6 JUL. 2025

SAMEER, FARES, ALEX

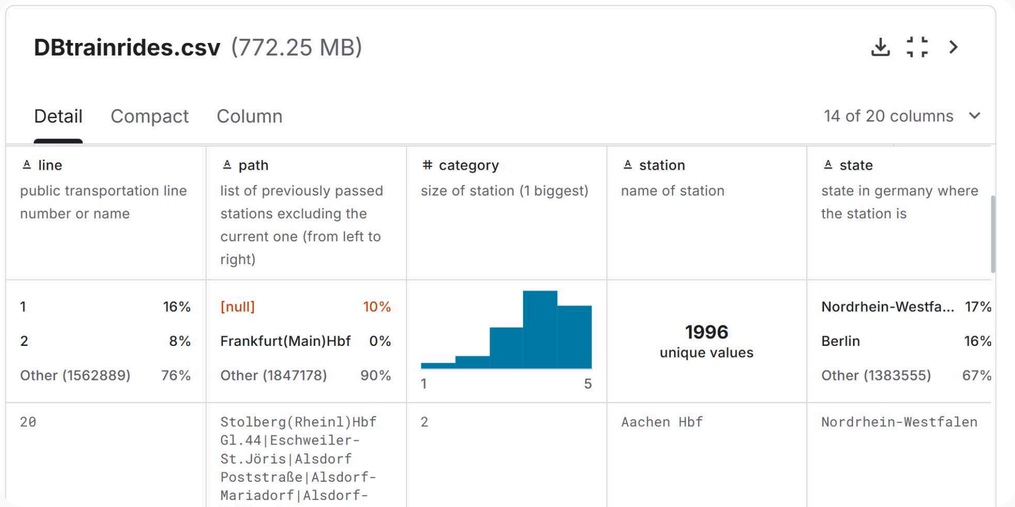
Using Supervised Regression to

Predict *Deutsche Bahn* Train Delays

PROBLEM

Train delays impact passengers and operations.

This project uses supervised regression to predict Deutsche Bahn train delays based on past schedule and station data, helping anticipate disruptions before they happen.

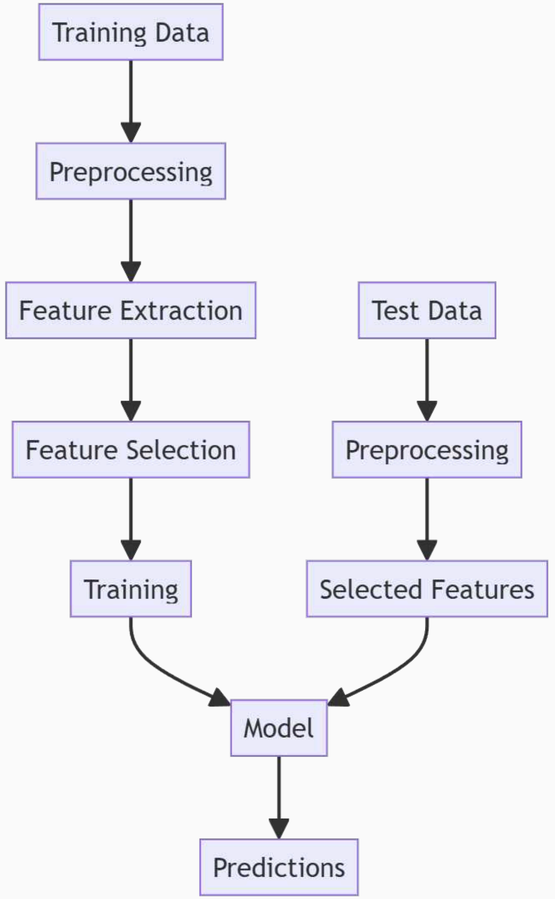


DATASET

2+ MILLION DEUTSCHE BAHN TRAIN RECORD

TARGET VARIABLE

`'arrival_delay_m'` **(CONTINUOUS)**



KEY PRINCIPLE

"The data quality should be good"

OUR APPROACH

SYSTEMATIC PREPROCESSING BEFORE MODELING

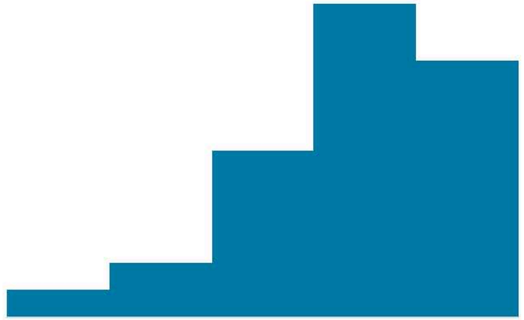
After choosing the dataset, we proceeded with

Data Quality *Assessment*

"Look at all your data (if possible)"

category

size of station (1 biggest)



Valid	2.06m	100%
Mismatched	0	0%
Missing	0	0%
Mean	3.87	
Std. Deviation	1.04	
Quantiles	1	Min
	5	Max

A station

name of station

1996
unique values

Valid	2.06m	100%
Mismatched	0	0%
Missing	0	0%
Unique	1996	
Most Common	München D...	0%

INITIAL DATA INSPECTION

```
Loading data from: /kaggle/input/deutsche-bahn-db-delays/DBtrainrides.csv

Loading dataset...
Dataset loaded successfully!
Shape: (2061357, 20)
Memory usage: 1700.75 MB
Memory Usage: 3.35 GB
```

DATASET SIZE

2,061,357 RECORDS × 20 FEATURES

MEMORY USAGE

1.7 GB

DATA TYPES

DATETIME, NUMERIC, CATEGORICAL

DUPLICATE ANALYSIS

6,725 DUPLICATES REMOVED (0.33%)

INITIAL DATA INSPECTION

DATASET SIZE

2,061,357 RECORDS × 20 FEATURES

MEMORY USAGE

1.7 GB

DATA TYPES

DATETIME, NUMERIC, CATEGORICAL

DUPLICATE ANALYSIS

6,725 DUPLICATES REMOVED (0.33%)

```
================================================================
Dataset Overview:
================================================================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2061357 entries, 0 to 2061356
Data columns (total 20 columns):
 #   Column                Dtype
---  ------                -----
 0   ID                    object
 1   line                  object
 2   path                  object
 3   eva_nr                int64
 4   category              int64
 5   station               object
 6   state                 object
 7   city                  object
 8   zip                   int64
 9   long                  float64
 10  lat                   float64
 11  arrival_plan          datetime64[ns]
 12  departure_plan        datetime64[ns]
 13  arrival_change        datetime64[ns]
 14  departure_change      datetime64[ns]
 15  arrival_delay_m       int64
 16  departure_delay_m     int64
 17  info                  object
 18  arrival_delay_check   object
 19  departure_delay_check object
```

INITIAL DATA INSPECTION

DATASET SIZE

2,061,357 RECORDS × 20 FEATURES

MEMORY USAGE

1.7 GB

DATA TYPES

DATETIME, NUMERIC, CATEGORICAL

DUPLICATE ANALYSIS

6,725 DUPLICATES REMOVED (0.33%)

```
================================================================================
DUPLICATE REMOVAL
================================================================================
Rows before removing duplicates: 2,061,357
Rows after removing duplicates: 2,054,632
Duplicates removed: 6,725 (0.33%)
```

