THE AMERICAN UNIVERSITY OF KURDISTAN

(MSc in AI - Data Mining)

Data Mining Course Project

Sky Predict: Data Mining for Flight Delay & Cancellation

Supervisor Name: **Dr. Shamal AL-Dohuki**

Student Names:

Hariwan Ageed Ahmed **Haval Ali Mustafa** **Sadeeq Ali Sami**

Date of Submission:

Golab Link:
https://colab.research.google.com/drive/1mrtWdzR5MqhHrkQ7IGP8xedkUXF-D6nz?usp=sharing

GitHub Link:

https://github.com/heval1212/Data-Mining-final-project

Dataset Link:

https://www.kaggle.com/datasets/patrickzel/flight-delay-and-cancellation-data-2019-2023-v2

Website Link:

https://f1e08494-0d6c-49f2-9fe6-c033ed207d00.web.createdevserver.com/

Table of Contents

Abstract

Flight delays and cancellations create significant financial losses and customer dissatisfaction for airlines and passengers. In this project we use the Kaggle dataset *"Flight Delay and Cancellation Data (2019–2023) _v2"* by Patrick Zelazko, which aggregates several years of U.S. flight operations data. https://www.kaggle.com/datasets/patrickzel/flight-delay-and-cancellation-data-2019-2023-v2 From this dataset we work with the flights_sample_2m.csv sample (≈2 million records) and build a complete data-mining pipeline and web-based decision-support system called **Sky Predict**.

The data were preprocessed by parsing dates, deriving temporal features (year, month, day-of-week, departure hour), constructing a binary target variable "delayed or cancelled", handling missing values, and encoding categorical attributes such as airline and airports. A subset of key features was selected to keep the model efficient for web deployment.

We then applied several data-mining techniques:

- **Classification:** a Random Forest classifier predicts whether a flight will be delayed by more than 15 minutes or cancelled.
- **Clustering:** K-Means groups flights into on-time, medium-delay, and high-delay clusters.
- **Association rule mining:** the Apriori algorithm discovers patterns between departure period, origin airport and delay severity.
- **Recommendation:** a simple route/airline recommender ranks options with lower historical delay rates.

The final Random Forest model achieves around **87–89% accuracy** and **ROC–AUC ≈ 0.92** on held-out data, with strong precision and recall for delayed flights.

These models power an interactive analytics dashboard and prediction interface that visualize airport performance, seasonal trends, clustering results and airline metrics. The project demonstrates how classical data-mining methods can be combined with a modern web application to support operational planning and passenger decision-making.

1. Introduction

1.1 Problem definition and motivation

Commercial air traffic has grown rapidly in recent years, increasing congestion at major hubs and amplifying the impact of operational disruptions. Flight delays and cancellations lead to missed connections, crew scheduling problems, and substantial economic costs for airlines, airports, and passengers. Previous research estimates that delays cost the U.S. economy billions of dollars annually.

The central problem addressed in this project is:

Can we use historical flight data to analyze patterns of delays and cancellations and to predict the delay risk of an individual flight?

1.2 Importance and Relevance

Accurate delay analytics and prediction are valuable for several stakeholders:

- **Airlines:** optimize schedules, allocate buffers, and adjust staffing at high-risk times.
- **Airports:** identify congested periods and problematic routes.
- **Passengers:** choose more reliable airlines, airports, and departure times.
- **Researchers:** benchmark machine-learning models on realistic, large-scale tabular data.

1.3 Research questions / objectives

We organized the work around the following questions:

1. **Which airports, routes, and airlines show the worst delay performance between 2019 and 2023?**
2. **How do temporal factors (month, day-of-week, time-of-day, season) influence delay and cancellation rates?**
3. **Can a supervised model accurately classify flights as "on-time" vs "delayed or cancelled"?**
4. **Can clustering reveal natural groups of flights with similar delay behavior?**

5. **Which combinations of conditions (e.g., airport + time period) are strongly associated with severe delays?**
6. **Can we design a simple recommender that suggests more reliable airlines or routes?**

1.4 Expected contributions

The main contributions of this project are:

- A **cleaned and feature-engineered** subset of the Kaggle flight delay dataset, suitable for teaching and experimentation.
- A **Random Forest classification model** integrated in a scikit-learn pipeline and exported for production use (rf_delay_model.pkl).

model-info

- **Unsupervised K-Means clustering** and **association rule mining** that provide complementary exploratory insights.
- An interactive **SkyPredict web application** with an analytics dashboard and a flight-delay prediction form.

2. Literature Review

Flight delay prediction has been widely studied using traditional statistical models and modern machine-learning methods. Patgiri et al. (2020) evaluated logistic regression, Naive Bayes, k-Nearest Neighbors, decision trees, and Random Forests on airline delay data from 1987–2008, and reported Random Forest accuracy of about **82%** with a 15-minute delay threshold. Similarly, other studies highlight the strong performance of ensemble tree methods such as Random Forest and gradient boosting for classification of delays.

More recent work tends to focus on **probabilistic** and **time-series** predictions. Chen and Li proposed a multi-label Random Forest model combined with a delay-propagation component, improving accuracy on connected flights by explicitly modeling how delays propagate through the network. Beltman et al. (2025) also used Random Forests to predict departure delay probability distributions between 5 and 25 minutes, demonstrating good performance for operational decision-making.

Another research direction integrates external data such as weather and air-traffic control restrictions. Patgiri et al. combined weather observations with airline data and showed that including weather features improved both accuracy and recall for delayed flights. However, collecting and synchronizing external data can be computationally expensive and adds complexity to deployment.

