Data Analytics on Indigo Airlines’ Business Model

Index

Description

Project Title: Data Analytics on Indigo Business Model
This project aims to analyse the operational strategies and performance of Indigo Airlines using real-world aviation data. By studying air traffic patterns, sector-wise demand, and market share metrics, the goal is to gain a deeper understanding of the factors contributing to Indigo's dominance in the Indian aviation industry.

Data & Problem Context

Problem Statement:
Indigo Airlines holds the largest market share in India’s aviation sector. To

sustain and strengthen this position, it is crucial to understand the factors driving its success. This project addresses:

- Top-performing air travel sectors in India
- Sectors with increasing future demand
- Airlines likely to grow based on market share
- Impact of delays on operations
- Data-driven insights into Indigo's new route decisions

Purpose & Outcome:
To create an interactive, modular data analytics app capable of delivering visual insights into key aviation metrics. The goal is to support business decisions using data.

Benefits:

- Improved sector targeting
- Demand forecasting for route expansion
- Competitive benchmarking among airlines
- GUI-based visualization for enhanced user interaction

Plan

- Gather relevant datasets from DGCA & Kaggle
- Preprocess and clean the data
- Analyze traffic and market trends
- Build GUI for user interaction
- Generate outputs for each major insight
- Add delay pattern module for extended analysis

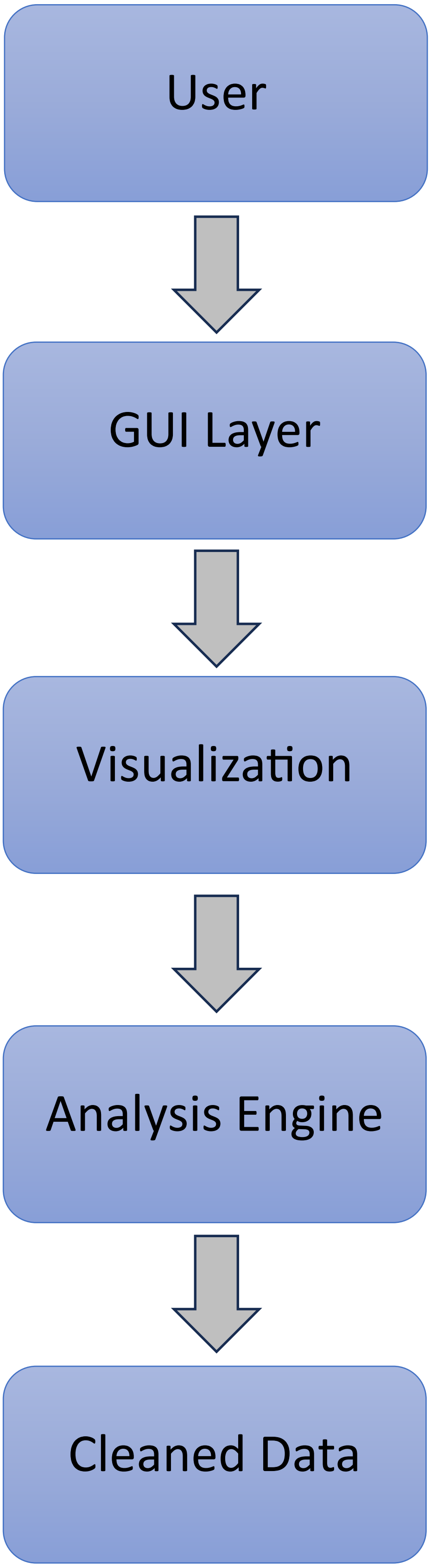
Design

Modular Structure:

- **DataLoader Module:** Reads and cleans raw data

- **Analysis Module:** Core analytics (sectors, trends, shares)
- **Visualization Module:** Plot diagrams via matplotlib
- **GUI Module:** User interface via Tkinter

Data Flow Diagram (DFD)



Implementation

Technologies Used:

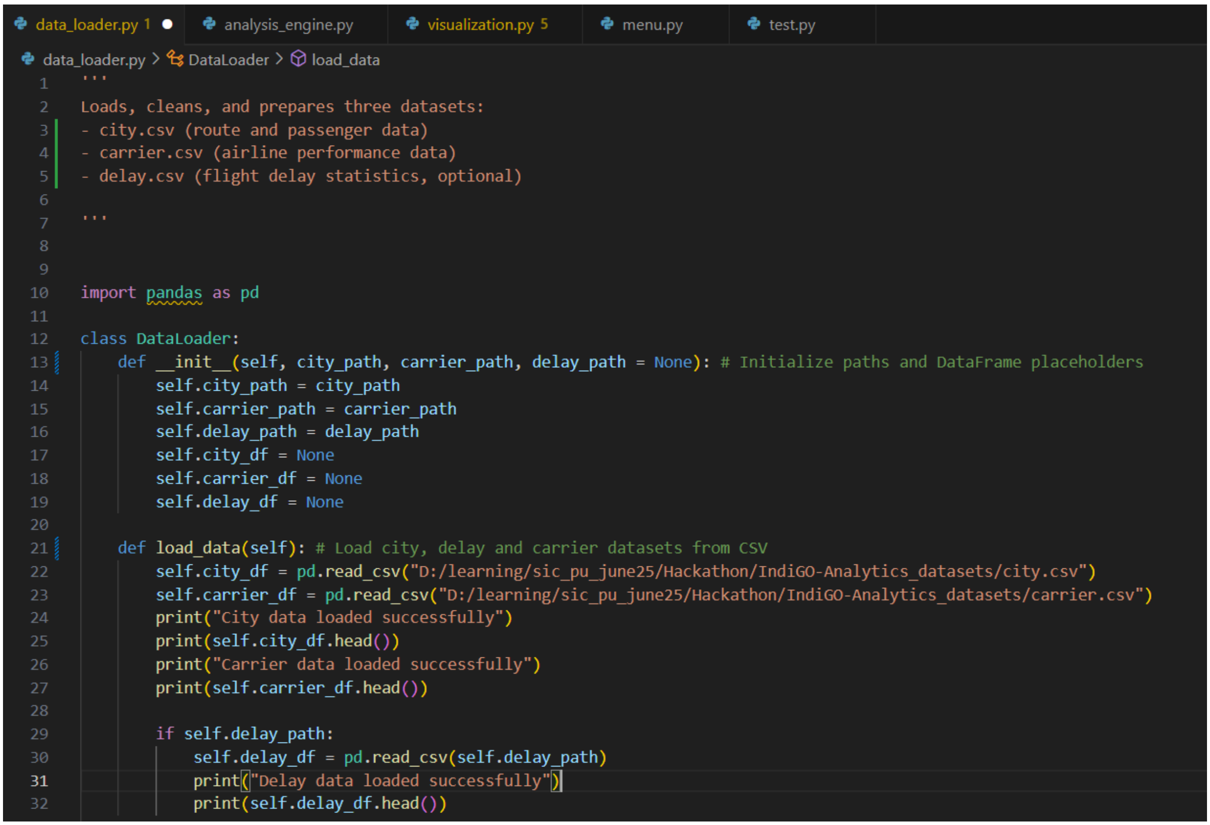
- **Language:** Python
- **Libraries:** pandas, numpy, matplotlib, seaborn, tkinter
- **Paradigm:** Object-Oriented Programming
- **Data Sources:**
    - Directorate General of Civil Aviation (DGCA)
    - Kaggle (open-source datasets)

Features:

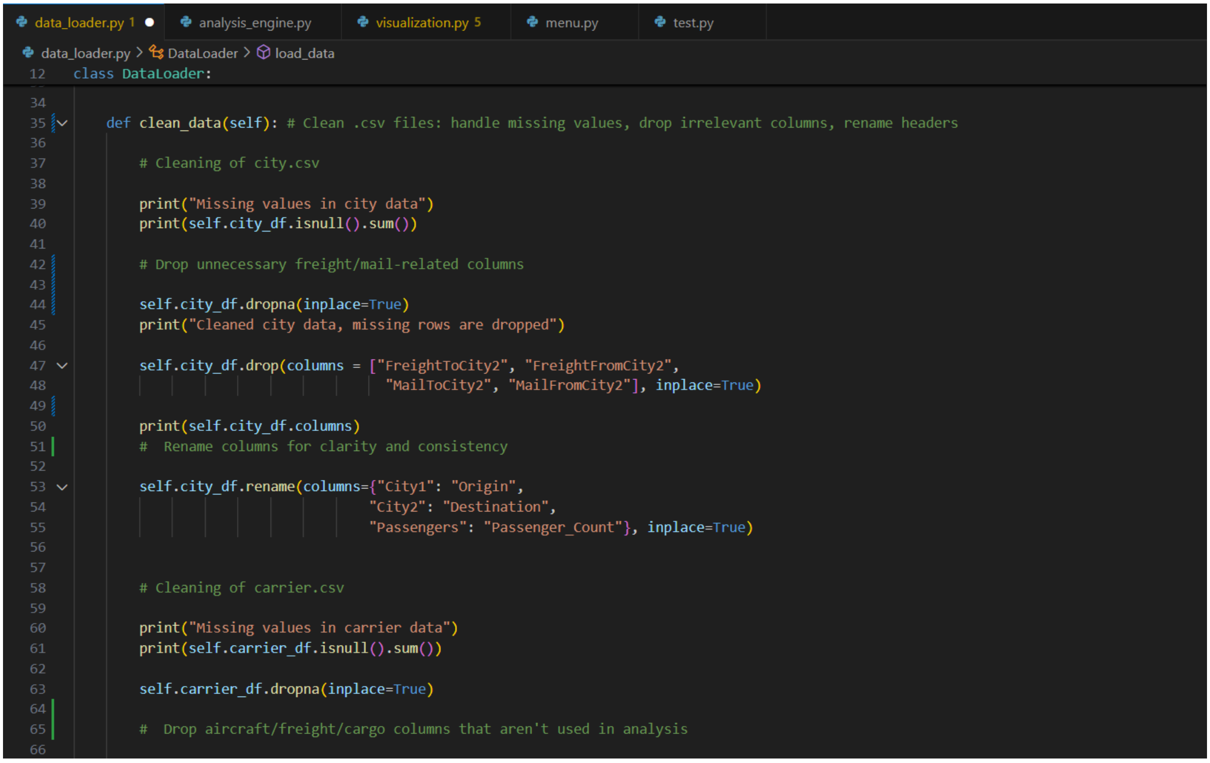
- Sector-wise traffic analysis
- Forecast model for demand growth
- Airline market share charts
- Delay analysis for punctuality insights
- Tkinter-based interactive GUI

