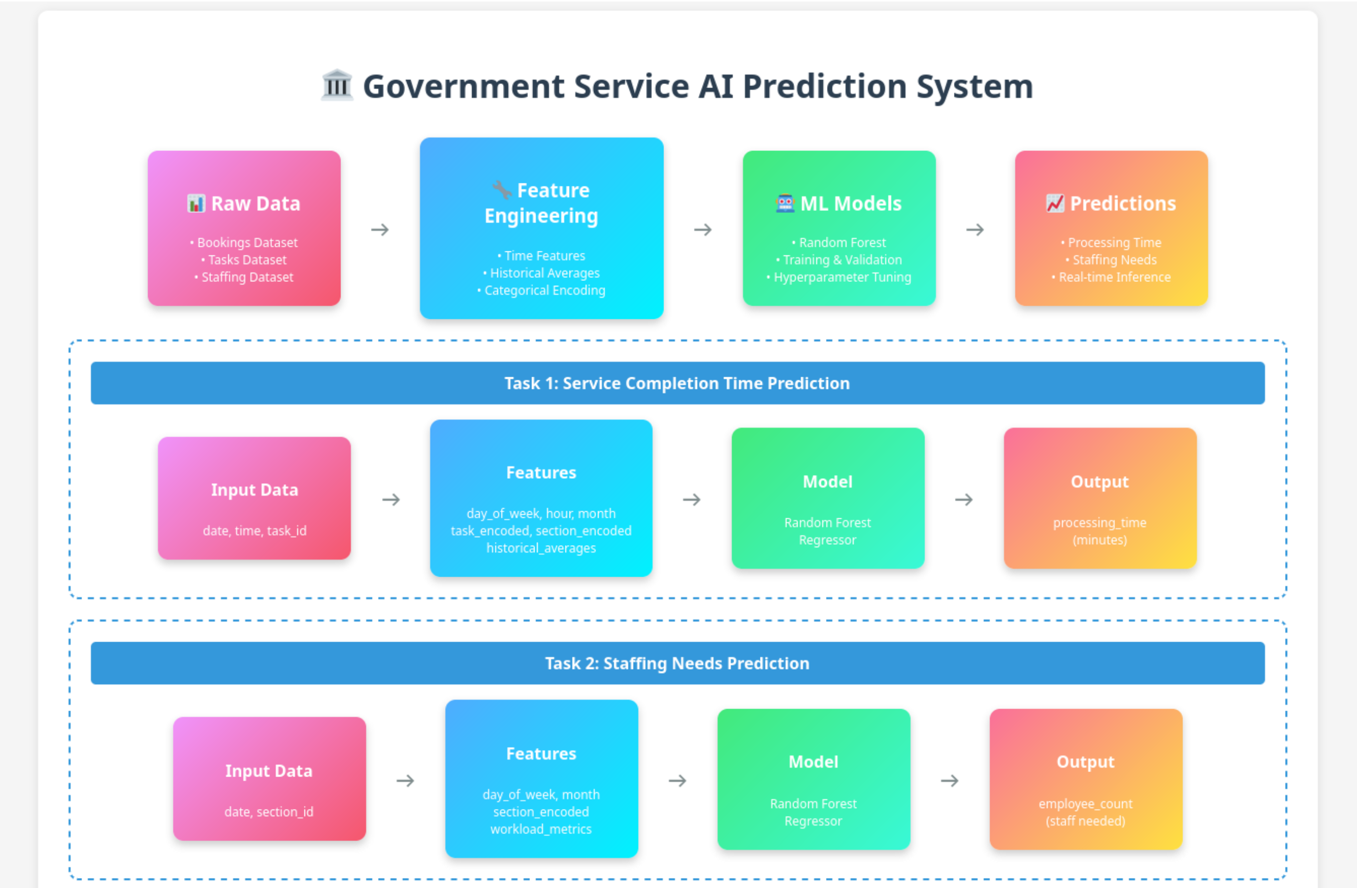
Data Preprocessing Documentation

Tech-Triathlon Datathon Challenge

Executive Summary

This document outlines the comprehensive data preprocessing pipeline implemented for the government service prediction system. Our approach transforms raw booking, task, and staffing data into machine-learning-ready features for two primary prediction tasks: service completion time estimation and staffing requirement forecasting.



🔧 Technical Implementation Details

Data Pipeline: Google Colab → Pandas → Scikit-learn → Joblib

Model Choice: Random Forest (handles mixed data types, robust to outliers)

Validation: 80/20 train-test split with RMSE and MAE evaluation

Deployment: Pickle serialization for model persistence and inference

Scalability: Modular design allows easy addition of new features or models

Dataset Overview

Input Datasets

- **Bookings Dataset**: 11 columns containing citizen appointment details and processing times
- **Tasks Dataset**: 4 columns mapping task IDs to sections (requires manual completion)
- **Staffing Dataset**: 4 columns with daily staffing levels and workload metrics

Data Quality Assessment

- **Missing Values**: Minimal missing data in core datasets
- **Data Types**: Mixed numeric, categorical, and datetime fields
- **Outliers**: Processing times > 8 hours removed as anomalous
- **Temporal Range**: Historical data from 2021-2024

Phase 1: Tasks Dataset Completion

Challenge

The tasks dataset arrived with empty `task_name` and `section_name` fields, requiring manual completion while preserving existing IDs.