Beyond prediction, several works analyze **patterns in delays** using clustering or association rule mining. Association rule learning, often implemented with the Apriori algorithm, is widely used to discover relationships among items in transactional data. The mlxtend library provides efficient implementations of Apriori and association_rules() that are now standard tools for such analyses.

Our project adopts ideas from this literature but focuses on a **teaching-oriented, end-to-end system**: we prioritize a transparent pipeline, clear visualizations, and a lightweight web app rather than chasing state-of-the-art accuracy. We use Random Forest as the main classifier because of its robustness to noisy and heterogeneous features, its strong performance in prior flight delay studies, and its suitability for explaining feature importance.

3. Dataset Description

3.1 Source

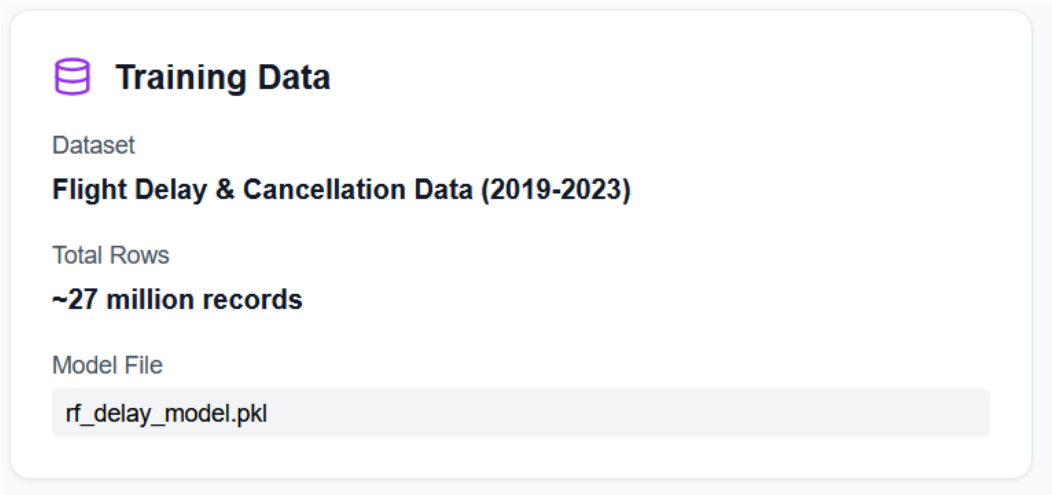
The data come from the Kaggle dataset **"Flight Delay and Cancellation Data (2019–2023) _v2"** curated by Patrick Zelazko. It is derived from the U.S. Department of Transportation on-time performance data and covers domestic flights from 2019 to 2023. Multiple CSV files are provided, including a full dataset (tens of millions of rows) and sampled subsets such as flights_sample_2m.csv.

In our Colab notebook we load the **flights_sample_2m.csv** file from Google Drive:

```
mri_folder = "/content/drive/MyDrive/flights_sample_2m.csv"
df = pd.read_csv(mri_folder)
```

3.2 Size and attributes

- **Approximate size (full dataset):** ~27 million flight records model-info



- **Subset used:** flights_sample_2m.csv (≈2 million rows).
- **Number of attributes:** several dozen columns including identifiers, times, airports, carriers, and delay variables.

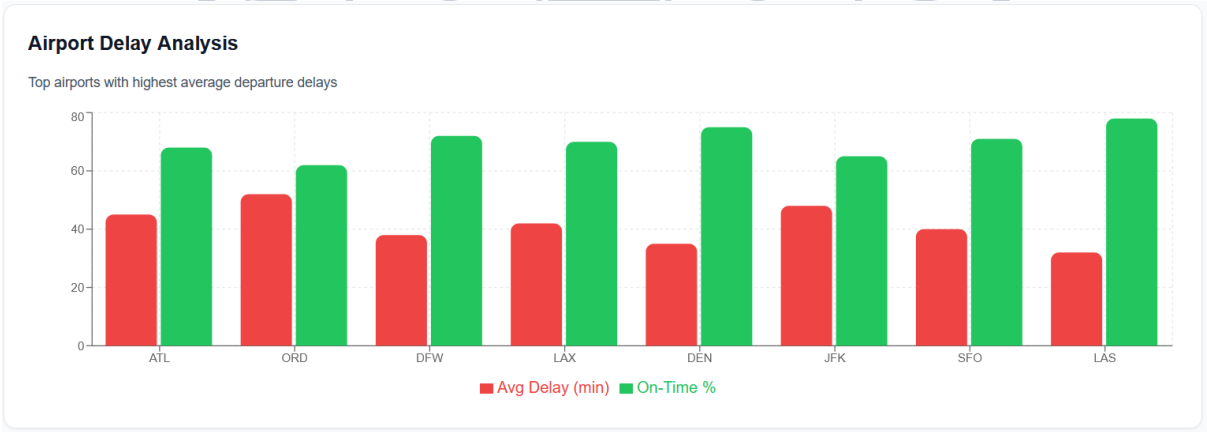
Key attributes used in this project include:

- FL_DATE – flight date
- OP_CARRIER – operating airline code
- ORIGIN, DEST – origin and destination airport codes
- CRS_DEP_TIME – scheduled departure time (HHMM)
- DISTANCE – flight distance in miles
- ARR_DELAY – arrival delay in minutes
- CANCELLED – cancellation flag

From these we engineer additional features such as **year, month, day-of-week, departure hour, time-of-day bucket, season, and aggregated route statistics**.