MISSING VALUE *ANALYSIS*

CRITICAL FINDING

68.7% MISSING IN 'INFO' COLUMN

TARGET VARIABLE

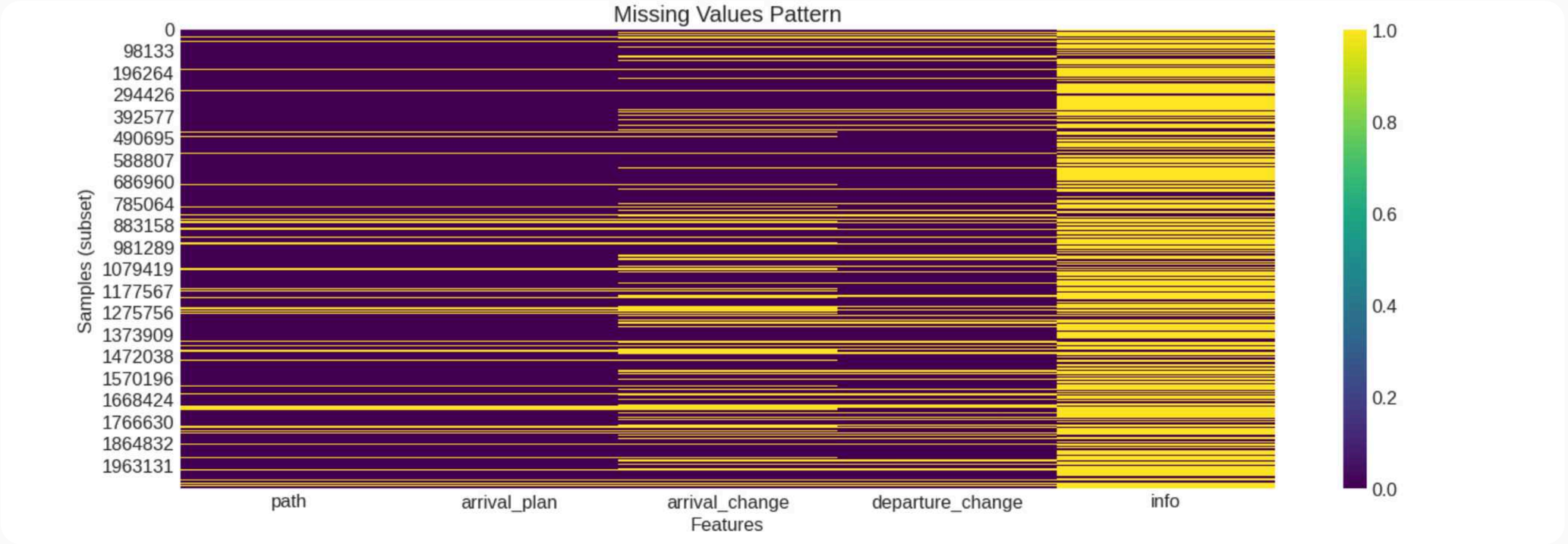
COMPLETE (NO MISSING VALUES)

STRATEGY

REMOVE ROWS WITH MISSING CRITICAL FEATURES

THEORY

MISSING DATA PATTERNS INFORM TREATMENT STRATEGY



MISSING VALUE *ANALYSIS*

CRITICAL FINDING

68.7% MISSING IN 'INFO' COLUMN

TARGET VARIABLE

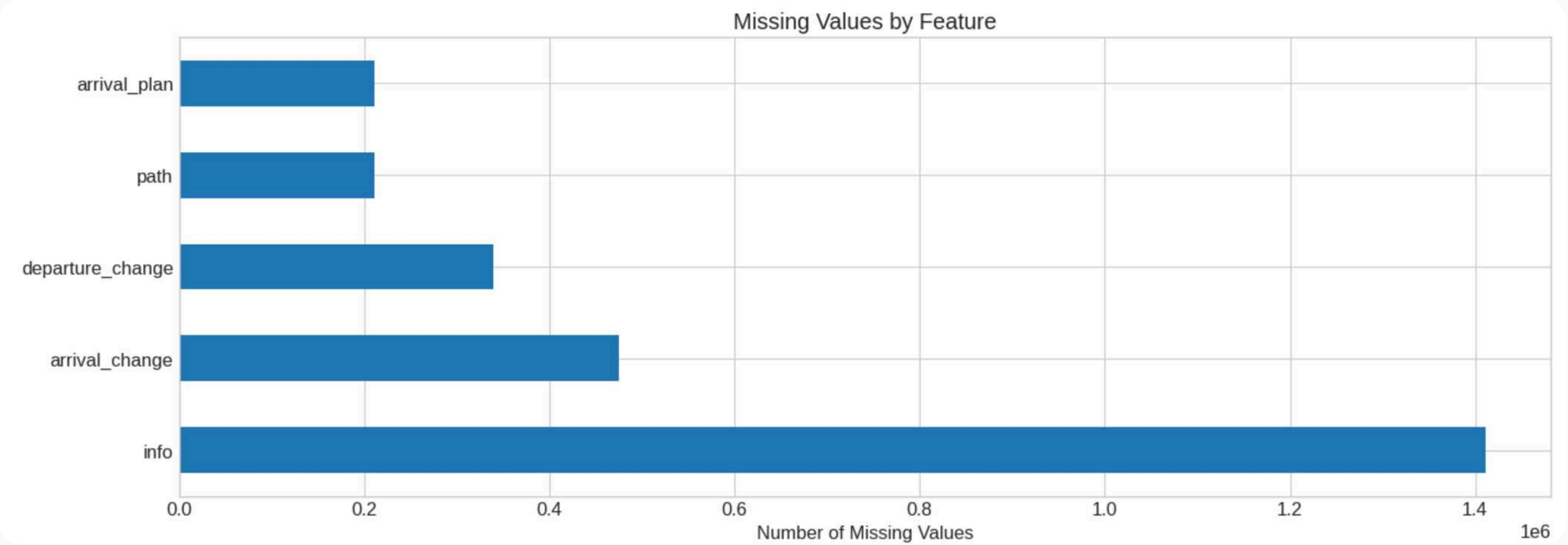
COMPLETE (NO MISSING VALUES)

STRATEGY

REMOVE ROWS WITH MISSING CRITICAL FEATURES

THEORY

MISSING DATA PATTERNS INFORM TREATMENT STRATEGY



Dataset Treatment

"Before doing classification/ regression experiments, you should be familiar with the data"

Target variable missing

Remove rows with missing delays

Critical features missing

Remove rows

Dataset Treatment

"Before doing classification/ regression experiments, you should be familiar with the data"

Result

2,054,632 clean records

SAMEER, FARES, ALEX

Feature Engineering

Lecture Warning

Temptation: Just take features with best correlation to your goal.
This is deeply wrong.

Correct Approach

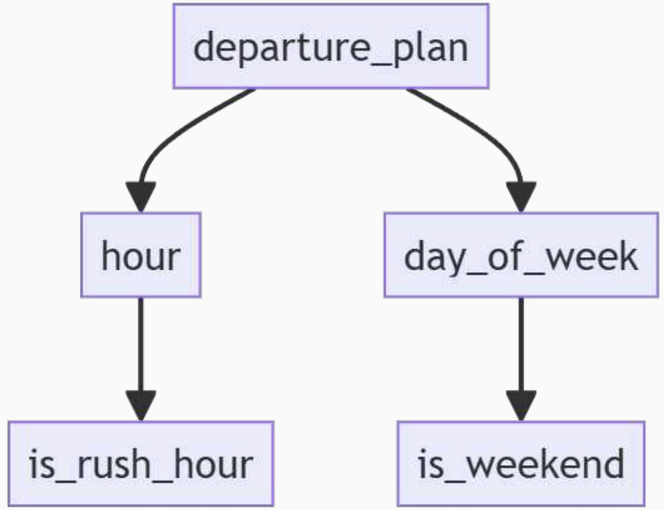
Domain-informed feature creation

NOT

Correlation-based selection

Implementation:

Time-based patterns




```
# Extract temporal features
print("Creating temporal features...")
df['hour'] = df['departure_plan'].dt.hour
df['day_of_week'] = df['departure_plan'].dt.dayofweek
df['is_weekend'] = (df['day_of_week'] >= 5).astype(int)
df['is_rush_hour'] = df['hour'].apply(lambda x: 1 if 6
```

Aggregated Features

"Past behavior predicts future performance"

'station_avg_delay'

Historical average delay per station

'station_std_delay'

Delay variability per station

'station_complexity'

Number of lines per station

SAMEER, FARES, ALEX

Exploratory

Data Analysis (EDA)