Code with Explanation

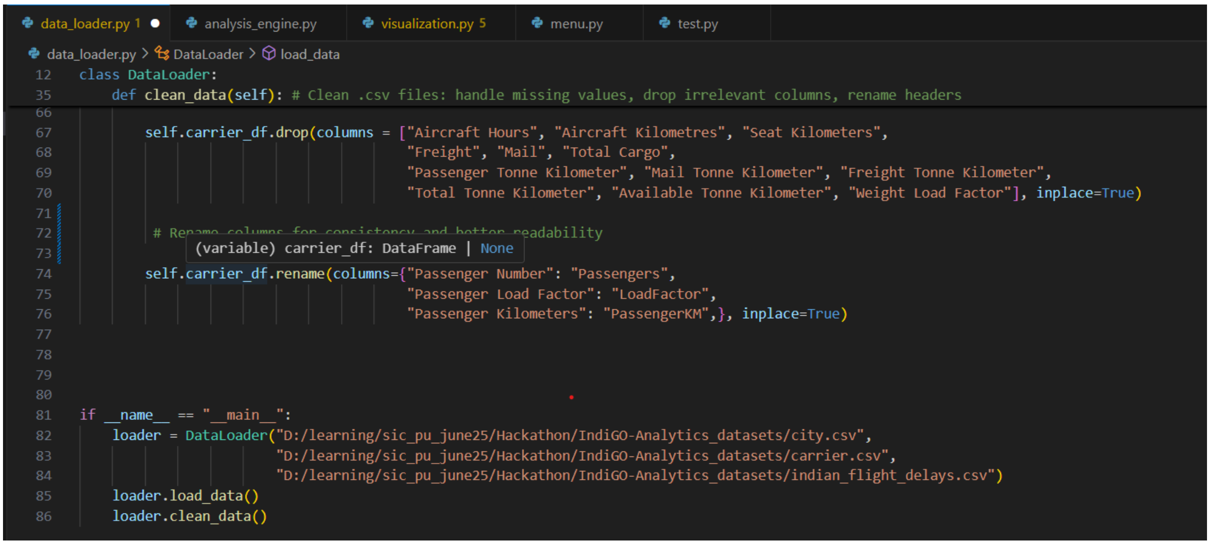
DataLoader Code:



```python
'''
Loads, cleans, and prepares three datasets:
- city.csv (route and passenger data)
- carrier.csv (airline performance data)
- delay.csv (flight delay statistics, optional)

'''


import pandas as pd

class DataLoader:
    def __init__(self, city_path, carrier_path, delay_path = None): # Initialize paths and DataFrame placeholders
        self.city_path = city_path
        self.carrier_path = carrier_path
        self.delay_path = delay_path
        self.city_df = None
        self.carrier_df = None
        self.delay_df = None

    def load_data(self): # Load city, delay and carrier datasets from CSV
        self.city_df = pd.read_csv("D:/learning/sic_pu_june25/Hackathon/IndiGO-Analytics_datasets/city.csv")
        self.carrier_df = pd.read_csv("D:/learning/sic_pu_june25/Hackathon/IndiGO-Analytics_datasets/carrier.csv")
        print("City data loaded successfully")
        print(self.city_df.head())
        print("Carrier data loaded successfully")
        print(self.carrier_df.head())

        if self.delay_path:
            self.delay_df = pd.read_csv(self.delay_path)
            print("Delay data loaded successfully")
            print(self.delay_df.head())
```

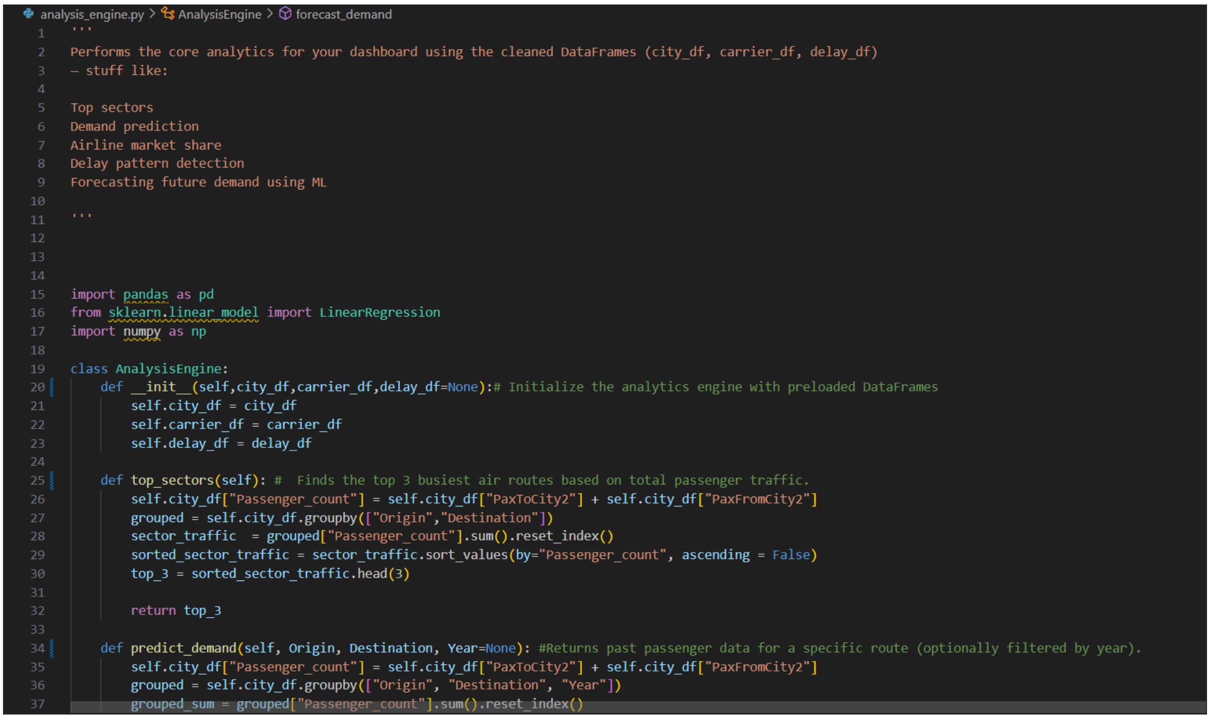


```python
class DataLoader:

    def clean_data(self): # Clean .csv files: handle missing values, drop irrelevant columns, rename headers

        # Cleaning of city.csv

        print("Missing values in city data")
        print(self.city_df.isnull().sum())

        # Drop unnecessary freight/mail-related columns

        self.city_df.dropna(inplace=True)
        print("Cleaned city data, missing rows are dropped")

        self.city_df.drop(columns = ["FreightToCity2", "FreightFromCity2",
                                     "MailToCity2", "MailFromCity2"], inplace=True)

        print(self.city_df.columns)
        #  Rename columns for clarity and consistency

        self.city_df.rename(columns={"City1": "Origin",
                                     "City2": "Destination",
                                     "Passengers": "Passenger_Count"}, inplace=True)


        # Cleaning of carrier.csv

        print("Missing values in carrier data")
        print(self.carrier_df.isnull().sum())

        self.carrier_df.dropna(inplace=True)

        #  Drop aircraft/freight/cargo columns that aren't used in analysis