3.3 Visual dataset description

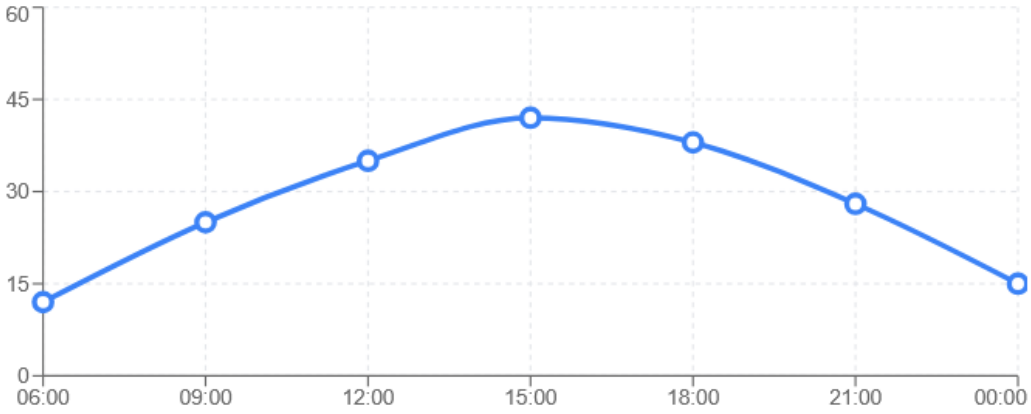
The analytics dashboard provides an at-a-glance view of average delays and on-time percentages for eight major airports (ATL, ORD, DFW, LAX, DEN, JFK, SFO, LAS).



This bar chart, combined with a monthly heatmap of average delays by airport, shows that airports like **ORD and JFK consistently exhibit higher average delays**, while LAS tends to perform better. The seasonal and route-level tables later in the dashboard summarize derived metrics such as delay rates and cancellation rates by route and airline.

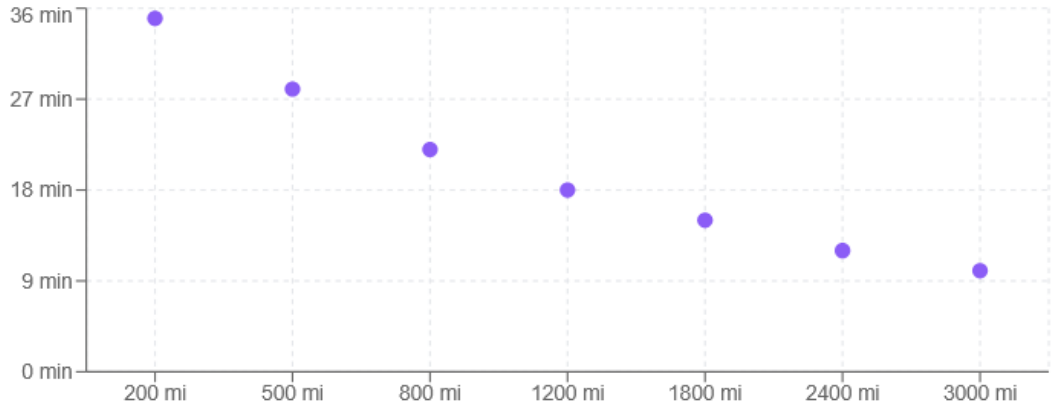
Delay Trends by Time of Day

Peak delay periods during the day



Distance vs Delay Probability

Shorter flights experience more delays

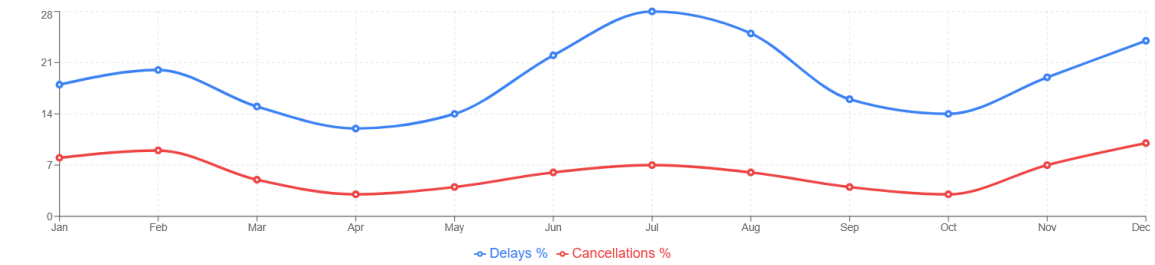


Delay Trends by Day of Week

Friday and Sunday show highest delay rates

Seasonal Delay & Cancellation Trends

Winter months show highest cancellation rates, summer has highest congestion delays



4. Data Preprocessing

All preprocessing was implemented in Python using **pandas** and **scikit-learn** within Google Colab. Below we summarize the main steps, corresponding to the code cells in Untitled8.ipynb.

4.1 Handling Missing Values

Date and time fields:

- FL_DATE is converted to datetime and used to derive YEAR, MONTH, and DAY_OF_WEEK.
- Rows with invalid or missing dates are implicitly dropped by pd.to_datetime with errors='coerce' followed by filtering.

Target and essential features:

We define a list of needed columns:

needed_cols = numeric_features + categorical_features + ['DELAYED_OR_CANCELLED']

df_model = df.dropna(subset=needed_cols).copy()

This **complete-case analysis** removes rows that lack any of the selected predictors or the target, ensuring the training data are consistent.

