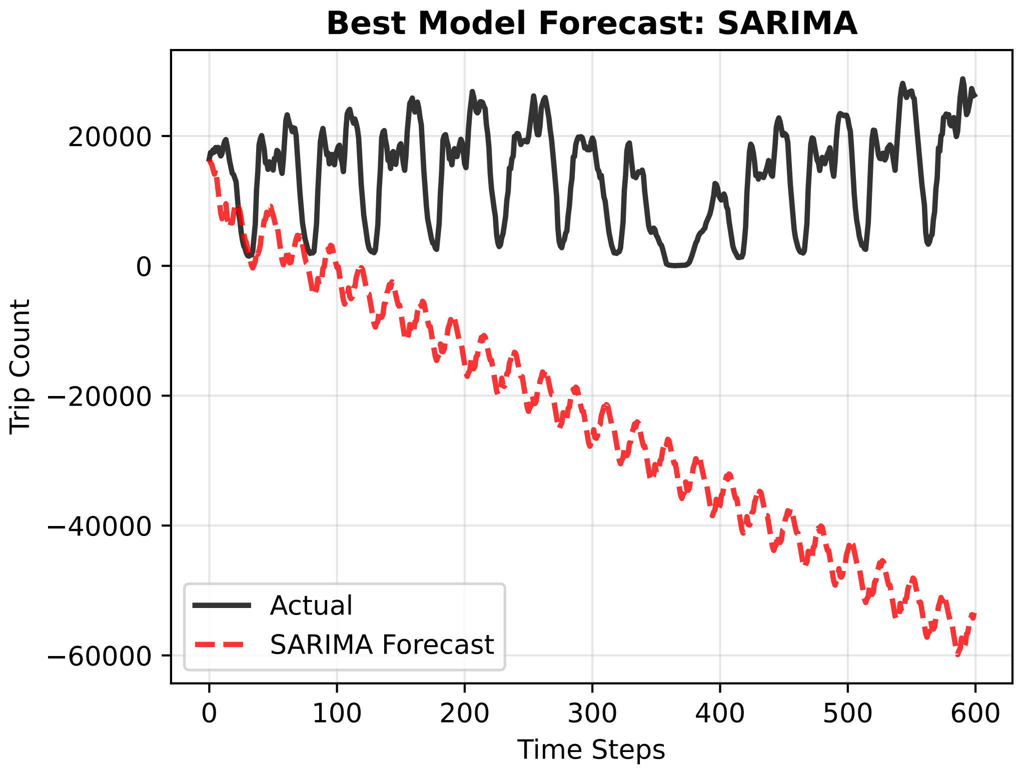

Rolling Window Correlations
(Smoothing Features)
Correlation with Original
Window Size
1.0
0.8
0.6
0.4
0.2
0.0
3
6
12
24
48
72
96