TARGET VARIABLE DISTRIBUTION

MEAN

1.2 MIN

MEDIAN

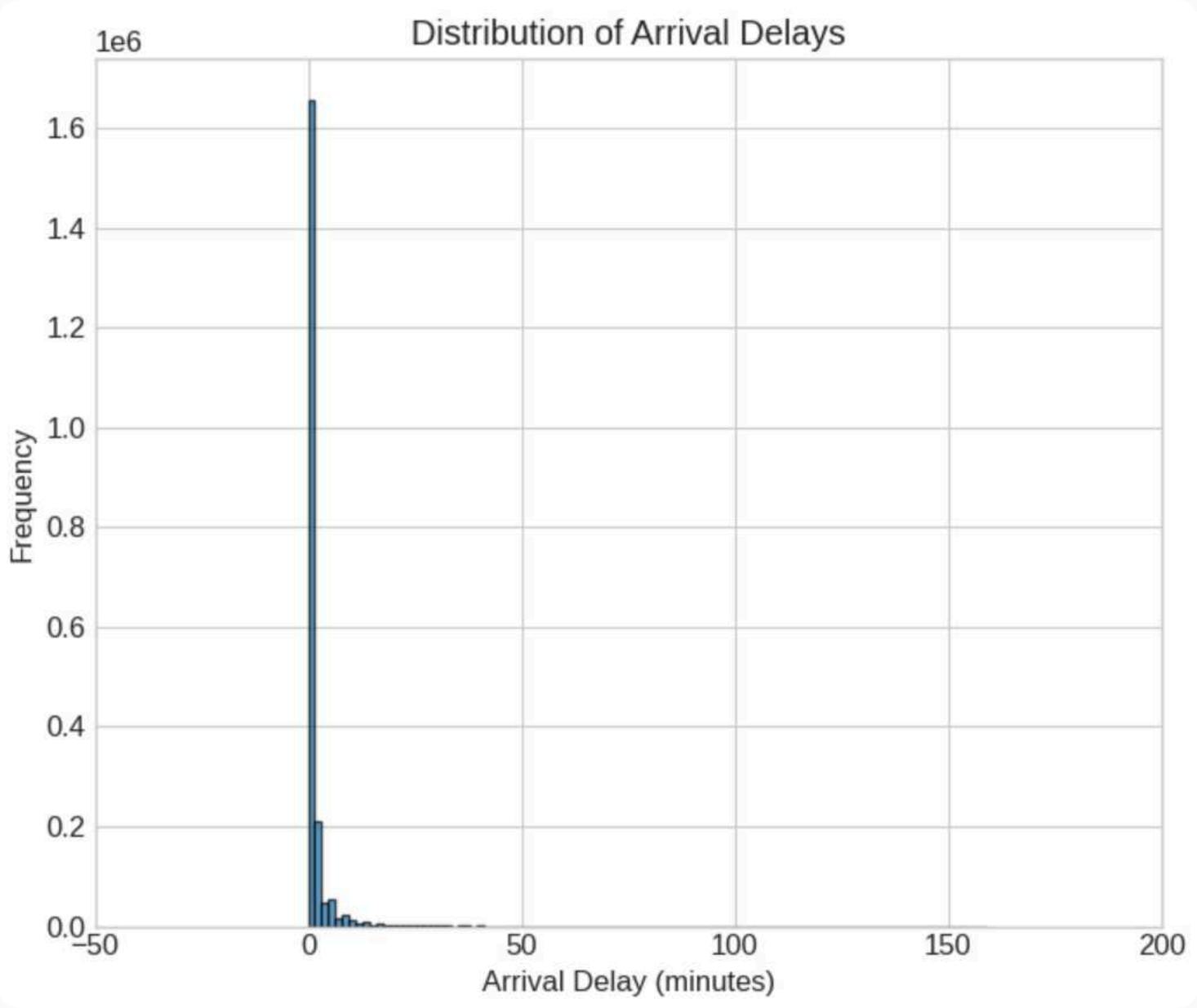
0 MINUTES (~70% ON TIME!)

MAX

159 MINUTES

KEY INSIGHT

MOST TRAINS PERFORM WELL, BUT OUTLIERS EXIST



TARGET VARIABLE DISTRIBUTION

MEAN

1.2 MIN

MEDIAN

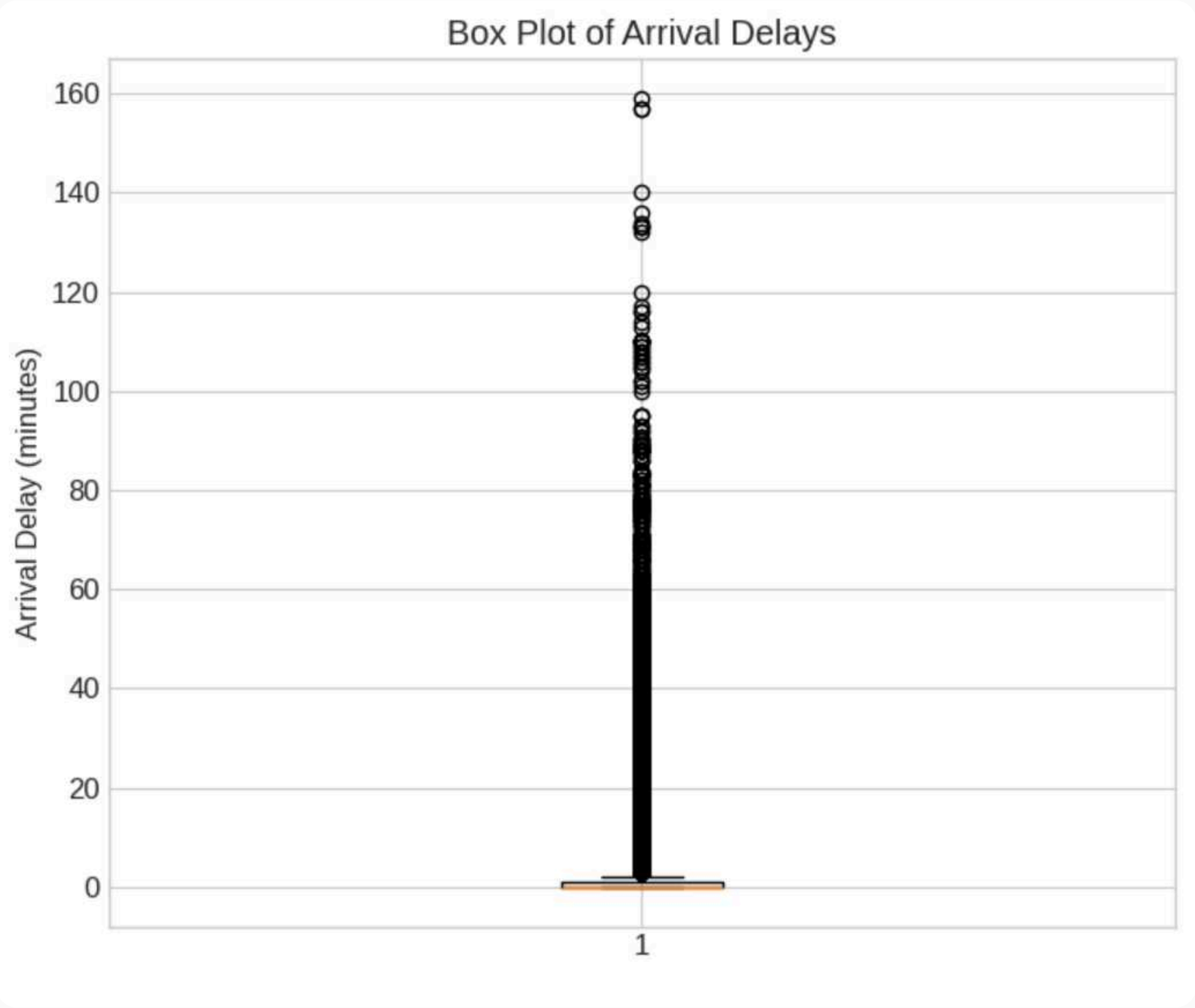
0 MINUTES (~70% ON TIME!)