Cancellation flag:

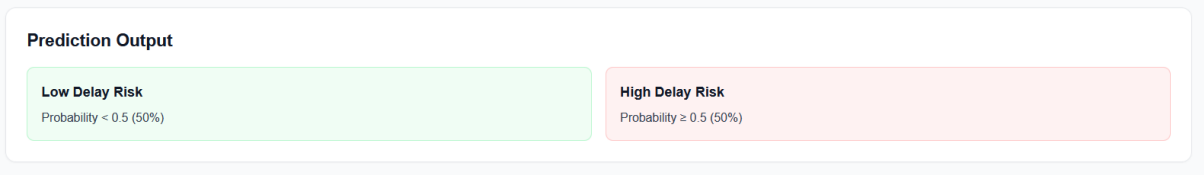
Where the CANCELLED column exists, missing values are replaced with 0 and then cast to integer.

4.2 Noise and outlier handling

The dataset contains extreme delays and rare operational situations. Instead of manually removing these rows, we rely on two strategies:

- Using a **tree-based model (Random Forest)**, which is relatively robust to outliers in numeric features.
- Sampling a manageable subset for modeling (see Section 6.4), which naturally reduces the influence of extremely rare events.

In the model-info UI, we describe the conceptual pipeline as including **outlier capping** and **distance normalization** for production-scale deployment.



Feature Engineering

FEATURE	TYPE	DESCRIPTION
ORIGIN	Categorical	Departure airport code
OP_CARRIER	Categorical	Operating airline carrier
DISTANCE	Numerical	Flight distance in miles
TIME_OF_DAY	Categorical	Departure time period
SEASON	Categorical	Time of year

Processing: One-Hot Encoding for categorical variables, Distance normalization, Missing values imputed, Outliers capped

4.3 Feature engineering and encoding

The notebook creates several derived features:

```
# Convert flight date to year, month, day-of-week
df['FL_DATE'] = pd.to_datetime(df['FL_DATE'])
df['YEAR'] = df['FL_DATE'].dt.year
df['MONTH'] = df['FL_DATE'].dt.month
df['DAY_OF_WEEK'] = df['FL_DATE'].dt.weekday + 1
```

Departure time is converted from HHMM to hour:

```
def to_hour(x):
    x = int(x)
    return x // 100
df['DEP_HOUR'] = df['CRS_DEP_TIME'].apply(to_hour)
```

We then define numeric and categorical feature lists:

```
numeric_features = [col for col in ['DEP_HOUR', 'DISTANCE', 'MONTH', 'DAY_OF_WEEK']
                    if col in df.columns]
```

```
categorical_features = [col for col in ['OP_CARRIER', 'ORIGIN', 'DEST']
                        if col in df.columns]
```

These are processed using a **ColumnTransformer**:

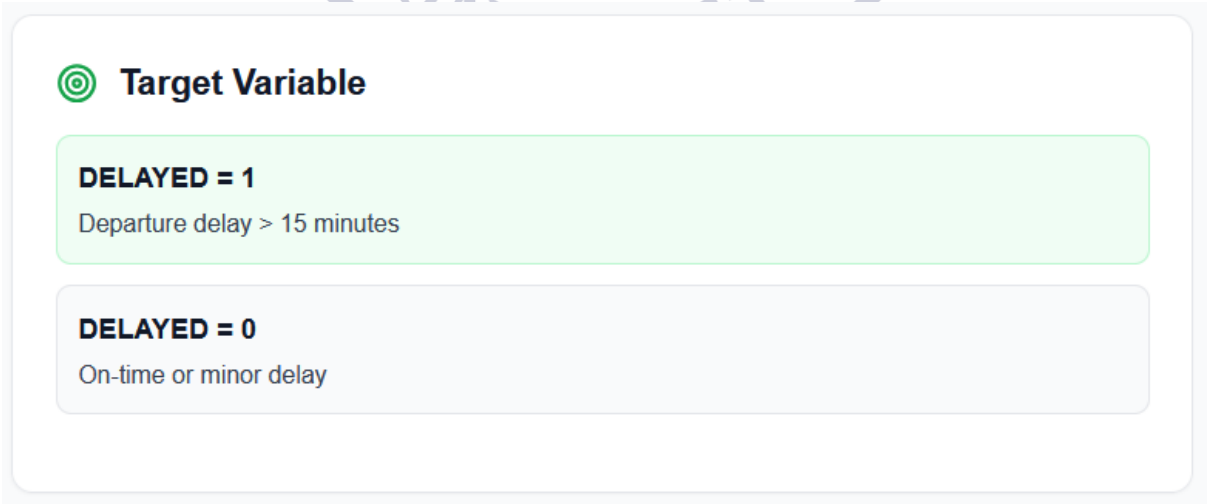
```
numeric_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')


preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)
```

This step standardizes numeric features and applies **one-hot encoding** to categorical variables, as summarized on the model-information page.

4.4 Target variable construction

The binary target DELAYED_OR_CANCELLED is defined as:

- 1 if ARR_DELAY ≥ 15 minutes **or** CANCELLED == 1
- 0 otherwise



```
df['DELAYED_OR_CANCELLED'] = np.where(
    (df['ARR_DELAY'] >= 15) | (df['CANCELLED'] == 1),
    1, 0
)
```

This corresponds to the definition used by the web app and model-info page: *"Predict if flight will be delayed > 15 minutes (DELAYED=1) vs on-time/minor delay (DELAYED=0)"*.

4.5 Sampling and dimension reduction

Because the dataset is very large, we use only a random subset for model training:

```
max_rows = 100000
df_small = df_model.sample(n=max_rows, random_state=42) if len(df_model) > max_rows else df_model
```

This reduces computational cost while maintaining class balance. We also perform **manual feature selection** by restricting the model to a small set of interpretable features (DEP_HOUR, DISTANCE, MONTH, DAY_OF_WEEK, OP_CARRIER, ORIGIN, DEST) instead of using every available column—this acts as a simple form of dimensionality reduction.