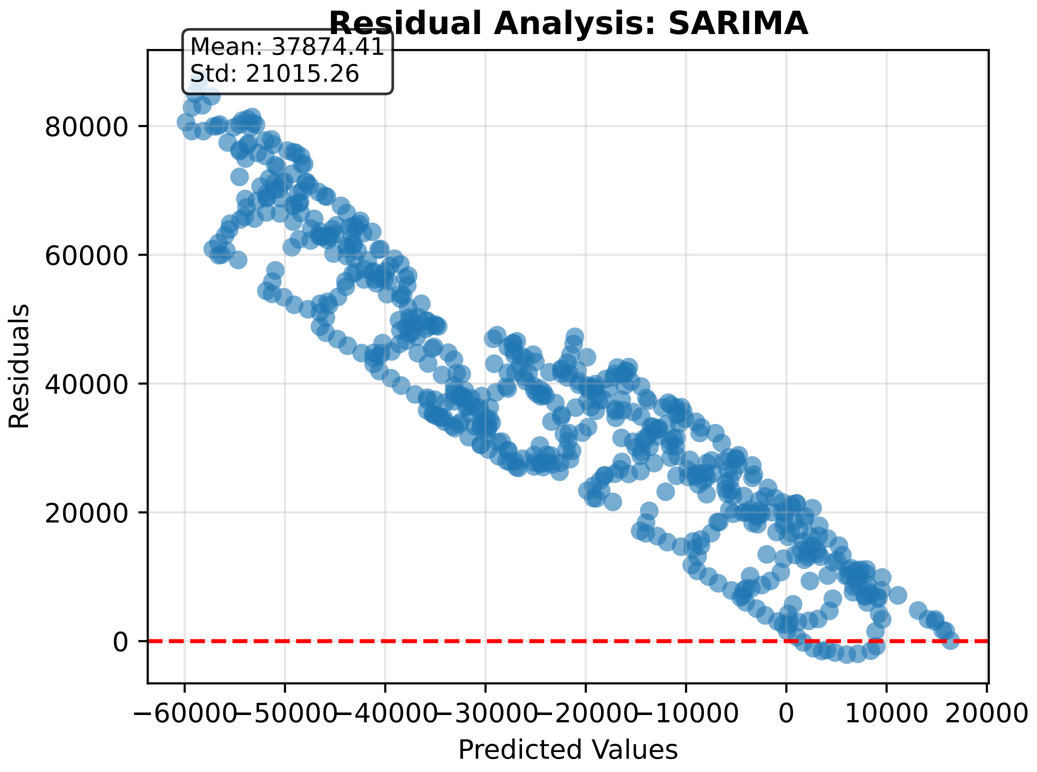

Time Feature Effect Sizes
(Coefficient of Variation)
Effect Size (CV)
0.378
0.077
0.040
Hour Effect
Day of Week Effect
Month Effect
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00


Feature Correlation with Target
(Including Interactions)
lag_1
0.971
hour_weekend
0.168
is_weekend
0.069
hour
0.659
Absolute Correlation
0.0
0.2
0.4
0.6
0.8
1.0


Error Metrics Comparison
MAE
RMSE
37916
Error Value
0
10000
20000
30000
40000
SARIMA


Model Quality: R² vs Directional Accuracy
Directional Accuracy (%)
59
60
61
62
63
64
65
R² Score
−32.5
−32.0
−31.5
−31.0
−30.5
−30.0
−29.5
SARIMA


Best Model Forecast: SARIMA
Trip Count
Time Steps
Actual
SARIMA Forecast
20000
0
−20000
−40000
−60000
0
100
200
300
400
500
600


Residual Analysis: SARIMA
Mean: 37874.41
Std: 21015.26
Residuals
Predicted Values
0
20000
40000
60000
80000
−60000
−50000
−40000
−30000
−20000
−10000
0
10000
20000

Technical Implementation Details

MODEL IMPLEMENTATION SPECIFICATIONS

NAIVE FORECASTING

Algorithm Implementation:
```
def naive_forecast(data, steps):
    return np.full(steps, data.iloc[-1])
```

Computational Complexity:
- Time: O(1) - constant time
- Space: O(1) - constant space
- Parameters: 0

Production Requirements:
- CPU: Minimal (any modern processor)
- Memory: <1MB
- Storage: Historical data only
- Latency: <1ms

Advantages:
- Zero training time
- Perfect interpretability
- No hyperparameter tuning
- Robust to data quality issues

Limitations:
- Poor performance in volatile periods
- No pattern recognition
- No seasonality handling

SARIMA MODELING

Algorithm Implementation:
```
SARIMAX(endog, order=(p,d,q), seasonal_order=(P,D,Q,s))
```
- p: AR order (1)
- d: Differencing order (1)
- q: MA order (1)
- P: Seasonal AR order (1)
- D: Seasonal differencing (1)
- Q: Seasonal MA order (1)
- s: Seasonal period (48)

Mathematical Foundation:
```
(1-φL)(1-ΦL^48)(1-L)(1-L^48)y_t = (1+θL)(1+ΘL^48)ε_t
```

Computational Complexity:
- Time: O(n × max(p,q,P,Q)) for fitting
- Space: O(max(p,q,P,Q) + s)
- Parameters: 6 (φ,θ,Φ,Θ,σ²,intercept)

Production Requirements:
- CPU: Moderate (2+ cores recommended)
- Memory: 100-500MB depending on data size
- Storage: Model state + seasonal data
- Training time: 1-5 minutes
- Prediction latency: <100ms

