

AIRLINE FLIGHT DATA

MALLA REDDY ENGINEERING COLLEGE
FOR WOMEN
(TELANGANA- HYDERABAD)

P. THRISHUJUNA
B-TECH - IV YR

AIRLINES

Milestone 1: Data Foundation and Cleaning

Week 1: Project Initialization and Dataset Setup

- Define goals, KPIs, and workflow
- Load CSVs using pandas
- Explore schema, types, size, and nulls
- Perform sampling and memory optimizations

Week 2: Preprocessing and Feature Engineering

- Handle nulls in delay and cancellation columns
- Create derived features: Month, Day of Week, Hour, Route
- Format datetime columns
- Save preprocessed data for fast reuse

Deliverables:

- Cleaned dataset
- Summary of preprocessing logic
- Feature dictionary

INTRODUCTION

The **AirFly Insights** project focuses on analyzing large-scale airline flight data to uncover operational trends, delay patterns, and cancellation reasons. By leveraging data visualization and exploratory data analysis (EDA) techniques, the project aims to provide actionable insights that can help airlines, airports, and analysts improve decision-making, optimize flight schedules, and enhance passenger experience.

The dataset used for this project contains over **60 million flight records**, covering details such as flight dates, times, delays, cancellations, routes, and carriers. Using **pandas** for preprocessing and **visualization libraries** such as Matplotlib, Seaborn, Plotly, and Folium, we performed a thorough analysis of flight operations.

This document summarizes the data cleaning steps, metrics generated, and insights discovered during the project. It also highlights deliverables that can serve as a foundation for building interactive dashboards and future predictive models.

1. Define Goals, KPIs, and Workflow

At the beginning of the project, it is important to clearly define the purpose and expected outcomes. Goals provide direction, while KPIs (Key Performance Indicators) allow tracking of progress and success. A structured workflow ensures tasks are executed systematically.

Key Steps:

- Identify the business problem and translate it into data-driven goals.
- Define KPIs such as accuracy, precision, recall (if ML project), or revenue growth, customer retention (if business-oriented).
- Outline the project workflow: data collection, preprocessing, analysis, modeling, evaluation, and reporting.

2. Load CSVs using pandas

Datasets are often provided in CSV format, which can be easily handled with Python's pandas library. The first step after receiving the dataset is to load it into memory for exploration.

Example Code:

```
import pandas as pd
df = pd.read_csv('dataset.csv')
print(df.head())
```

Here, `read_csv` loads the data into a DataFrame, and `head()` shows the first few rows to quickly inspect the dataset.

3. Explore Schema, Types, Size, and Nulls

After loading the data, exploration helps in understanding its structure and quality. The following aspects are examined:

- **Schema:** Check all column names and their purpose.
- **Data Types:** Identify if columns are integers, floats, objects, or dates.
- **Dataset Size:** Determine the number of rows and columns to understand scale.
- **Missing Values:** Identify null values that may require imputation or removal.

Useful Commands:

```
df.info()    # Overview of schema and data types
df.describe() # Statistical summary of numerical data
df.isnull().sum() # Count of missing values per column
```

4. Perform Sampling and Memory Optimizations

Large datasets can be computationally expensive to process. To improve efficiency, two key approaches are applied: sampling and memory optimization.

Sampling:

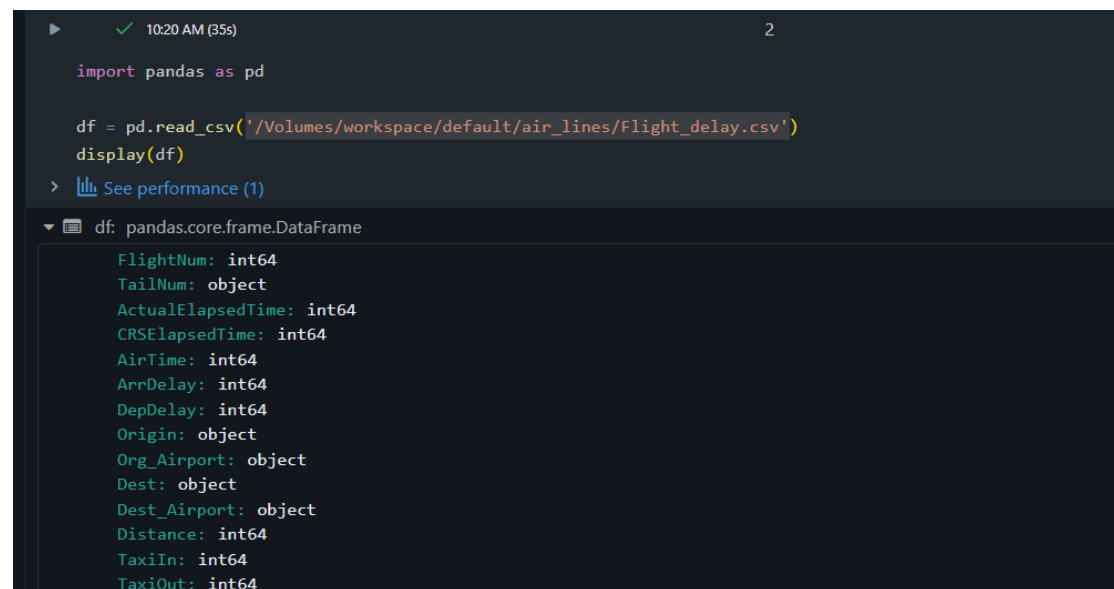
- Selects a subset of the dataset for faster testing and prototyping.
- Example: `df.sample(1000)` retrieves 1000 random rows.

Memory Optimizations:

- Convert object/string columns with limited categories to 'category' type.
- Downcast numerical columns (e.g., float64 to float32) to reduce memory usage.
- Drop unnecessary columns that do not contribute to the analysis.

These techniques help ensure the dataset is manageable and efficient to work with, especially in environments with limited computational resources.

Implementation:



```
import pandas as pd

df = pd.read_csv('/Volumes/workspace/default/air_lines/Flight_delay.csv')
display(df)
```

> [See performance \(1\)](#)

df: pandas.core.frame.DataFrame

```
FlightNum: int64
TailNum: object
ActualElapsedTime: int64
CRSElapsedTime: int64
AirTime: int64
ArrDelay: int64
DepDelay: int64
Origin: object
Org_Airport: object
Dest: object
Dest_Airport: object
Distance: int64
TaxiIn: int64
TaxiOut: int64
```