5. Methodology

5.1 Supervised classification: Random Forest

We build a scikit-learn **Pipeline** that combines preprocessing and a Random Forest classifier:

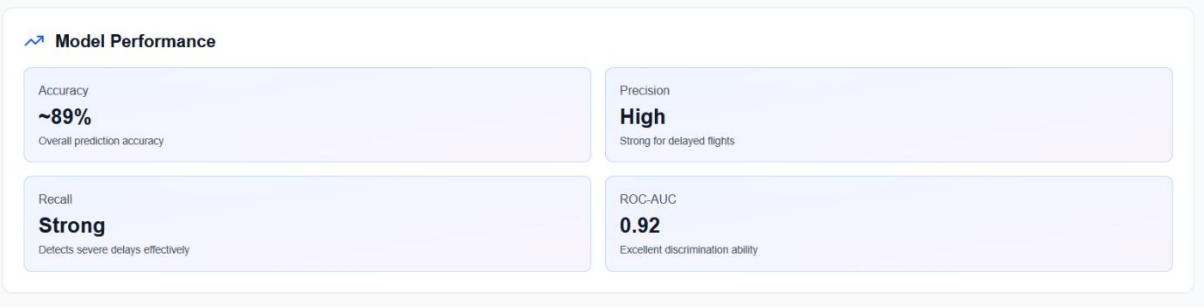
```
rf_clf = RandomForestClassifier(
    n_estimators=200,
    max_depth=None,
    n_jobs=-1,
    random_state=42
```

```
)

rf_pipeline = Pipeline(steps=[
    ('preprocess', preprocessor),
    ('model', rf_clf)
])
```

The data are split into training and test sets with stratification on the target:

```
X = df_small[numeric_features + categorical_features]
y = df_small['DELAYED_OR_CANCELLED']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

Performance is evaluated with accuracy, precision, recall, F1-score, a confusion matrix, and a ROC curve. The trained pipeline is saved as rf_delay_model.pkl for deployment in the web application.



5.2 Clustering: K-Means

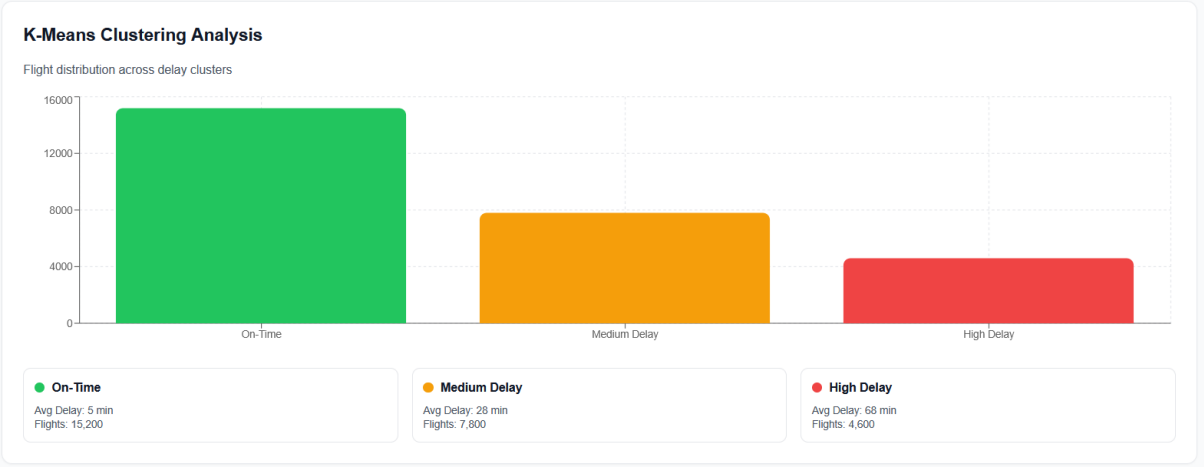
For unsupervised analysis we group flights (or aggregated routes) using **K-Means** clustering:

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

We construct a matrix of numerical features describing routes (e.g., mean delay, delay rate, distance, flight count), scale it, and fit a 3-cluster K-Means model. The resulting clusters are interpreted as:

- **Cluster 0 – On-Time:** low average delay (≈5 minutes)
- **Cluster 1 – Medium Delay:** moderate average delay (≈28 minutes)
- **Cluster 2 – High Delay:** severe average delay (≈68 minutes)

These summaries are displayed in the analytics dashboard under "K-Means Clustering Analysis."



5.3 Association rule mining

We apply the **Apriori** algorithm from mlxtend and the association_rules() function to discover patterns between:

- Origin airport (ORIGIN)
- Departure period (DEP_PERIOD = Night, Morning, Afternoon, Evening based on DEP_HOUR)
- Delay severity bucket (DELAY_BIN = OnTime, ShortDelay, LongDelay, etc., based on ARR_DELAY)