```

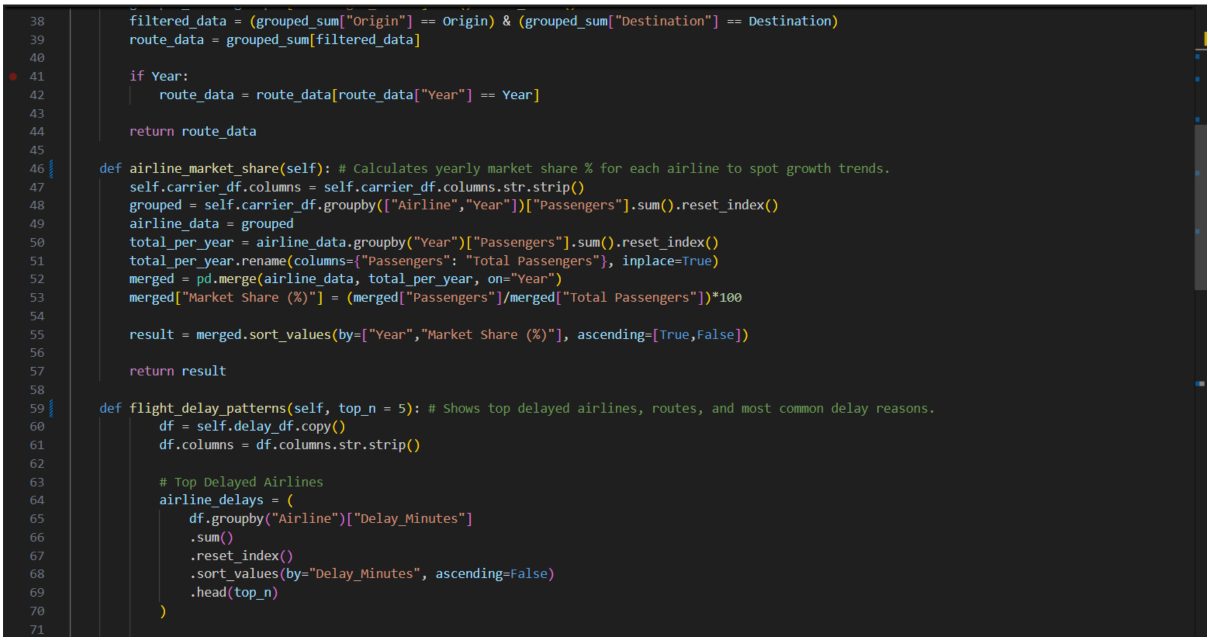

data_loader.py 1
analysis_engine.py
visualization.py 5
menu.py
test.py
data_loader.py > DataLoader > load_data
(variable) carrier_df: DataFrame | None

```python
class DataLoader:
    def clean_data(self): # Clean .csv files: handle missing values, drop irrelevant columns, rename headers

        self.carrier_df.drop(columns = ["Aircraft Hours", "Aircraft Kilometres", "Seat Kilometers",
                                        "Freight", "Mail", "Total Cargo",
                                        "Passenger Tonne Kilometer", "Mail Tonne Kilometer", "Freight Tonne Kilometer",
                                        "Total Tonne Kilometer", "Available Tonne Kilometer", "Weight Load Factor"], inplace=True)

         # Rename columns for consistency and better readability

        self.carrier_df.rename(columns={"Passenger Number": "Passengers",
                                        "Passenger Load Factor": "LoadFactor",
                                        "Passenger Kilometers": "PassengerKM",}, inplace=True)




if __name__ == "__main__":
    loader = DataLoader("D:/learning/sic_pu_june25/Hackathon/IndiGO-Analytics_datasets/city.csv",
                        "D:/learning/sic_pu_june25/Hackathon/IndiGO-Analytics_datasets/carrier.csv",
                        "D:/learning/sic_pu_june25/Hackathon/IndiGO-Analytics_datasets/indian_flight_delays.csv")
    loader.load_data()
    loader.clean_data()
```

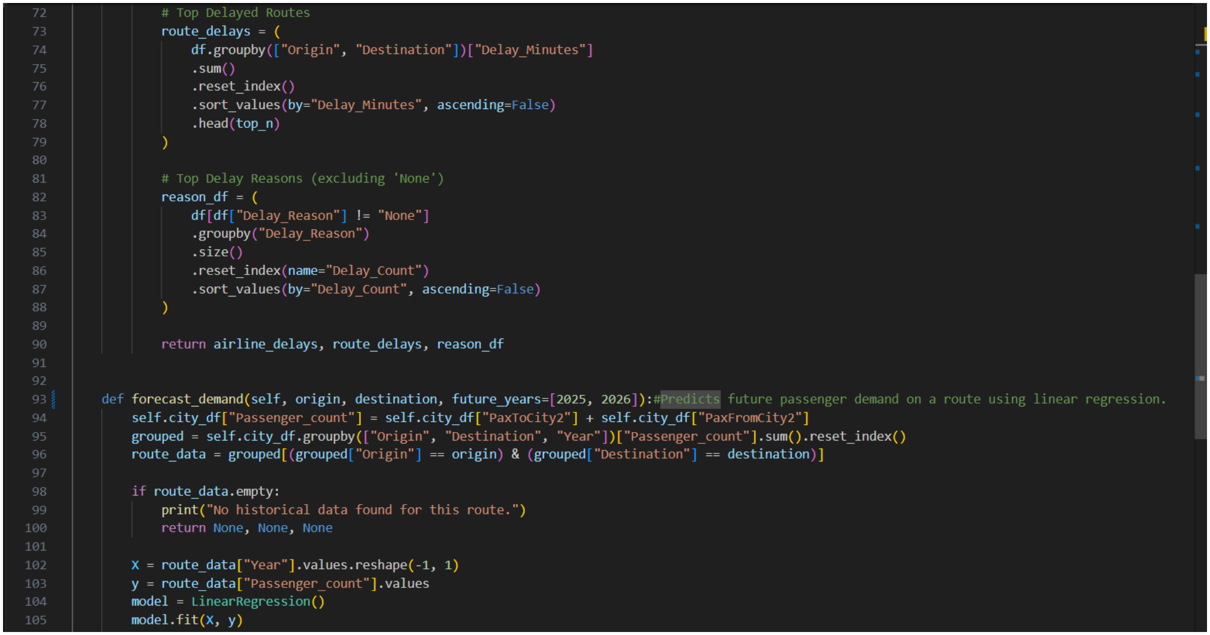
Analysis Code:



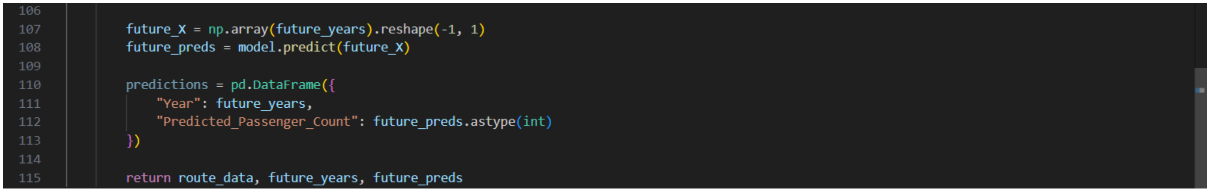
```python
'''
Performs the core analytics for your dashboard using the cleaned DataFrames (city_df, carrier_df, delay_df)
– stuff like:

Top sectors
Demand prediction
Airline market share
Delay pattern detection
Forecasting future demand using ML

'''



import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

class AnalysisEngine:
    def __init__(self,city_df,carrier_df,delay_df=None):# Initialize the analytics engine with preloaded DataFrames
        self.city_df = city_df
        self.carrier_df = carrier_df
        self.delay_df = delay_df

    def top_sectors(self): #  Finds the top 3 busiest air routes based on total passenger traffic.
        self.city_df["Passenger_count"] = self.city_df["PaxToCity2"] + self.city_df["PaxFromCity2"]
        grouped = self.city_df.groupby(["Origin","Destination"])
        sector_traffic  = grouped["Passenger_count"].sum().reset_index()
        sorted_sector_traffic = sector_traffic.sort_values(by="Passenger_count", ascending = False)
        top_3 = sorted_sector_traffic.head(3)

        return top_3

    def predict_demand(self, Origin, Destination, Year=None): #Returns past passenger data for a specific route (optionally filtered by year).
        self.city_df["Passenger_count"] = self.city_df["PaxToCity2"] + self.city_df["PaxFromCity2"]
        grouped = self.city_df.groupby(["Origin", "Destination", "Year"])
        grouped_sum = grouped["Passenger_count"].sum().reset_index()
```



```python
        filtered_data = (grouped_sum["Origin"] == Origin) & (grouped_sum["Destination"] == Destination)
        route_data = grouped_sum[filtered_data]

        if Year:
            route_data = route_data[route_data["Year"] == Year]

        return route_data

    def airline_market_share(self): # Calculates yearly market share % for each airline to spot growth trends.
        self.carrier_df.columns = self.carrier_df.columns.str.strip()
        grouped = self.carrier_df.groupby(["Airline","Year"])["Passengers"].sum().reset_index()
        airline_data = grouped
        total_per_year = airline_data.groupby("Year")["Passengers"].sum().reset_index()
        total_per_year.rename(columns={"Passengers": "Total Passengers"}, inplace=True)
        merged = pd.merge(airline_data, total_per_year, on="Year")
        merged["Market Share (%)"] = (merged["Passengers"]/merged["Total Passengers"])*100

        result = merged.sort_values(by=["Year","Market Share (%)"], ascending=[True,False])

        return result

    def flight_delay_patterns(self, top_n = 5): # Shows top delayed airlines, routes, and most common delay reasons.
            df = self.delay_df.copy()
            df.columns = df.columns.str.strip()

            # Top Delayed Airlines
            airline_delays = (
                df.groupby("Airline")["Delay_Minutes"]
                .sum()
                .reset_index()
                .sort_values(by="Delay_Minutes", ascending=False)
                .head(top_n)
            )