Implementation Details:
- Requires stationarity testing
- Parameter estimation via Maximum Likelihood
- Model diagnostics essential
- Periodic retraining needed

Advantages:
- Strong statistical foundation
- Handles seasonality naturally
- Prediction intervals available
- Interpretable parameters

Limitations:
- Assumes linear relationships
- Sensitive to outliers
- Requires parameter tuning
- May need differencing

RANDOM FOREST

Algorithm Implementation:
```
RandomForestRegressor(
    n_estimators=100,
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    bootstrap=True
)
```

Feature Engineering Pipeline:
```
features = [
    'lag_1', 'lag_2', 'lag_3', 'lag_24', 'lag_48',
    'rolling_mean_3', 'rolling_mean_12', 'rolling_mean_24',
    'hour', 'day_of_week', 'month', 'is_weekend'
]
```

Computational Complexity:
- Training: O(n_trees × n_features × n_samples × log(n_samples))
- Prediction: O(n_trees × log(tree_depth))
- Space: O(n_trees × tree_nodes)
- Parameters: ~1000-5000 per tree

Production Requirements:
- CPU: Multi-core beneficial (4+ cores)
- Memory: 1-5GB for large datasets
- Storage: Model file 10-100MB
- Training time: 5-30 minutes
- Prediction latency: <50ms

Implementation Details:
- Feature preprocessing pipeline critical
- Missing value handling built-in
- Feature importance analysis available
- No assumptions about data distribution

Advantages:
- Handles non-linear relationships
- Feature importance interpretability
- Robust to outliers and missing data
- No hyperparameter sensitivity

Limitations:
- Can overfit with too many features
- Memory intensive for large datasets
- Limited extrapolation capability
- Feature engineering dependency

LSTM NEURAL NETWORK

Architecture Implementation:
```
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(48, 1)),
    Dropout(0.2),
    LSTM(50, return_sequences=False),
    Dropout(0.2),
    Dense(25),
    Dense(1)
])
```

Training Configuration:
- Optimizer: Adam(learning_rate=0.001)
- Loss: Mean Squared Error
- Batch size: 32
- Epochs: 50-100 with early stopping
- Validation split: 20%

Computational Complexity:
- Training: O(seq_len × hidden_units² × epochs)
- Prediction: O(seq_len × hidden_units²)
- Parameters: 4×(hidden_units² + hidden_units×input_dim)
- Memory: O(batch_size × seq_len × hidden_units)

Production Requirements:
- CPU: High-performance (8+ cores) or GPU
- Memory: 2-8GB GPU memory preferred
- Storage: Model file 50-200MB
- Training time: 30-120 minutes
- Prediction latency: <500ms

Implementation Details:
- Data normalization mandatory (MinMaxScaler)
- Sequence windowing required
- Gradient clipping recommended
- Learning rate scheduling beneficial

Advantages:
- Captures complex temporal dependencies
- Handles multivariate inputs naturally
- State-of-the-art sequence modeling
- Flexible architecture