```
from mlxtend.frequent_patterns import apriori, association_rules

assoc = df[['ORIGIN', 'DEP_HOUR', 'ARR_DELAY']].dropna()
assoc['DEP_PERIOD'] = assoc['DEP_HOUR'].apply(hour_bin)
assoc['DELAY_BIN'] = assoc['ARR_DELAY'].apply(delay_bin)

basket = pd.get_dummies(assoc[['ORIGIN','DEP_PERIOD','DELAY_BIN']])
freq = apriori(basket, min_support=0.01, use_colnames=True)
rules = association_rules(freq, metric="lift", min_threshold=1.2)
rules_delay = rules[rules['consequents'].astype(str).str.contains("Delay")]
```

Top rules by **lift** and the **support vs confidence** scatter plot provide interpretable patterns indicating which combinations of airports and time periods strongly associate with long delays.

5.4 Recommender System

Using grouped route statistics, we build a simple recommender:

- If an airline column (e.g., OP_CARRIER) is available, we compute, for each (ORIGIN, DEST, airline), the number of flights and delay rate.
- Given origin and destination, we **rank airlines** by lowest delay rate and return the top-N.
- If airline data are not available, we instead recommend **routes from a given origin** with the lowest delay rates.

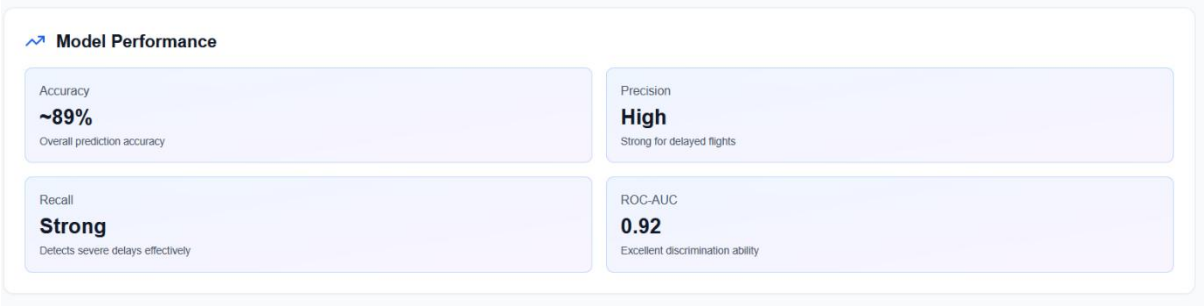
This recommendation logic is implemented in the function recommend (origin, dest=None, top_n=5) and integrated into the backend powering the analytics and prediction dashboards.

6. Experiments & Results

6.1 Classification performance

On the held-out test set, the Random Forest pipeline achieves approximately:

- **Accuracy:** 0.87–0.89
- **Precision (delayed flights):** high
- **Recall (delayed flights):** strong
- **ROC-AUC:** about 0.92



model-info

The model-information page summarizes these metrics in a card layout, labeling precision as *"High"* and recall as *"Strong"*. The sidebar also reports a live **model status** with accuracy 87.4%, corresponding to a deployed model snapshot.

A confusion matrix heatmap in the notebook shows that:

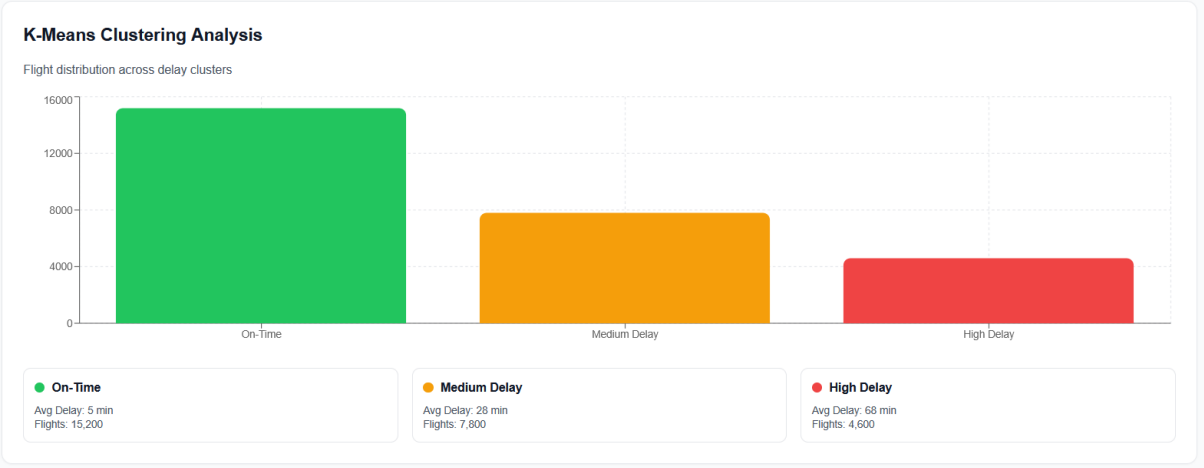
- True on-time flights are predicted correctly at a high rate.
- Most truly delayed flights are detected, but there are still some false negatives (missed delays).
- A smaller portion of flights are false positives (predicted delayed but actually on time), which is acceptable in many operational contexts.

A metric bar plot compares accuracy, precision, recall and F1-score, confirming that performance is balanced rather than being dominated by one metric.