```

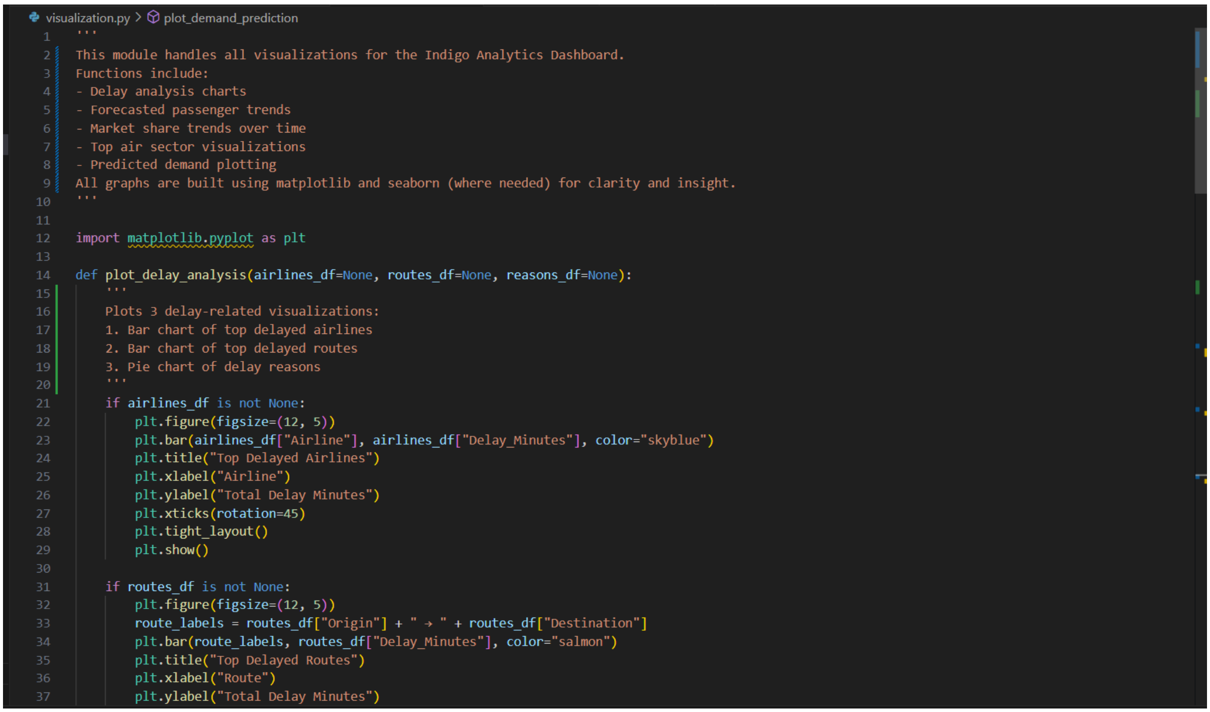


```python
            # Top Delayed Routes
            route_delays = (
                df.groupby(["Origin", "Destination"])["Delay_Minutes"]
                .sum()
                .reset_index()
                .sort_values(by="Delay_Minutes", ascending=False)
                .head(top_n)
            )

            # Top Delay Reasons (excluding 'None')
            reason_df = (
                df[df["Delay_Reason"] != "None"]
                .groupby("Delay_Reason")
                .size()
                .reset_index(name="Delay_Count")
                .sort_values(by="Delay_Count", ascending=False)
            )

            return airline_delays, route_delays, reason_df


    def forecast_demand(self, origin, destination, future_years=[2025, 2026]):#Predicts future passenger demand on a route using linear regression.
        self.city_df["Passenger_count"] = self.city_df["PaxToCity2"] + self.city_df["PaxFromCity2"]
        grouped = self.city_df.groupby(["Origin", "Destination", "Year"])["Passenger_count"].sum().reset_index()
        route_data = grouped[(grouped["Origin"] == origin) & (grouped["Destination"] == destination)]

        if route_data.empty:
            print("No historical data found for this route.")
            return None, None, None

        X = route_data["Year"].values.reshape(-1, 1)
        y = route_data["Passenger_count"].values
        model = LinearRegression()
        model.fit(X, y)
```

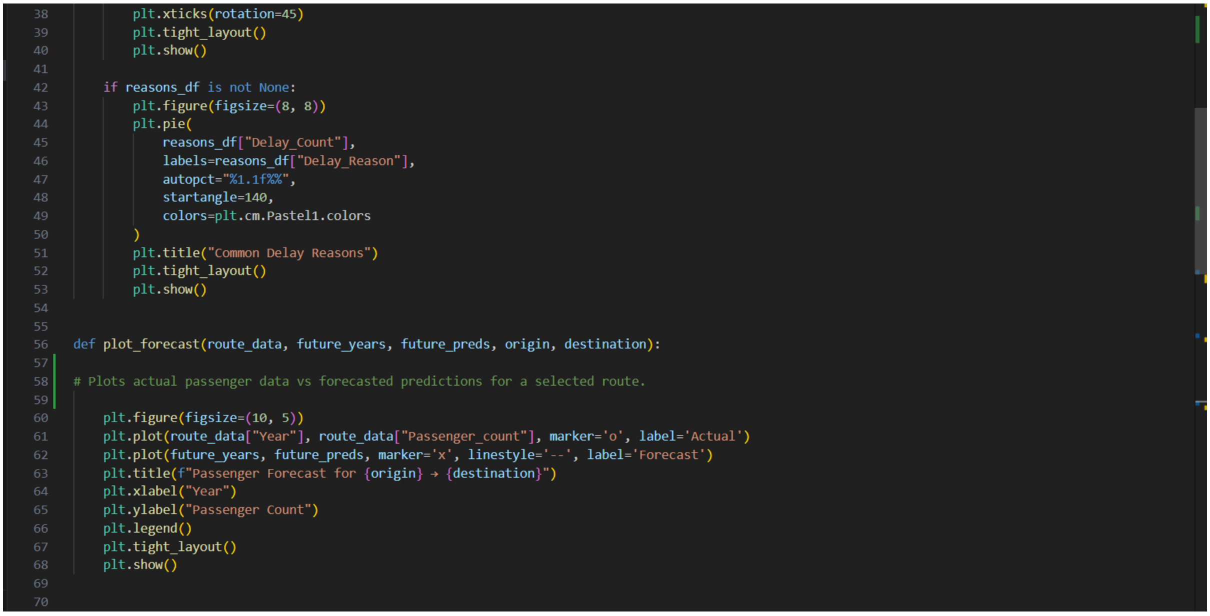


```python

        future_X = np.array(future_years).reshape(-1, 1)
        future_preds = model.predict(future_X)

        predictions = pd.DataFrame({
            "Year": future_years,
            "Predicted_Passenger_Count": future_preds.astype(int)
        })

        return route_data, future_years, future_preds
```

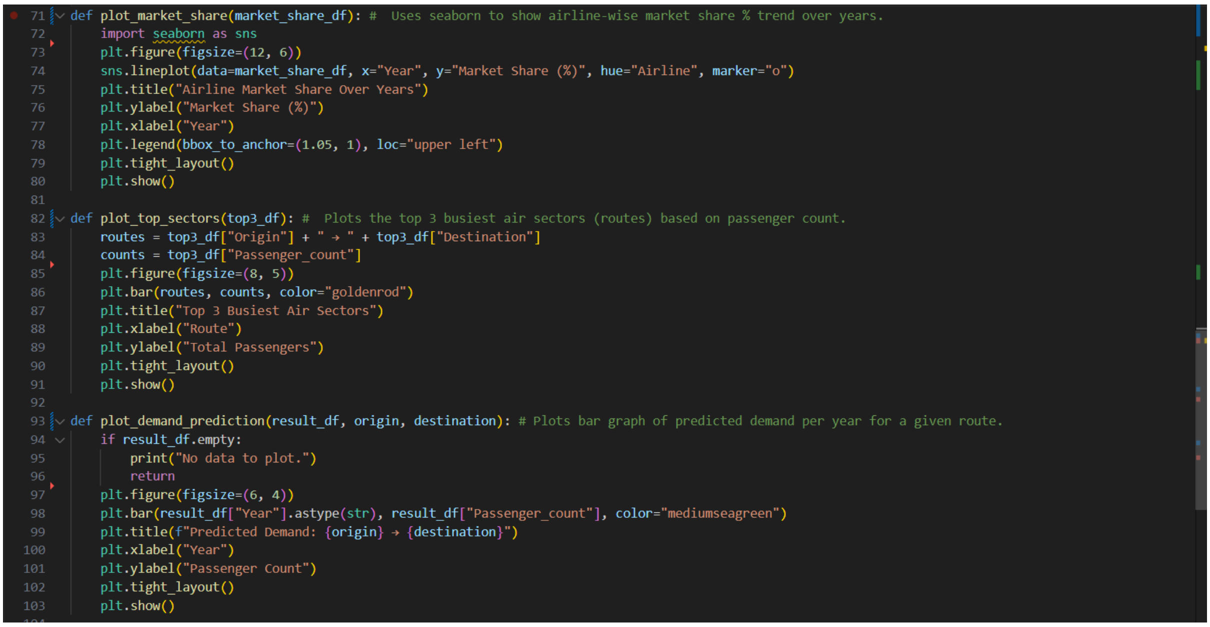
Visualization Code:



```python
'''
This module handles all visualizations for the Indigo Analytics Dashboard.
Functions include:
- Delay analysis charts
- Forecasted passenger trends
- Market share trends over time
- Top air sector visualizations
- Predicted demand plotting
All graphs are built using matplotlib and seaborn (where needed) for clarity and insight.
'''

import matplotlib.pyplot as plt

def plot_delay_analysis(airlines_df=None, routes_df=None, reasons_df=None):
    '''
    Plots 3 delay-related visualizations:
    1. Bar chart of top delayed airlines
    2. Bar chart of top delayed routes
    3. Pie chart of delay reasons
    '''
    if airlines_df is not None:
        plt.figure(figsize=(12, 5))
        plt.bar(airlines_df["Airline"], airlines_df["Delay_Minutes"], color="skyblue")
        plt.title("Top Delayed Airlines")
        plt.xlabel("Airline")
        plt.ylabel("Total Delay Minutes")
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

    if routes_df is not None:
        plt.figure(figsize=(12, 5))
        route_labels = routes_df["Origin"] + " → " + routes_df["Destination"]
        plt.bar(route_labels, routes_df["Delay_Minutes"], color="salmon")
        plt.title("Top Delayed Routes")
        plt.xlabel("Route")
        plt.ylabel("Total Delay Minutes")
```