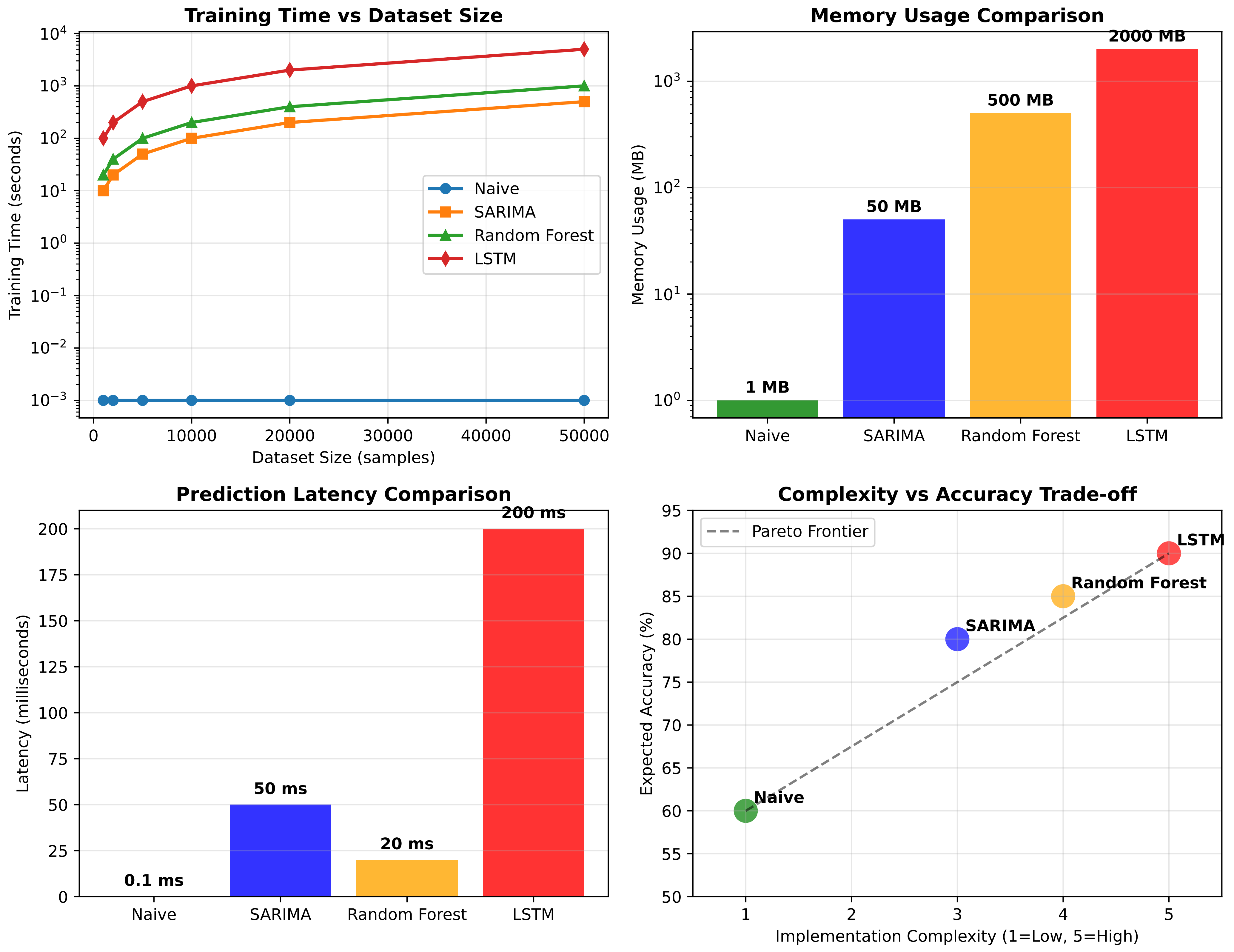
Limitations:
- Computationally intensive
- Requires large datasets
- Hyperparameter sensitivity
- Black-box interpretability

DEPLOYMENT ARCHITECTURE RECOMMENDATIONS

Production Stack:
- Containerization: Docker
- Orchestration: Kubernetes
- API Framework: FastAPI/Flask
- Model Serving: MLflow/TensorFlow Serving
- Monitoring: Prometheus + Grafana
- Data Pipeline: Apache Kafka/Airflow

Scaling Strategy:
- Horizontal scaling for API layer
- Model versioning and A/B testing
- Caching for frequent predictions
- Load balancing across model instances

Monitoring Requirements:
- Prediction accuracy tracking
- Model drift detection
- Performance metrics (latency, throughput)
- Data quality monitoring
- Alert systems for anomalies


Training Time vs Dataset Size
Training Time (seconds)
Dataset Size (samples)
Naive
SARIMA
Random Forest
LSTM
Memory Usage Comparison
Memory Usage (MB)
1 MB
50 MB
500 MB
2000 MB
Prediction Latency Comparison
Latency (milliseconds)
0.1 ms
50 ms
20 ms
200 ms
Complexity vs Accuracy Trade-off
Expected Accuracy (%)
Implementation Complexity (1=Low, 5=High)
Pareto Frontier