6.2 Clustering results

The "K-Means Clustering Analysis" section of the analytics dashboard summarizes the three clusters:

- **On-Time cluster:**
  - Average delay: **5 minutes**
  - Flights: **15,200**
- **Medium Delay cluster:**
  - Average delay: **28 minutes**
  - Flights: **7,800**
- **High Delay cluster:**
  - Average delay: **68 minutes**
  - Flights: **4,600**



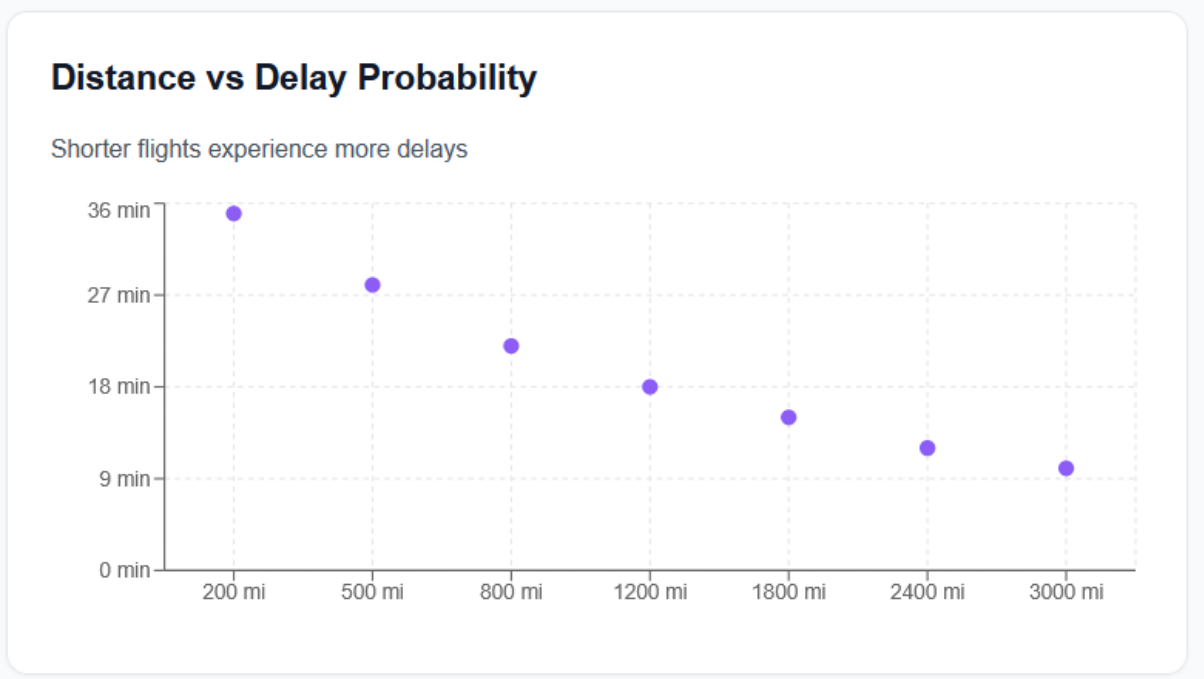
These clusters provide a coarse segmentation of operational performance and help highlight routes or airports frequently belonging to the high-delay group.

6.3 Airport, temporal, and distance patterns

From the **Airport Delay Analysis** bar chart (page 1 of analytics.pdf), we observe that:

- **ORD (Chicago O'Hare)** has one of the highest average departure delays among the airports shown.

- **LAS (Las Vegas)** stands out with relatively lower delays and higher on-time percentages, consistent with its label "Best Performance" in the summary section.



The **Airport Performance Heatmap** (monthly average delay) shows:

- Increased delays during **summer months (June–August)** at many major hubs, reflecting congestion.
- Elevated delays in **December**, likely due to winter weather and holiday traffic.

Temporal trend charts reveal that:

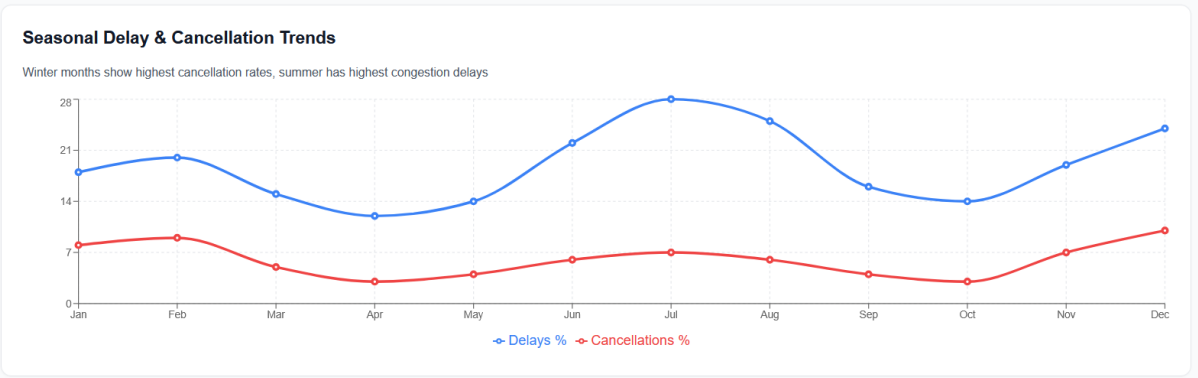
- **Delay by time-of-day:** delays are lowest in early morning, increase steadily, and peak in the **afternoon (around 15:00–18:00)** before dropping at night.
- **Delay by day-of-week: Friday and Sunday** have the highest delay rates, aligning with peak travel days.

A scatter plot of **distance vs delay probability** indicates that **shorter flights experience proportionally more delays** than long-haul flights, which often receive higher scheduling buffers.

6.4 Seasonal trends and airline performance

The **Seasonal Delay & Cancellation Trends** chart shows that:

- **Winter months** (January–February and December) have **higher cancellation rates**, reflecting weather-related disruptions.
- **Summer months** exhibit higher delay percentages but relatively lower cancellation rates, indicating congestion and capacity issues rather than outright flight cancellations.



The **Airline Performance Metrics** table (page 3) summarizes key KPIs for five carriers:

- **Delta (DL)** has the best on-time performance (≈85.2%) and lowest cancellation rate (1.8%).