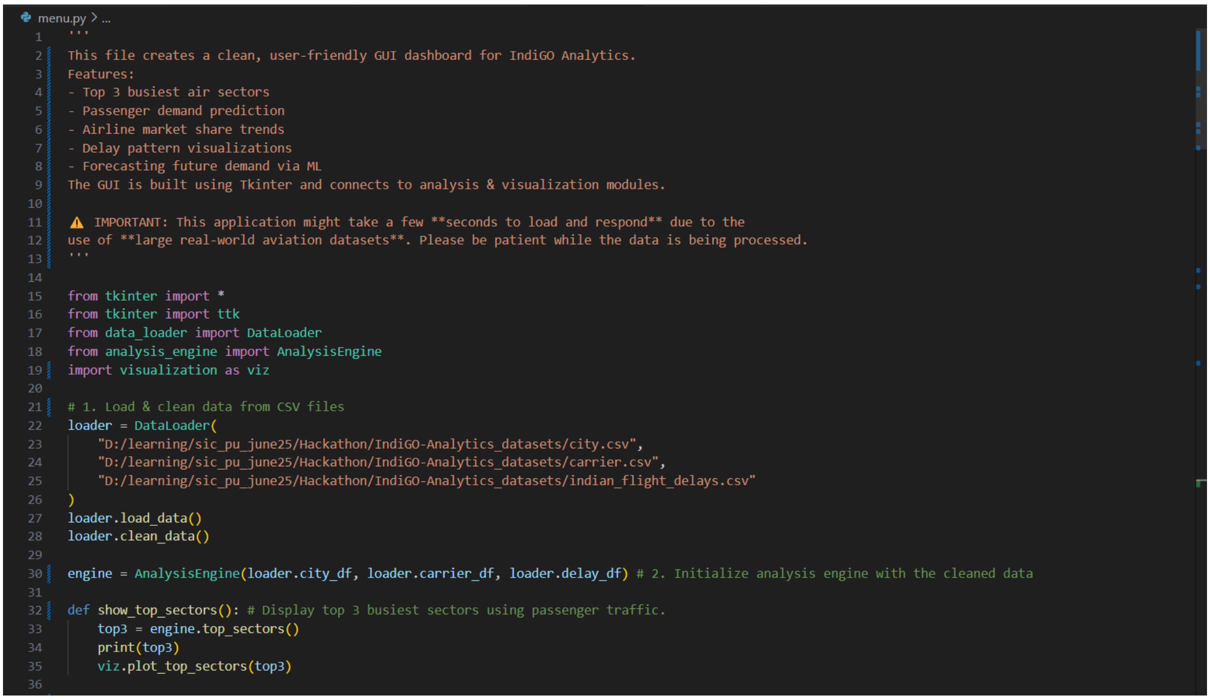


```python
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

    if reasons_df is not None:
        plt.figure(figsize=(8, 8))
        plt.pie(
            reasons_df["Delay_Count"],
            labels=reasons_df["Delay_Reason"],
            autopct="%1.1f%%",
            startangle=140,
            colors=plt.cm.Pastel1.colors
        )
        plt.title("Common Delay Reasons")
        plt.tight_layout()
        plt.show()


def plot_forecast(route_data, future_years, future_preds, origin, destination):

# Plots actual passenger data vs forecasted predictions for a selected route.

    plt.figure(figsize=(10, 5))
    plt.plot(route_data["Year"], route_data["Passenger_count"], marker='o', label='Actual')
    plt.plot(future_years, future_preds, marker='x', linestyle='--', label='Forecast')
    plt.title(f"Passenger Forecast for {origin} → {destination}")
    plt.xlabel("Year")
    plt.ylabel("Passenger Count")
    plt.legend()
    plt.tight_layout()
    plt.show()
```

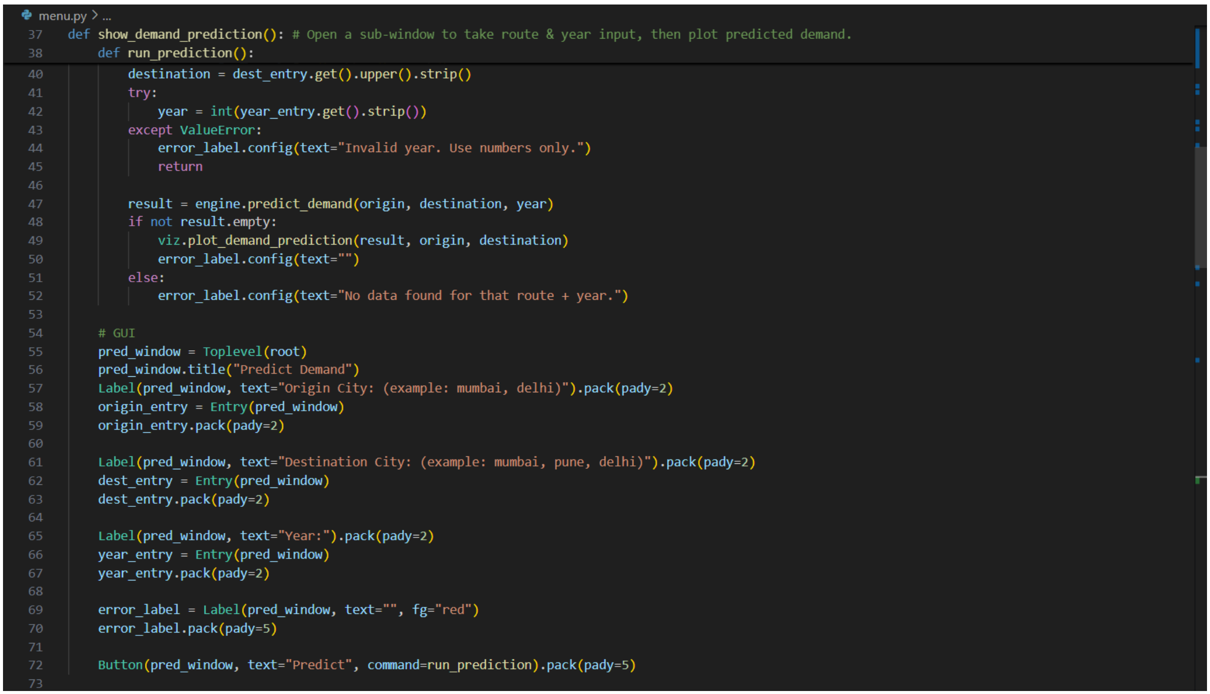


```python
def plot_market_share(market_share_df): #  Uses seaborn to show airline-wise market share % trend over years.
    import seaborn as sns
    plt.figure(figsize=(12, 6))
    sns.lineplot(data=market_share_df, x="Year", y="Market Share (%)", hue="Airline", marker="o")
    plt.title("Airline Market Share Over Years")
    plt.ylabel("Market Share (%)")
    plt.xlabel("Year")
    plt.legend(bbox_to_anchor=(1.05, 1), loc="upper left")
    plt.tight_layout()
    plt.show()

def plot_top_sectors(top3_df): #  Plots the top 3 busiest air sectors (routes) based on passenger count.
    routes = top3_df["Origin"] + " → " + top3_df["Destination"]
    counts = top3_df["Passenger_count"]
    plt.figure(figsize=(8, 5))
    plt.bar(routes, counts, color="goldenrod")
    plt.title("Top 3 Busiest Air Sectors")
    plt.xlabel("Route")
    plt.ylabel("Total Passengers")
    plt.tight_layout()
    plt.show()

def plot_demand_prediction(result_df, origin, destination): # Plots bar graph of predicted demand per year for a given route.
    if result_df.empty:
        print("No data to plot.")
        return
    plt.figure(figsize=(6, 4))
    plt.bar(result_df["Year"].astype(str), result_df["Passenger_count"], color="mediumseagreen")
    plt.title(f"Predicted Demand: {origin} → {destination}")
    plt.xlabel("Year")
    plt.ylabel("Passenger Count")
    plt.tight_layout()
    plt.show()
```