MAX

159 MINUTES

KEY INSIGHT

MOST TRAINS PERFORM WELL, BUT OUTLIERS EXIST



KEY RELATIONSHIPS

Top Finding

Departure delay = strongest predictor (linear relationship)

TEMPORAL

Rush hours show higher delays

CATEGORICAL

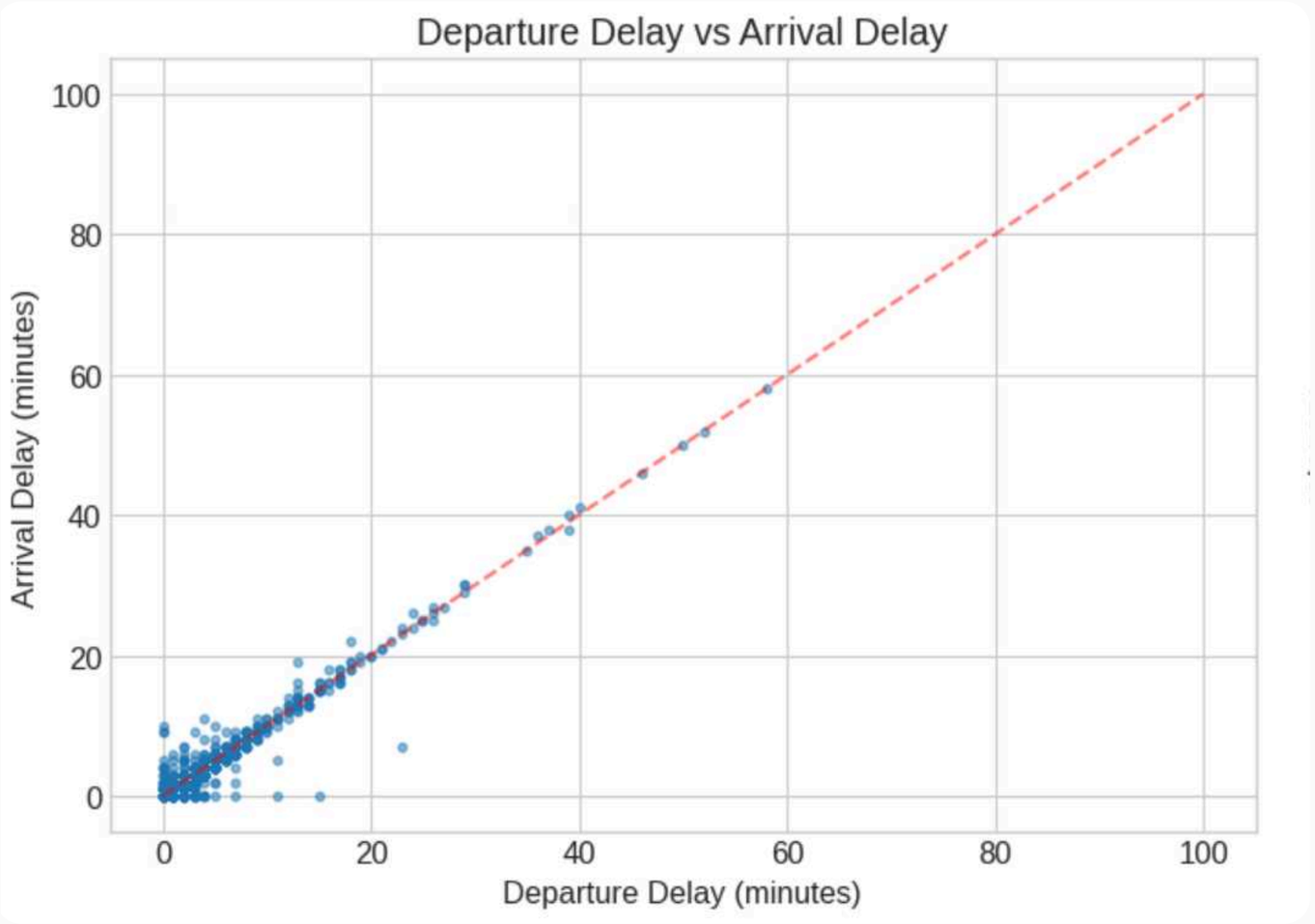
Train categories differ significantly

WEEKLY

Weekend effect visible

CORRELATION

Strong departure-arrival delay link



KEY RELATIONSHIPS

Top Finding

Departure delay = strongest predictor (linear relationship)

TEMPORAL

Rush hours show higher delays

CATEGORICAL

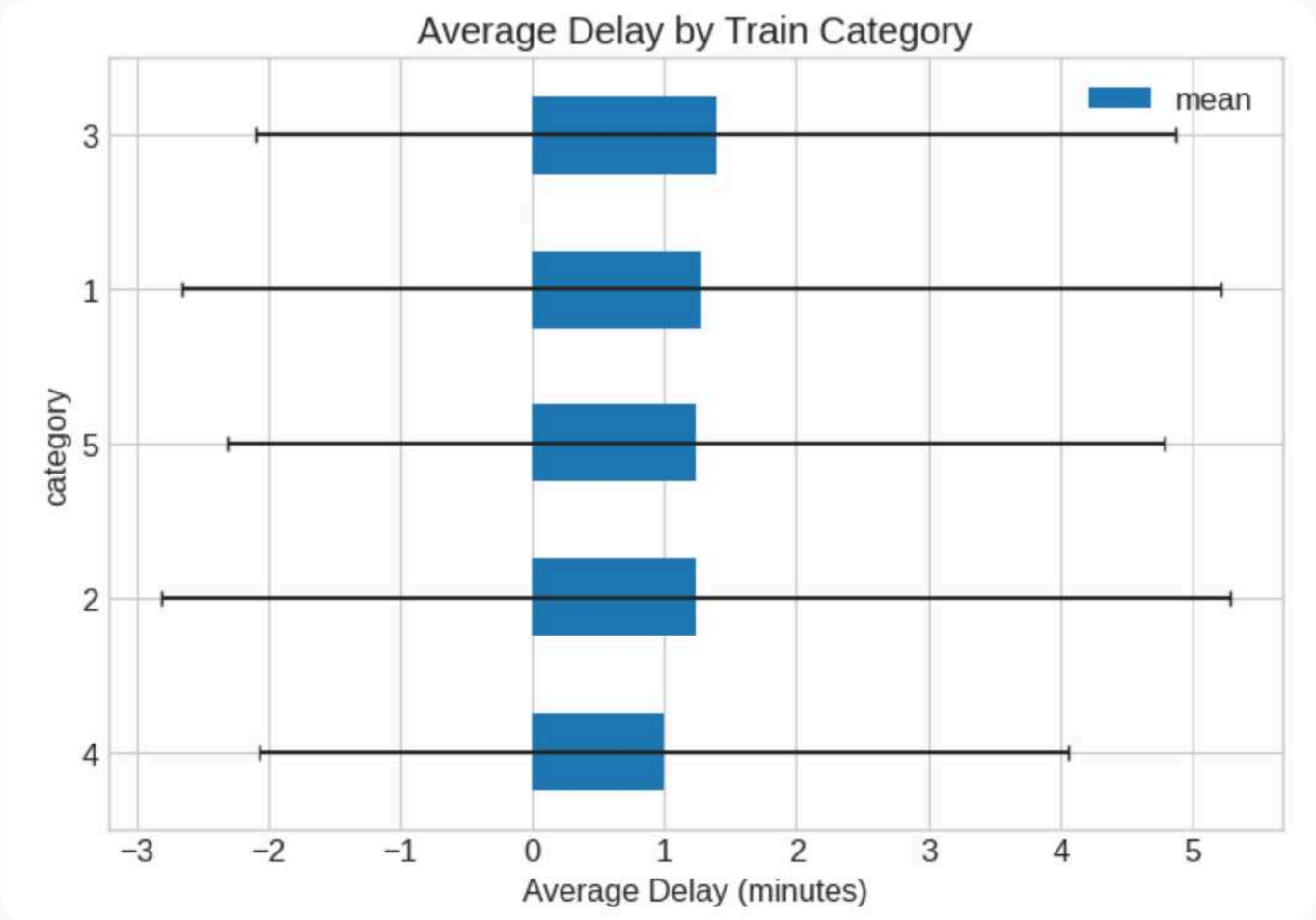
Train categories differ significantly

WEEKLY

Weekend effect visible

CORRELATION

Strong departure-arrival delay link



KEY RELATIONSHIPS

Top Finding

Departure delay = strongest predictor (linear relationship)

