File Formats Description

Flight Cancellation Prediction System

Data Science Project

January 9, 2026

1 Introduction

This document describes the file formats used for the "Prediction Objects" in the Flight Cancellation Prediction System. These files enable the server to load machine learning models and the data preparation pipeline at startup.

2 Manifest File (`prediction_objects.json`)

The entry point for the system configuration. It acts as a catalog, telling the server where to find the pipeline and model artifacts.

2.1 Format

JSON (JavaScript Object Notation)

2.2 Structure

```
{
    "description": "String description of the file",
    "pipeline_definition": "Path to the Python file (e.g., pipeline.py)",
    "models": {
        "model_key_1": "Path to serialized model file",
        "model_key_2": "Path to serialized model file"
    },
    "preprocessing": "Path to pipeline state file (e.g., models/pipeline.joblib)",
    "hyperparameters": {
        "model_key": {
            "algorithm": "Name of the algorithm class",
            "params": {
                "param_name": "param_value"
            }
        }
    },
    "pipeline_description": {
        "steps": ["List of strings describing steps"],
        "scaling_method": "String name of scaling method",
        "encoding_method": "String name of encoding method"
    }
}
```

2.3 Key Fields

Field	Description
`description`	Human-readable description of the manifest
`pipeline_definition`	Path to the Python file containing the `PredictionPipeline` class
`models`	Dictionary mapping model keys to their `.joblib` file paths
`preprocessing`	Path to the consolidated pipeline state file
`hyperparameters`	Dictionary of hyperparameters used to train each model
`pipeline_description`	Steps and methods used in the data preparation pipeline

3 Model Files (`*.joblib`)

Each classification model is serialized into its own binary file using the `joblib` library. This allows for efficient storage and loading of large NumPy arrays often found in scikit-learn models.

3.1 Format

Joblib Binary Serialization (Python-specific)

3.2 Content

A single Python object corresponding to a trained scikit-learn estimator.

3.3 Available Models

File	Scikit-learn Class
`models/naive_bayes.joblib`	`sklearn.naive_bayes.GaussianNB`
`models/knn.joblib`	`sklearn.neighbors.KNeighborsClassifier`
`models/logistic_regression.joblib`	`sklearn.linear_model.LogisticRegression`
`models/decision_tree.joblib`	`sklearn.tree.DecisionTreeClassifier`
`models/mlp.joblib`	`sklearn.neural_network.MLPClassifier`
`models/random_forest.joblib`	`sklearn.ensemble.RandomForestClassifier`

4 Pipeline State File (`models/pipeline.joblib`)

The data preparation pipeline's internal state is consolidated into a single binary file to satisfy the "One single file" requirement from the project specification.

4.1 Format

Joblib Binary Serialization (Python-specific)

4.2 Content

A Python `dictionary` containing all necessary objects to transform raw data.

4.3 Dictionary Keys

Key	Content
`"encoder"`	Fitted `sklearn.preprocessing.OrdinalEncoder`
`"scaler"`	Fitted `sklearn.preprocessing.MinMaxScaler`
`"cat_cols"`	List of categorical column names
`"shifts"`	Dictionary mapping columns to shift values (to handle negative encoded integers)
`"mvi_stats"`	Dictionary containing mean/mode values for missing value imputation
`"final_features"`	List of column names expected by the trained models

5 Pipeline Definition (`pipeline.py`)

While not a data file, this Python source file defines the `PredictionPipeline` class logic. The server imports this class and populates it with the data loaded from `models/pipeline.joblib`.

5.1 Key Methods

- `__init__()` — Initializes the pipeline and loads all artifacts from disk.
- `transform(data)` — Applies the full transformation pipeline to input data.
- `predict_single(instance, model_name)` — Predicts a single instance using the specified model.
- `evaluate(df, model_name)` — Evaluates model performance on a validation dataset.