Menu Code:

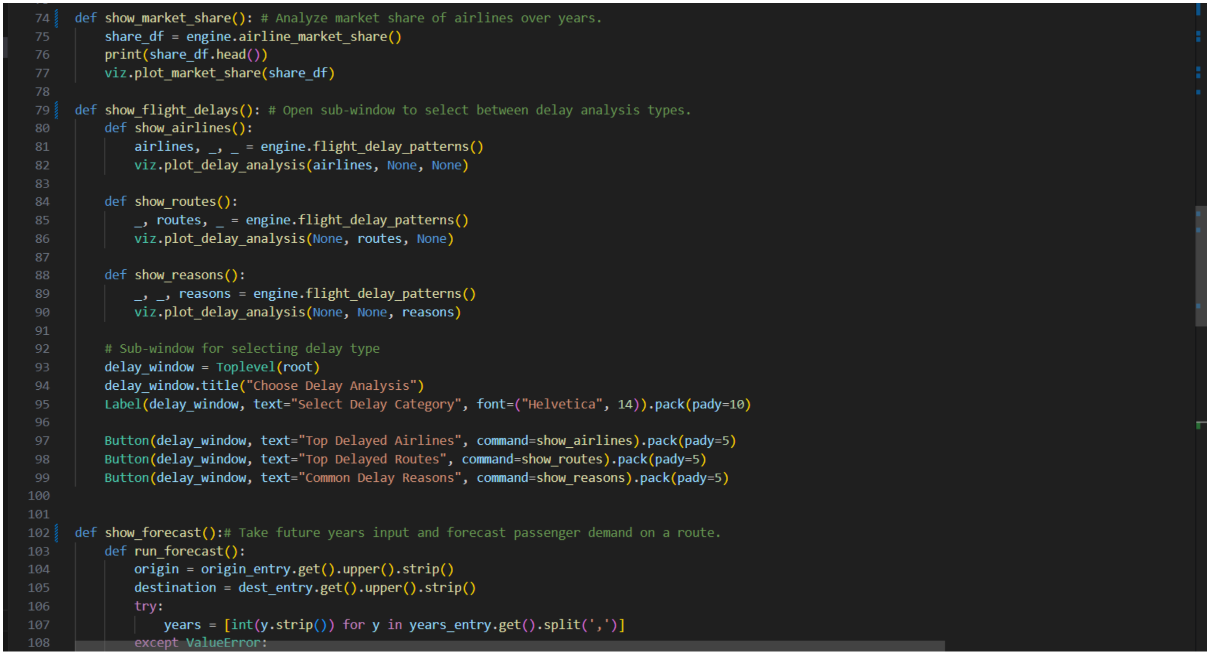


```python
'''
This file creates a clean, user-friendly GUI dashboard for IndiGO Analytics.
Features:
- Top 3 busiest air sectors
- Passenger demand prediction
- Airline market share trends
- Delay pattern visualizations
- Forecasting future demand via ML
The GUI is built using Tkinter and connects to analysis & visualization modules.

⚠ IMPORTANT: This application might take a few **seconds to load and respond** due to the
use of **large real-world aviation datasets**. Please be patient while the data is being processed.
'''

from tkinter import *
from tkinter import ttk
from data_loader import DataLoader
from analysis_engine import AnalysisEngine
import visualization as viz

# 1. Load & clean data from CSV files
loader = DataLoader(
    "D:/learning/sic_pu_june25/Hackathon/IndiGO-Analytics_datasets/city.csv",
    "D:/learning/sic_pu_june25/Hackathon/IndiGO-Analytics_datasets/carrier.csv",
    "D:/learning/sic_pu_june25/Hackathon/IndiGO-Analytics_datasets/indian_flight_delays.csv"
)
loader.load_data()
loader.clean_data()

engine = AnalysisEngine(loader.city_df, loader.carrier_df, loader.delay_df) # 2. Initialize analysis engine with the cleaned data

def show_top_sectors(): # Display top 3 busiest sectors using passenger traffic.
    top3 = engine.top_sectors()
    print(top3)
    viz.plot_top_sectors(top3)

```

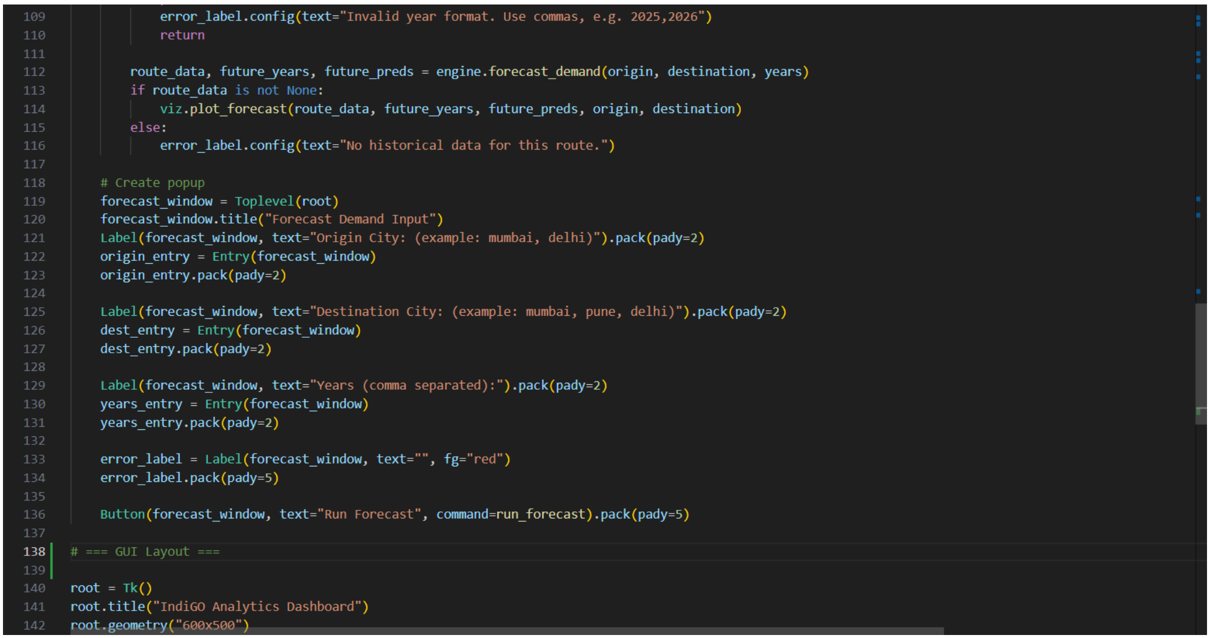


menu.py > ...

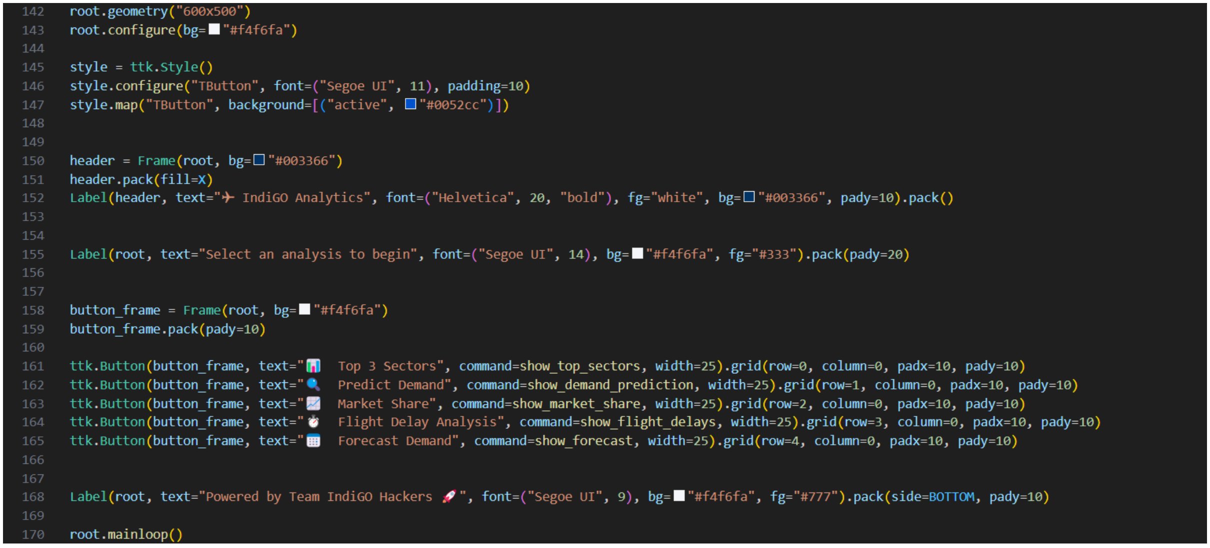
```python
def show_demand_prediction(): # Open a sub-window to take route & year input, then plot predicted demand.
    def run_prediction():
        destination = dest_entry.get().upper().strip()
        try:
            year = int(year_entry.get().strip())
        except ValueError:
            error_label.config(text="Invalid year. Use numbers only.")
            return

        result = engine.predict_demand(origin, destination, year)
        if not result.empty:
            viz.plot_demand_prediction(result, origin, destination)
            error_label.config(text="")
        else:
            error_label.config(text="No data found for that route + year.")

    # GUI
    pred_window = Toplevel(root)
    pred_window.title("Predict Demand")
    Label(pred_window, text="Origin City: (example: mumbai, delhi)").pack(pady=2)
    origin_entry = Entry(pred_window)
    origin_entry.pack(pady=2)

    Label(pred_window, text="Destination City: (example: mumbai, pune, delhi)").pack(pady=2)
    dest_entry = Entry(pred_window)
    dest_entry.pack(pady=2)

    Label(pred_window, text="Year:").pack(pady=2)
    year_entry = Entry(pred_window)
    year_entry.pack(pady=2)

    error_label = Label(pred_window, text="", fg="red")
    error_label.pack(pady=5)

    Button(pred_window, text="Predict", command=run_prediction).pack(pady=5)