TEMPORAL

Rush hours show higher delays

CATEGORICAL

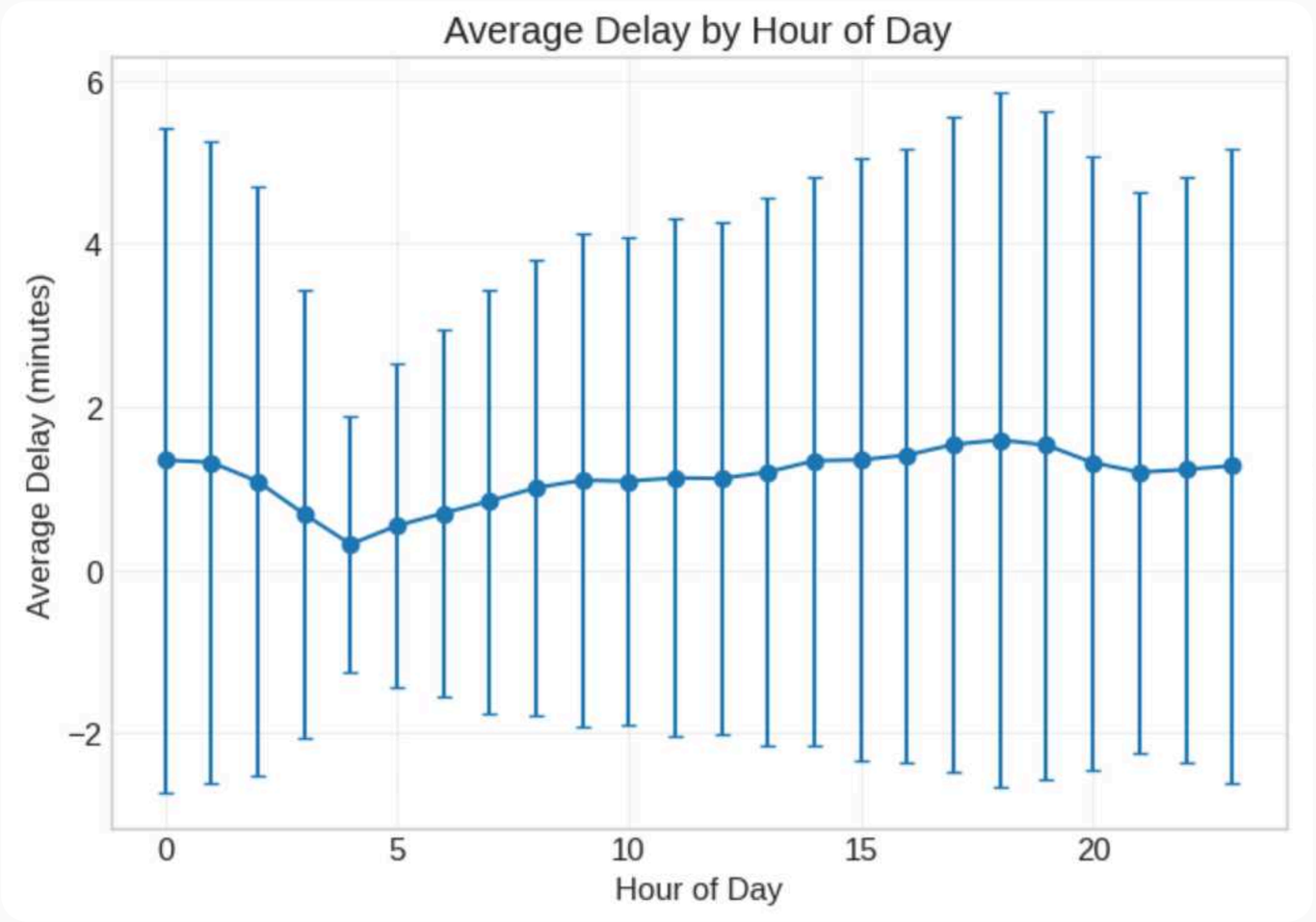
Train categories differ significantly

WEEKLY

Weekend effect visible

CORRELATION

Strong departure-arrival delay link



CATEGORICAL FEATURE CARDINALITY

Challenge

High-cardinality features

"Curse of dimensionality"

STATIONS

1,996 unique values

LINES

296 unique values

SOLUTION PREVIEW

Dimensionality reduction through aggregation

```
================================================================
DISTINCT-LEVEL ANALYSIS FOR CATEGORICAL FEATURES
================================================================

STATION:
  Distinct levels: 1996
  Most common: München Donnersbergerbrücke (8,725 occurrences)
  Least common: Neubrandenburg (5 occurrences)
  Rare levels (<100 obs): 5

LINE:
  Distinct levels: 296
  Most common: 1 (327,092 occurrences)
  Least common: N14 (2 occurrences)
  Rare levels (<100 obs): 36

STATE:
  Distinct levels: 16
  Most common: Nordrhein-Westfalen (341,778 occurrences)
  Least common: Bremen (10,428 occurrences)

CATEGORY:
  Distinct levels: 5
  Most common: 4 (786,108 occurrences)
  Least common: 1 (70,520 occurrences)

EVA_NR:
  Distinct levels: 1996
  Most common: 8004128 (8,725 occurrences)
  Least common: 8010241 (5 occurrences)
  Rare levels (<100 obs): 5
```

Now lets explore

Model Development *and (a bit of) Theory*

Problem

1,996 stations + 296 lines =
potential 2,292 dummy variables

Solution

Engineer single features instead

1. Create station complexity score instead of one-hot encoding

2. Create delay risk score combining station and line effects

Result

4 features instead of 2,292

TRAIN-VALIDATION-TEST SPLIT *(60-20-20)*

Data Splitting

```python
# Define features and target
feature_columns = [
    # Numerical feature
    'departure_delay_m',
    'station_complexity', 'station_delay_risk',
    'line_delay_risk', 'combined_delay_risk',
    'hour', 'day_of_week', 'is_weekend', 'is_rush_hour',
    'station_avg_delay', 'station_std_delay',
    # Categorical feature
    'category', 'state'
]
```

```
================================================
DATA SPLITTING
================================================
Total samples: 2,054,632
Number of features: 13

Dataset splits:
Training set: 1,232,778 samples (60.0%)
Validation set: 410,927 samples (20.0%)
Test set: 410,927 samples (20.0%)

Target variable statistics by split:
Train - Mean: 1.18, Std: 3.41
Val   - Mean: 1.19, Std: 3.44
Test  - Mean: 1.18, Std: 3.39
```

'StandardScaler' and 'OneHotEncoder' in Pipeline

"A good way to handle this is to standardize the data so that all variables are given a mean of zero and a standard deviation of one."

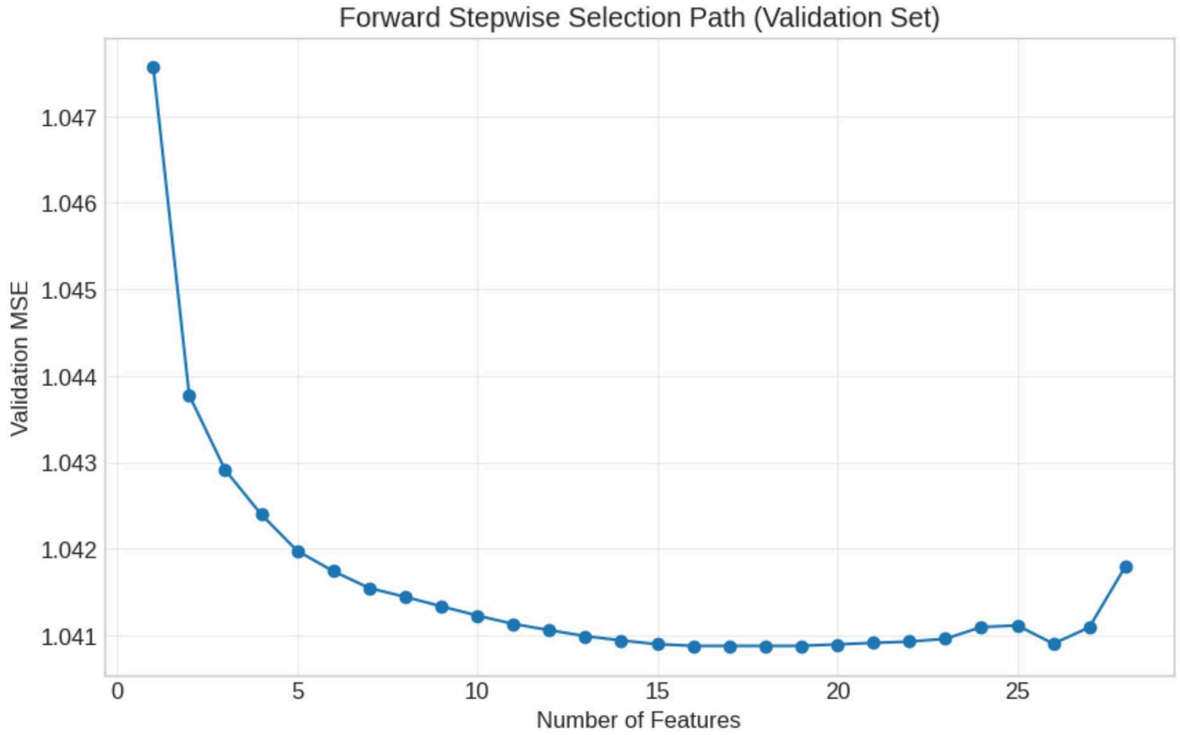
- ITSL Section 4.7.4

```
============================================================
PREPROCESSING PIPELINE SETUP
============================================================
Numerical features (9): ['departure_delay_m', 'station_complexity
ur', 'station_avg_delay', 'station_std_delay']
Categorical features (2): ['category', 'state']

Fitting preprocessor on training data...

Total features after preprocessing: 28
Preprocessing pipeline created and fitted on training data only!
```