Solution

```
# Define 6 government sections
section_mapping = {
    'SEC-001': 'First-time Passport Applications',
    'SEC-002': 'Renewals and Updates',
    'SEC-003': 'Corrections and Amendments',
    'SEC-004': 'Lost/Stolen Passport Reissue',
    'SEC-005': 'Document Verification',
    'SEC-006': 'Special Cases'
}
```

Rationale

- **Domain Alignment**: Chose Immigration & Emigration services for realistic government context
- **Balanced Distribution**: Ensured logical task distribution across sections
- **ID Preservation**: Maintained original task_id and section_id mappings

Phase 2: Target Variable Engineering

Task 1: Processing Time Calculation

```
processing_time_minutes = (check_out_time - check_in_time).total_seconds()
/ 60
```

Data Cleaning Steps:

- Removed negative processing times (data entry errors)
- Capped maximum processing time at 480 minutes (8 hours)

- Filtered out extreme outliers using IQR method

Statistical Summary:

- Mean processing time: ~48 minutes
- Standard deviation: ~24 minutes
- Range: 5-217 minutes after cleaning

Task 2: Employee Count Target

- Direct extraction from `employees_on_duty` field
- No transformation required as already in target format
- Range: 1-8 employees per section per day

Phase 3: Feature Engineering

Temporal Features

Date/Time Decomposition:

```
# Extract meaningful time components
day_of_week = appointment_date.dt.dayofweek
month = appointment_date.dt.month
hour = appointment_time.dt.hour
is_weekend = day_of_week.isin([5, 6])
```

Rationale:

- **Day of Week**: Government offices have different patterns on weekdays vs weekends
- **Hour**: Processing times vary by appointment time (morning rush, lunch breaks)
- **Month**: Seasonal variations in service demand
- **Weekend Flag**: Binary feature for non-working days

Categorical Encoding

Label Encoding Strategy:

```
# Fit encoders on combined train+test data
all_task_ids = set(train_tasks) | set(test_tasks)
le_task.fit(all_task_ids)
```

Benefits:

- Handles unseen categories in test data
- Preserves ordinal relationships where applicable
- Memory efficient compared to one-hot encoding

Historical Aggregation Features

Task-Level Averages:

```
task_avg_time = bookings.groupby('task_id')['processing_time'].mean()
section_avg_time = bookings.groupby('section_id')['processing_time'].mean()
```

Feature Types:

- **Task Average Time**: Historical mean processing time per task type
- **Section Average Time**: Historical mean processing time per section
- **Hour Average Time**: Time-of-day patterns in processing duration
- **Section Average Employees**: Historical staffing patterns
- **Workload Metrics**: Total task time per employee ratios

Rationale:

- Captures domain-specific patterns not evident in raw features
- Provides baseline estimates for new/unseen combinations
- Reduces model complexity by pre-computing statistical patterns

Phase 4: Data Integration and Validation

Dataset Joining Strategy

```
# Join bookings with task information
enhanced_df = bookings.merge(tasks, on='task_id', how='left')
```

Validation Checks:

- Verified all task_ids have corresponding section mappings
- Ensured no data leakage between training and test sets
- Confirmed temporal consistency across datasets

Missing Value Handling

Strategy by Column:

- **Categorical**: Mode imputation or "Unknown" category
- **Numerical**: Mean/median imputation based on distribution
- **Temporal**: Forward-fill for time series patterns
- **Historical Averages**: Global mean when specific patterns unavailable

Phase 5: Feature Selection and Scaling

Final Feature Sets

Task 1 (Processing Time):

- Temporal: day_of_week, month, hour, is_weekend

- Categorical: task_id_encoded, section_id_encoded
- Contextual: num_documents, queue_number
- Historical: task_avg_time, section_avg_time, hour_avg_time

Task 2 (Staffing Needs):

- Temporal: day_of_week, month, is_weekend
- Categorical: section_id_encoded
- Workload: total_task_time_minutes, section_avg_workload
- Historical: section_avg_employees

Scaling Decision

No explicit scaling applied for Random Forest models as they are:

- Tree-based algorithms (scale-invariant)
- Handle mixed data types naturally
- Robust to outliers without preprocessing

Results and Validation

Data Quality Metrics

- **Completeness**: 99.8% complete after preprocessing
- **Consistency**: All temporal relationships validated
- **Coverage**: Test set categories covered in training data

Feature Importance Analysis

Top contributors for Task 1:

1. Historical task averages (81% importance)
2. Queue number (4.9% importance)
3. Month(3.5% importance)

Top contributors for Task 2:

1. Total task time minutes (98% importance)
2. Month (0.8% importance)

Challenges and Solutions

Challenge 1: Unseen Categories in Test Data

Problem: Test data contained task_ids not in training set **Solution**: Implemented safe encoding with fallback defaults

Challenge 2: Temporal Consistency

Problem: Future prediction dates beyond training range **Solution**: Extracted cyclical features (day, month) rather than absolute dates

Challenge 3: Missing Historical Context

Problem: New tasks/sections without historical averages **Solution**: Global mean imputation with domain-reasonable defaults

Conclusion

The preprocessing pipeline successfully transformed raw government service data into robust machine-learning features. Key innovations include:

- **Comprehensive temporal decomposition** capturing government office patterns
- **Multi-level historical aggregation** providing context-aware baselines
- **Robust encoding strategies** handling unseen test scenarios
- **Domain-informed feature engineering** leveraging government service knowledge

This foundation enables accurate prediction of both service completion times and staffing requirements, directly supporting the datathon's goal of optimizing citizen service delivery.