```



```python
def show_market_share(): # Analyze market share of airlines over years.
    share_df = engine.airline_market_share()
    print(share_df.head())
    viz.plot_market_share(share_df)

def show_flight_delays(): # Open sub-window to select between delay analysis types.
    def show_airlines():
        airlines, _, _ = engine.flight_delay_patterns()
        viz.plot_delay_analysis(airlines, None, None)

    def show_routes():
        _, routes, _ = engine.flight_delay_patterns()
        viz.plot_delay_analysis(None, routes, None)

    def show_reasons():
        _, _, reasons = engine.flight_delay_patterns()
        viz.plot_delay_analysis(None, None, reasons)

    # Sub-window for selecting delay type
    delay_window = Toplevel(root)
    delay_window.title("Choose Delay Analysis")
    Label(delay_window, text="Select Delay Category", font=("Helvetica", 14)).pack(pady=10)

    Button(delay_window, text="Top Delayed Airlines", command=show_airlines).pack(pady=5)
    Button(delay_window, text="Top Delayed Routes", command=show_routes).pack(pady=5)
    Button(delay_window, text="Common Delay Reasons", command=show_reasons).pack(pady=5)


def show_forecast():# Take future years input and forecast passenger demand on a route.
    def run_forecast():
        origin = origin_entry.get().upper().strip()
        destination = dest_entry.get().upper().strip()
        try:
            years = [int(y.strip()) for y in years_entry.get().split(',')]
        except ValueError:
```

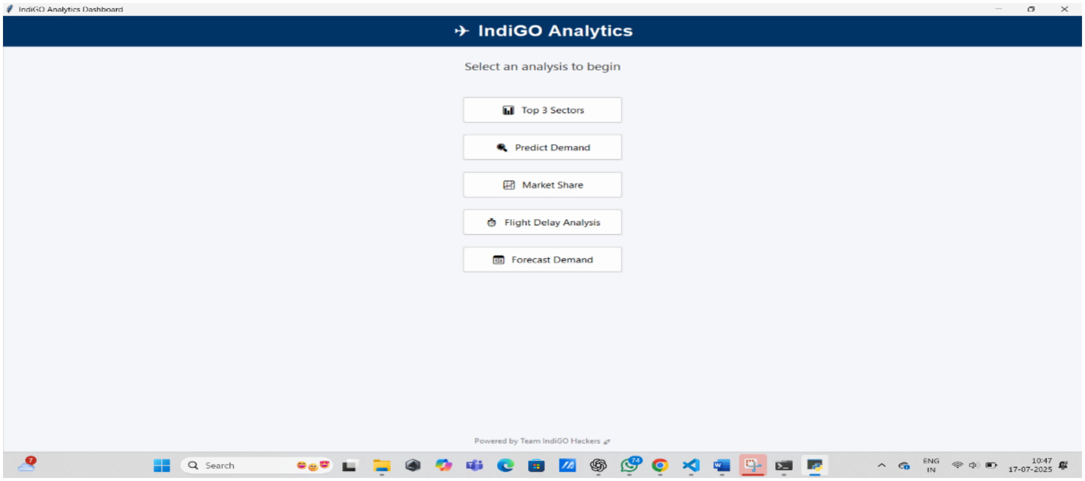


```python
            error_label.config(text="Invalid year format. Use commas, e.g. 2025,2026")
            return

        route_data, future_years, future_preds = engine.forecast_demand(origin, destination, years)
        if route_data is not None:
            viz.plot_forecast(route_data, future_years, future_preds, origin, destination)
        else:
            error_label.config(text="No historical data for this route.")

    # Create popup
    forecast_window = Toplevel(root)
    forecast_window.title("Forecast Demand Input")
    Label(forecast_window, text="Origin City: (example: mumbai, delhi)").pack(pady=2)
    origin_entry = Entry(forecast_window)
    origin_entry.pack(pady=2)

    Label(forecast_window, text="Destination City: (example: mumbai, pune, delhi)").pack(pady=2)
    dest_entry = Entry(forecast_window)
    dest_entry.pack(pady=2)

    Label(forecast_window, text="Years (comma separated):").pack(pady=2)
    years_entry = Entry(forecast_window)
    years_entry.pack(pady=2)

    error_label = Label(forecast_window, text="", fg="red")
    error_label.pack(pady=5)

    Button(forecast_window, text="Run Forecast", command=run_forecast).pack(pady=5)

# === GUI Layout ===

root = Tk()
root.title("IndiGO Analytics Dashboard")
root.geometry("600x500")
```



```python
root.geometry("600x500")
root.configure(bg="#f4f6fa")

style = ttk.Style()
style.configure("TButton", font=("Segoe UI", 11), padding=10)
style.map("TButton", background=[("active", "#0052cc")])


header = Frame(root, bg="#003366")
header.pack(fill=X)
Label(header, text="✈ IndiGO Analytics", font=("Helvetica", 20, "bold"), fg="white", bg="#003366", pady=10).pack()


Label(root, text="Select an analysis to begin", font=("Segoe UI", 14), bg="#f4f6fa", fg="#333").pack(pady=20)


button_frame = Frame(root, bg="#f4f6fa")
button_frame.pack(pady=10)

ttk.Button(button_frame, text="📊  Top 3 Sectors", command=show_top_sectors, width=25).grid(row=0, column=0, padx=10, pady=10)
ttk.Button(button_frame, text="🔍  Predict Demand", command=show_demand_prediction, width=25).grid(row=1, column=0, padx=10, pady=10)
ttk.Button(button_frame, text="📈  Market Share", command=show_market_share, width=25).grid(row=2, column=0, padx=10, pady=10)
ttk.Button(button_frame, text="⏰  Flight Delay Analysis", command=show_flight_delays, width=25).grid(row=3, column=0, padx=10, pady=10)
ttk.Button(button_frame, text="📅  Forecast Demand", command=show_forecast, width=25).grid(row=4, column=0, padx=10, pady=10)


Label(root, text="Powered by Team IndiGO Hackers 🚀", font=("Segoe UI", 9), bg="#f4f6fa", fg="#777").pack(side=BOTTOM, pady=10)

root.mainloop()
```

Output Screenshots



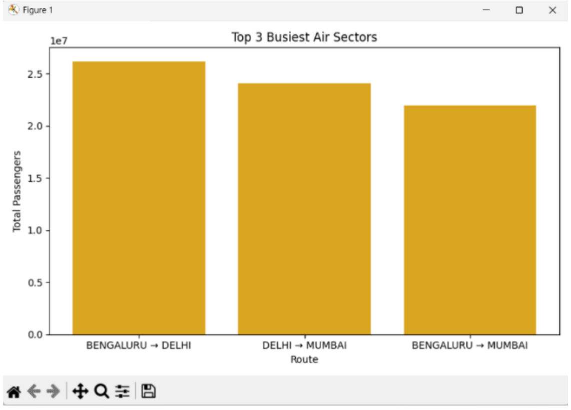


Figure 1
Top 3 Busiest Air Sectors
1e7
2.5
2.0
1.5
1.0
0.5
0.0
Total Passengers
BENGALURU → DELHI
DELHI → MUMBAI
BENGALURU → MUMBAI
Route

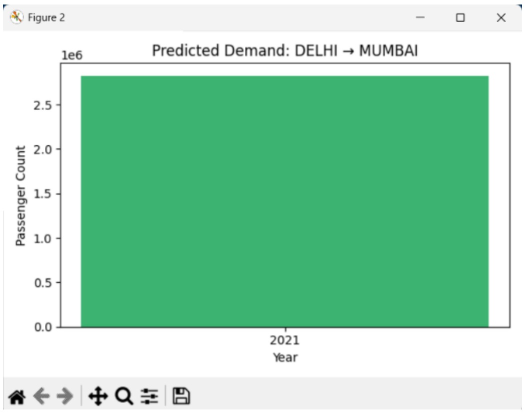


Figure 2
Predicted Demand: DELHI → MUMBAI
1e6
2.5
2.0
1.5
1.0
0.5
0.0
Passenger Count
2021
Year