Forward Stepwise Selection

"The selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model."
- ISLR Chapter 6.1.2

```
================================================================================
Using representative sample: n=10000, p=13

Performing forward stepwise selection (using validation set)...
Step 1: Added num__departure_delay_m, MSE=1.048
Step 2: Added num__station_delay_risk, MSE=1.044
Step 3: Added cat__state_Berlin, MSE=1.043
Step 4: Added cat__state_Hamburg, MSE=1.042
Step 5: Added cat__state_Nordrhein-Westfalen, MSE=1.042
```




```
Optimal number of features: 16
```

SAMEER, **FARES**, ALEX

Linear Model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$$

Assumptions

LINEARITY
Relationship is actually linear

INDEPENDENCE
Errors don't correlate

HOMOSCEDASTICITY
Constant error variance

NORMALITY
Errors follow bell curve

WHY CHECK?
Violations affect predictions

SAMEER, **FARES**, ALEX

Linear Model *Implementation*

1. Create and fit linear regression
2. Extract coefficients for interpretation
3. Make the predictions
4. Evaluate the performance

```
1. LINEAR REGRESSION (OLS)
--------------------------------------

Fitting via ordinary least squares...

Model fitted with 28 coefficients
Intercept (β₀): 0.979

Training RSS: 1,218,795
Training MSE: 0.99
Validation MSE: 1.03
```

Key Finding

4% gap suggests good generalization

RESIDUAL DIAGNOSTICS

RESIDUAL STANDARD DEVIATION

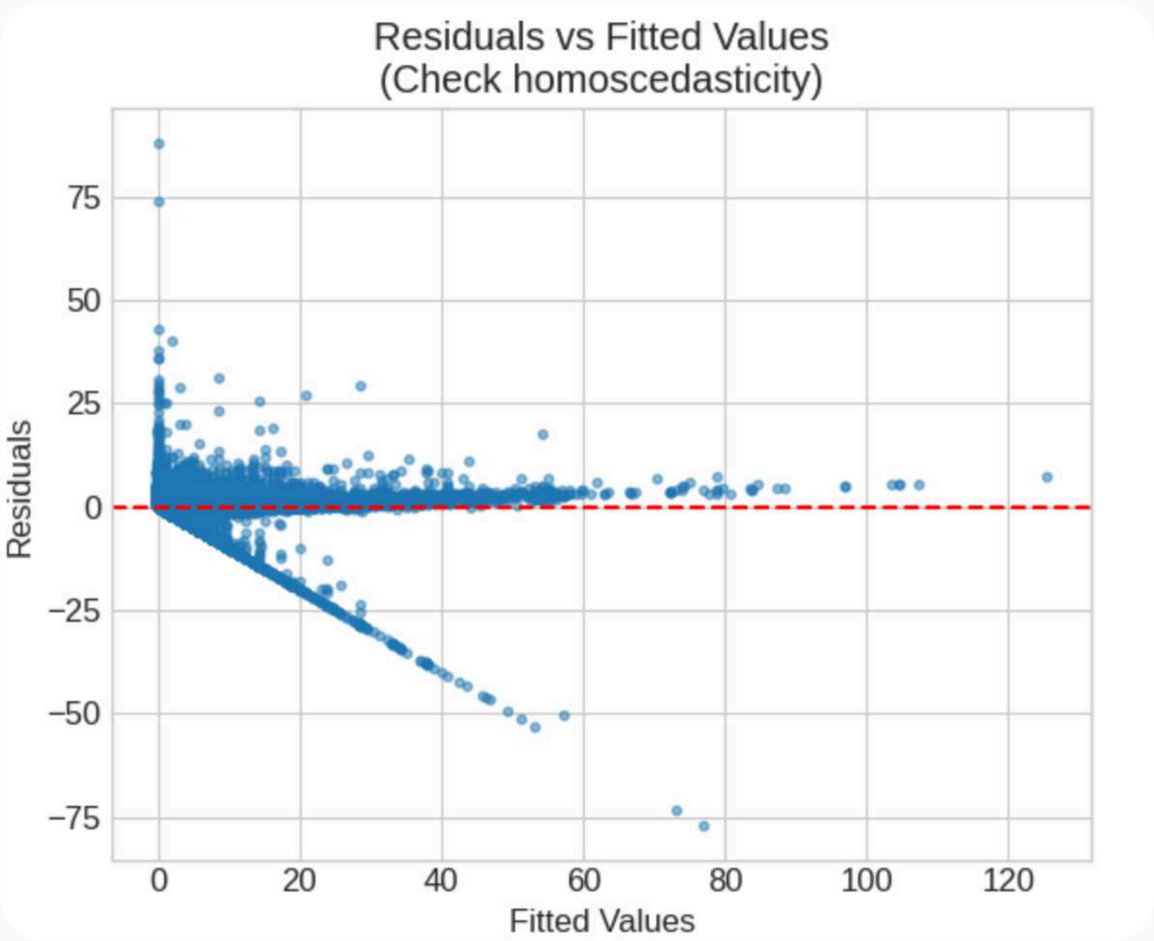
σˆ = 1.017

FINDING

0.27% outliers (|standardized residual| > 3)

CONCLUSION

Assumptions reasonably satisfied



KNN

"Based on human reasoning, not statistics or optimization"

SMALL K → LOW BIAS, HIGH VARIANCE (*FLEXIBLE*)

Prediction depends on few points, sensitive to noise

LARGE K → HIGH BIAS, LOW VARIANCE (*SMOOTH*)

Averages over many points, may miss local patterns

OUR TESTING

$K \in \{3, 5, 7, 9, 15, 25, 50\}$

```
2. K-Nearest Neighbors (KNN)
----------------------------------------
Using only forward-selected features for KNN (no SelectKBest)
Using 50,000 training and 10,000 validation samples for hyperparameter tuning
Testing K values from 3 (low bias, high variance) to 50 (high bias, low variance)
Fitting 1 folds for each of 14 candidates, totalling 14 fits

Best KNN parameters: {'metric': 'euclidean', 'n_neighbors': 9, 'weights': 'distance'}
Best CV MSE: 1.45
This means K=9 neighbors

KNN - Training Performance:
  MSE: 1.08 (minutes²)

KNN - Validation Performance:
  MSE: 1.13 (minutes²)

Chosen K=9 indicates smoother model (higher bias, lower variance)
```

SAMEER, **FARES**, ALEX

KNN

"Based on human reasoning, not statistics or optimization"

SMALL K → LOW BIAS, HIGH VARIANCE (*FLEXIBLE*)

Prediction depends on few points, sensitive to noise

LARGE K → HIGH BIAS, LOW VARIANCE (*SMOOTH*)

Averages over many points, may miss local patterns

OUR TESTING

K ∈ {3, 5, 7, 9, 15, 25, 50}

```
2. K-Nearest Neighbors (KNN)
----------------------------------------
Using only forward-selected features for KNN (no SelectKBest)
Using 50,000 training and 10,000 validation samples for hyperparameter tuning
Testing K values from 3 (low bias, high variance) to 50 (high bias, low variance)
Fitting 1 folds for each of 14 candidates, totalling 14 fits

Best KNN parameters: {'metric': 'euclidean', 'n_neighbors': 9, 'weights': 'distance'}
Best CV MSE: 1.45
This means K=9 neighbors

KNN - Training Performance:
  MSE: 1.08 (minutes²)

KNN - Validation Performance:
  MSE: 1.13 (minutes²)

Chosen K=9 indicates smoother model (higher bias, lower variance)
```

Random Forest

Why it works

Decorrelation Principle

SINGLE TREE

High variance, low bias

BAGGING

Reduces variance by averaging

RANDOM FEATURES

Decorrelates trees

RESULT

More effective variance reduction

```
3. Random Forest
----------------------------------------
Using only forward-selected features for Random Forest (no SelectKBest)
Performing hyperparameter tuning for Random Forest...
Note: Each tree uses bootstrap sampling of the training data
At each split, m ≈ √p features are randomly selected
Using 50,000 training and 10,000 validation samples for hyperparameter tuning
Fitting 1 folds for each of 24 candidates, totalling 24 fits

Best Random Forest parameters: {'max_depth': None, 'min_samples_leaf': 2, 'min_
Best CV MSE: 0.33

Random Forest - Training Performance:
  MSE: 0.98 (minutes²)

Random Forest - Validation Performance:
  MSE: 1.02 (minutes²)

Random Forest advantages over single tree:
- Bootstrap aggregation reduces variance
- Random feature selection (m ≈ √p) decorrelates trees
- Averaging decorrelated trees is more effective than correlated ones
```

Now, lets move to

Model Evaluation *and Deployment*

"The reality: Small data sets"
- 07_outliers_resampling

Our Reality: Large dataset (1.2M training samples)

Solution: Validation set approach

Our Reality: Large dataset (1.2M training samples)

Strategy

SPLIT

60% train, 20% validation, 20% test

Use validation for model selection

Reserve test for final unbiased estimate

ADVANTAGE

Computationally efficient for large data

```
================================================================================
MODEL SELECTION VIA VALIDATION SET PERFORMANCE
================================================================================
Theory Application: Using validation set approach on 100,000 samples
Justification: With n=1,232,778, validation variance is acceptable

Validation Set Performance (Model Selection):
------------------------------------------------
Linear          Validation MSE: 1.0352
KNN (K=9)       Validation MSE: 1.1491
RandomForest    Validation MSE: 0.9502

Selected best model by validation set: RandomForest
```

WINNER

Random Forest (5% better than Linear)

SAMEER, FARES, **ALEX**

What Learning Curves Reveal

X-AXIS

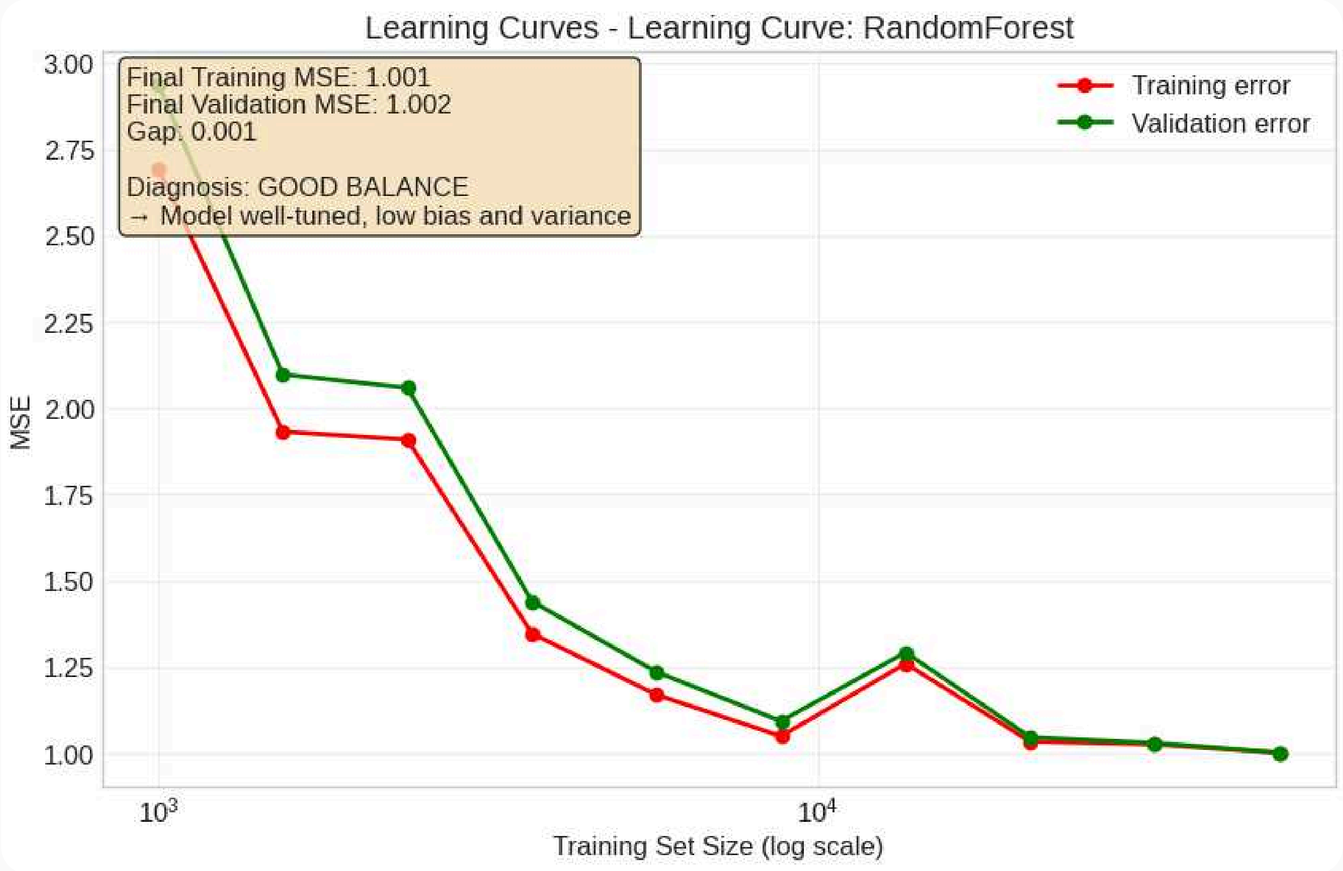
High variance, low bias

Y-AXIS

Error (MSE)