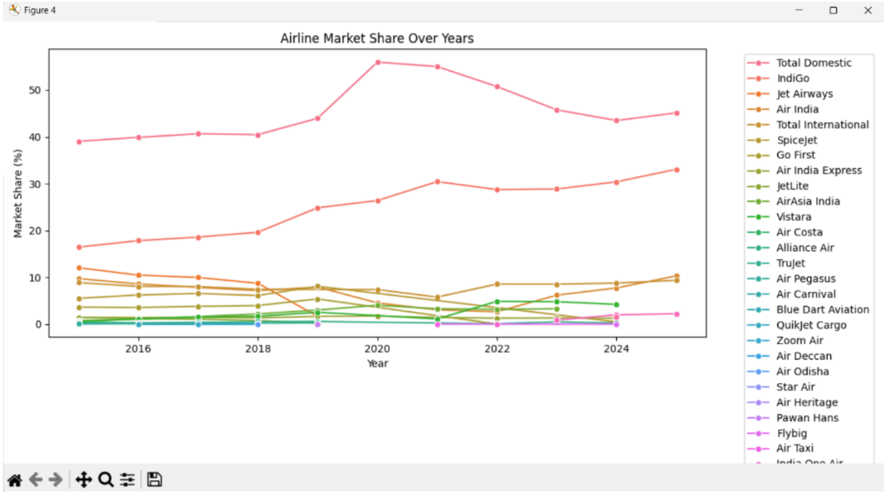


Figure 4
Airline Market Share Over Years
50
40
30
20
10
0
Market Share (%)
2016
2018
2020
2022
2024
Year
Total Domestic
IndiGo
Jet Airways
Air India
Total International
SpiceJet
Go First
Air India Express
JetLite
AirAsia India
Vistara
Air Costa
Alliance Air
TruJet
Air Pegasus
Air Carnival
Blue Dart Aviation
Quikjet Cargo
Zoom Air
Air Deccan
Air Odisha
Star Air
Air Heritage
Pawan Hans
Flybig
Air Taxi
India One Air

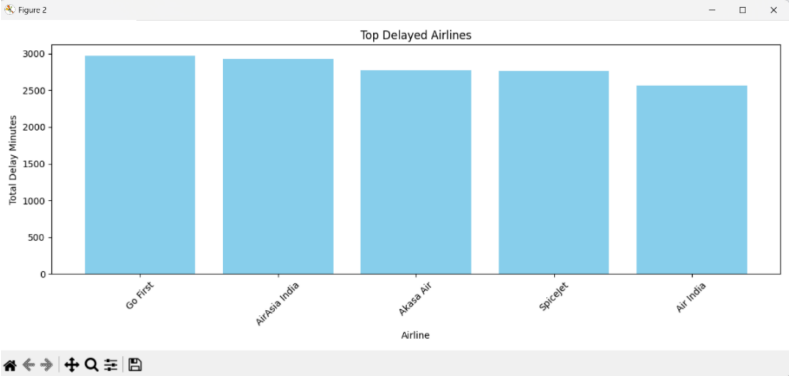


Figure 2
Top Delayed Airlines
3000
2500
2000
1500
1000
500
0
Total Delay Minutes
Go First
AirAsia India
Akasa Air
SpiceJet
Air India
Airline

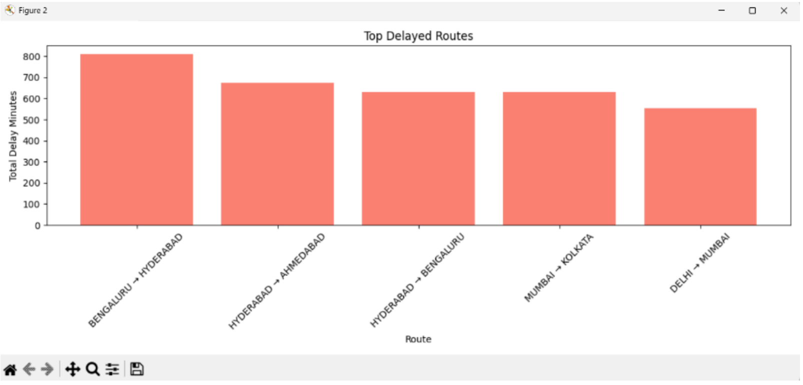
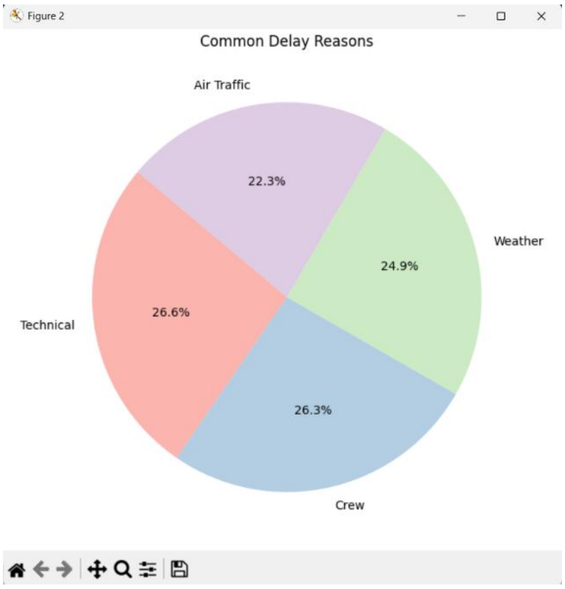
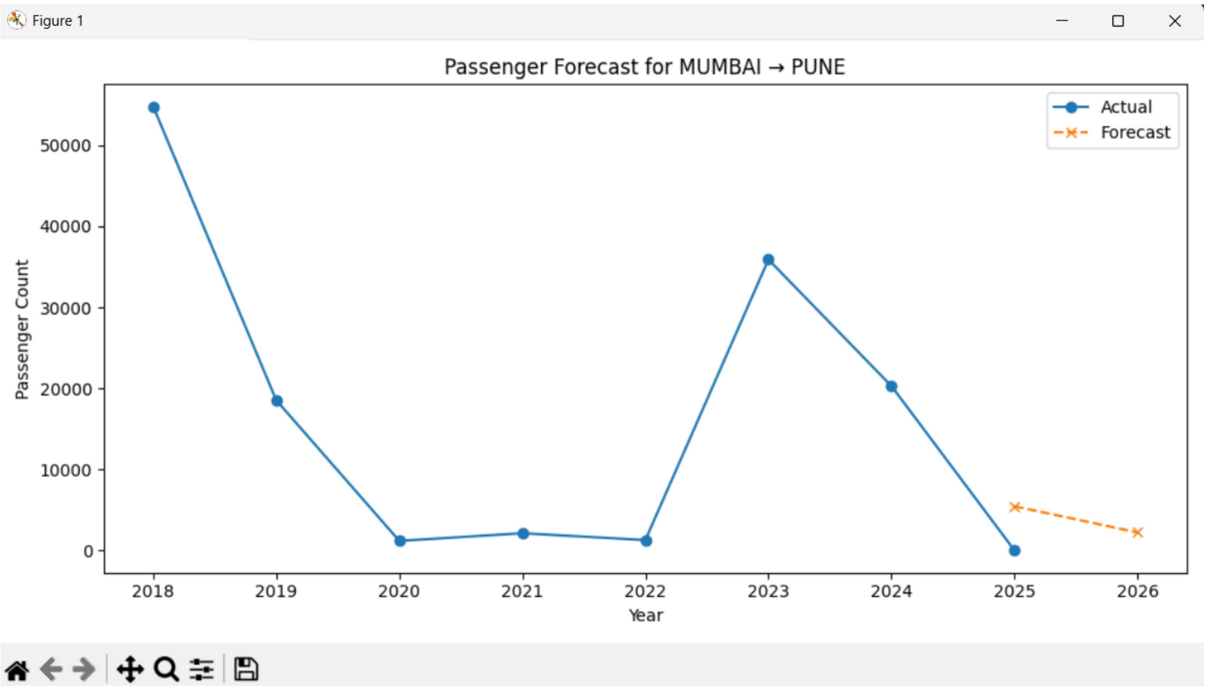


Figure 2
Top Delayed Routes
800
700
600
500
400
300
200
100
0
Total Delay Minutes
BENGALURU → HYDERABAD
HYDERABAD → AHMEDABAD
HYDERABAD → BENGALURU
MUMBAI → KOLKATA
DELHI → MUMBAI
Route





Closure

This project successfully bridges data science and aviation operations by analysing Indigo’s business strategy through an analytical lens. From identifying high-performing sectors to forecasting future demand and mapping market share trends, the application delivers valuable insights. The GUI enables easy access and interaction, making the tool practical for both academic and business use. Future enhancements include live data integration and advanced delay prediction models.

Bibliography

- DGCA Official Portal: https://dgca.gov.in
- Kaggle Aviation Datasets
- Python Docs: https://docs.python.org
- Matplotlib Docs: https://matplotlib.org
- Pandas Docs: https://pandas.pydata.org
- Seaborn Docs: https://seaborn.pydata.org
- Tkinter Docs: https://docs.python.org/3/library/tkinter.html