TWO LINES

Training error and validation error



SAMEER, FARES, **ALEX**

Diagnosis Patterns

HIGH BIAS

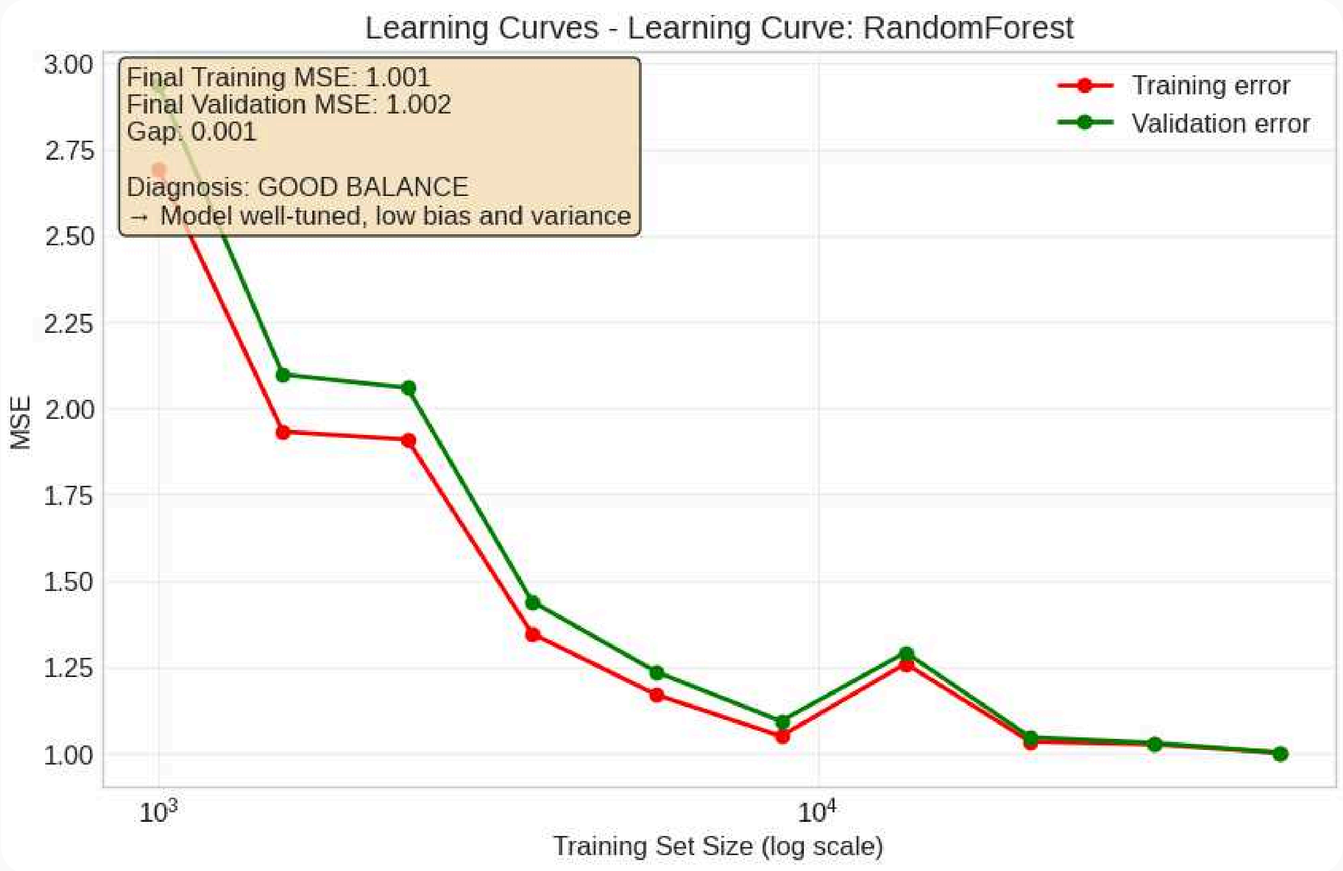
Both errors high and close together

HIGH VARIANCE

Big gap between training and validation

GOOD MODEL

Low error, small gap



SAMEER, FARES, **ALEX**

If High Bias (Underfitting)

- Add polynomial features (square terms, interactions)
- Try more complex model
- Reduce regularization

OUR MODEL

Not needed - error already low

If High Variance (Overfitting)

- Add regularization (Ridge/Lasso)
- Simplify model
- Simplify model

OUR MODEL

Not needed - gap near zero

SAMEER, FARES, **ALEX**

Final Model *and Deployment*

PRODUCTION MODEL TRAINING

"Use all available data for final model"

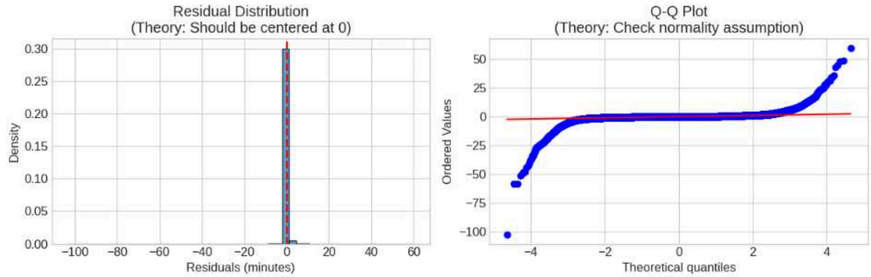
Production Pipeline

1. Combine training + validation = 1.64M samples
2. Retrain Random Forest with same hyperparameters
3. Make final predictions on test set

> **CRITICAL**
>
> *No new tuning! (Would invalidate test)*

> **TRAINING TIME**
>
> *340 seconds*

```
================================================================================
FINAL MODEL TRAINING (THEORY: FULL DATA UTILIZATION)
================================================================================
Step 1: Combining training + validation sets
Original training size: 1,232,778
Validation size: 410,927
Combined size: 1,643,705
Theory (Lecture 07): 'We want our train data to be as large as possible'

Step 2: Retraining best model on combined data
Theory: Using mini-batch training for efficiency
Training completed in 876.0 seconds

Step 3: Single test set evaluation (unbiased estimate)

Final Test Performance:
- Test MSE: 0.7520
- Test RMSE: 0.8672 minutes
- Test MAE: 0.2705 minutes

Theoretical Baseline Comparison:
- Baseline MSE (predict mean): 11.5403
- Improvement over baseline: 93.5%
```




```
Theoretical Conclusions:
----------------------------------------
1. Model Selection: RandomForest minimized validation set error
2. Bias-Variance (ITSL § 2.2.2 & Lecture 06): Final Training MSE: 1.001
Final Validation MSE: 1.002
Gap: 0.001

Diagnosis: GOOD BALANCE
→ Model well-tuned, low bias and variance
3. Test Performance: RMSE = 0.867 minutes
4. Statistical Significance: 93.5% better than baseline
```

FINAL TEST PERFORMANCE

Unbiased Test Result

TEST MSE

0.8791

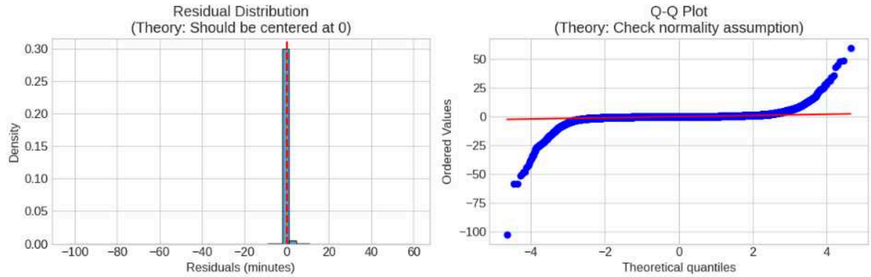
BASELINE MSE

11.54 *(predicting average delay)*

IMPROVEMENT

92.4% *better than baseline!*

```
================================================================================
FINAL MODEL TRAINING (THEORY: FULL DATA UTILIZATION)
================================================================================
Step 1: Combining training + validation sets
Original training size: 1,232,778
Validation size: 410,927
Combined size: 1,643,705
Theory (Lecture 07): 'We want our train data to be as large as possible'

Step 2: Retraining best model on combined data
Theory: Using mini-batch training for efficiency
Training completed in 876.0 seconds

Step 3: Single test set evaluation (unbiased estimate)

Final Test Performance:
- Test MSE: 0.7520
- Test RMSE: 0.8672 minutes
- Test MAE: 0.2705 minutes

Theoretical Baseline Comparison:
- Baseline MSE (predict mean): 11.5403
- Improvement over baseline: 93.5%
```




```
Theoretical Conclusions:
----------------------------------------
1. Model Selection: RandomForest minimized validation set error
2. Bias-Variance (ITSL § 2.2.2 & Lecture 06): Final Training MSE: 1.001
Final Validation MSE: 1.002
Gap: 0.001

Diagnosis: GOOD BALANCE
→ Model well-tuned, low bias and variance
3. Test Performance: RMSE = 0.867 minutes
4. Statistical Significance: 93.5% better than baseline
```

SAMEER, FARES, **ALEX**

It's time for

Outlook and **Summary**

Real-life datasets aren't always balanced.

> Target variable (`'arrival_delay_m'`) had 68.3% zero values

> This may lead to bias (heteroscedasticity)

It isn't easy to agree on one approach (especially in a team)

> In most times, there isn't a single right answer to a certain problem

Dominant predictor ≠ one-feature model

> In our case, `'departure_delay_m'` had the highest importance to the target feature

> But forward selection still needed 15 other features to reach the least MSE rate.

Re-frame as a classification problem

> Using `'arrival_delay_check'` as a target variable

> Predicting if a train would be delayed or not (>6 minutes according to the dataset)

Polynomial-regression upgrade

> **Degree-2** terms (delay², hour²)

> Apply Ridge regularisation to keep variance under control

Explicit feature-space lifting

> Ex. Add `'station_traffic × departure_delay_m'` interaction

Deploy & monitor in real time

> Daily re-score via batch API to flag risky trains

Thank you