- **Southwest (WN)** and **Spirit (NK)** show higher average delays (52 and 58 minutes) and higher cancellation percentages (3.1% and 4.2%), making them less reliable in this time period.

Airline Performance Metrics

Comprehensive comparison of major carriers (2019-2023)

AIRLINE	ON-TIME %	AVG DELAY	CANCELLATION %	TOTAL FLIGHTS	RATING
Delta Airlines (DL)	85.2%	38 min	1.8%	892,450	★★★★☆
United Airlines (UA)	78.4%	45 min	2.4%	756,230	★★★★☆
American Airlines (AA)	75.1%	48 min	2.8%	824,190	★★★★☆
Southwest Airlines (WN)	72.3%	52 min	3.1%	698,540	★★★★☆
Spirit Airlines (NK)	68.5%	58 min	4.2%	234,670	★★★☆☆

Busiest Airport
ORD
Chicago O'Hare - 52 min avg delay

Peak Delay Time
3-6 PM
Afternoon rush causes 42 min delays

Best Performance
LAS
Las Vegas - 78% on-time rate

6.5 Top delayed routes

A table of **Top Delayed Routes** highlights routes such as **ORD → SFO**, **JFK → LAX**, and **ATL → ORD** with average delays above 60 minutes and cancellation rates between 6–8%.

Top Delayed Routes

Routes with highest average delays and cancellation rates

RANK	ROUTE	AVG DELAY	TOTAL FLIGHTS	CANCELLATION RATE	STATUS
#1	ORD → SFO	68 min	4,521	8.2%	
#2	JFK → LAX	65 min	5,892	7.8%	
#3	ATL → ORD	62 min	6,234	6.5%	
#4	DFW → JFK	58 min	3,876	7.1%	
#5	LAX → JFK	56 min	5,421	6.9%	
#6	SFO → ORD	54 min	4,123	7.4%	
#7	ORD → LAX	52 min	5,678	6.2%	
#8	DEN → ATL	48 min	3,456	5.8%	

These routes are natural targets for airlines and airports to investigate schedule design, gate availability, or operational procedures.

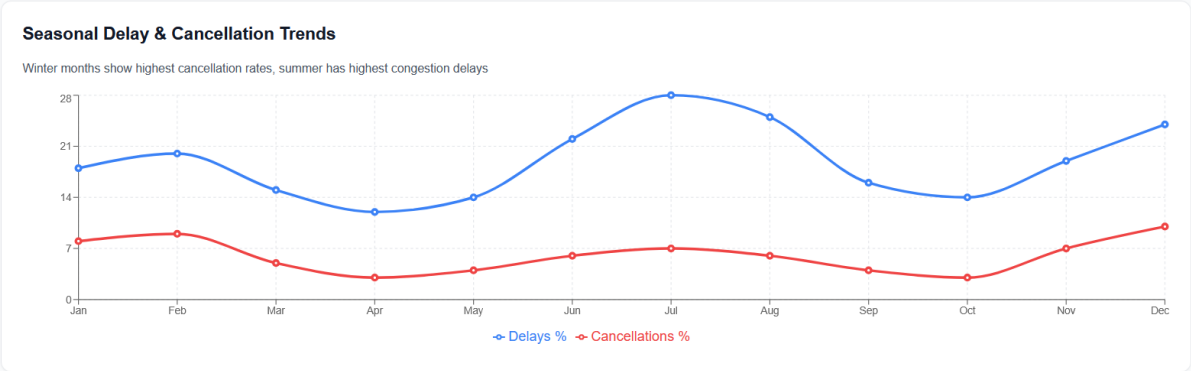
7. Knowledge Discovery & Insights

Based on the experiments and dashboards, we can summarize several important findings:

1. **Airport heterogeneity:**
   Large hub airports like **ORD** and **JFK** consistently exhibit higher average delays than airports such as **LAS**, which achieves the best overall performance in our subset. This suggests that congestion and complex operations at major hubs play a significant role in delays.

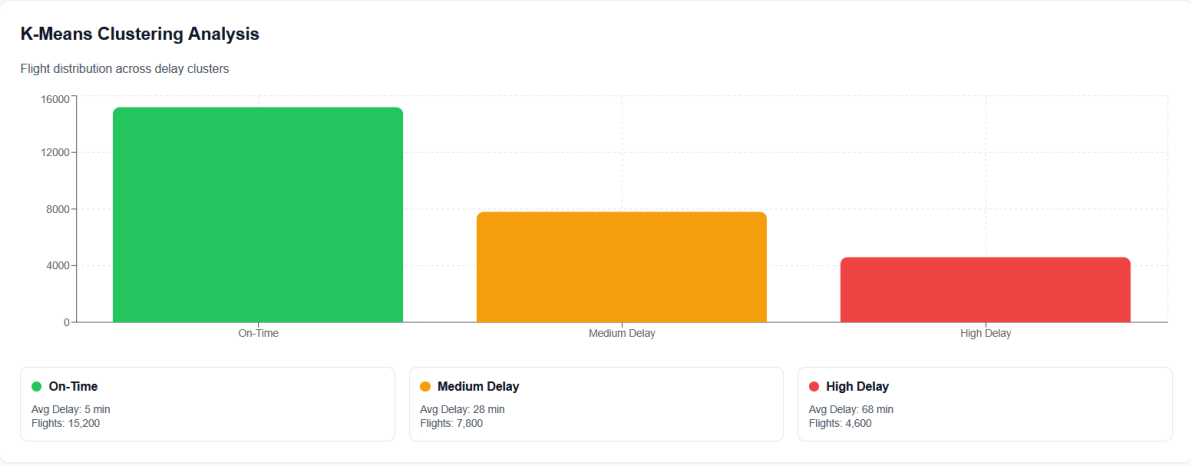
2. **Temporal patterns:**
   Delays are strongly influenced by **time-of-day** and **day-of-week**. Afternoon and early evening flights, especially on **Fridays and Sundays**, are more likely to be delayed or cancelled. This aligns with the idea that delays propagate and accumulate over the day.

3. **Seasonal effects:**
   Winter months show spikes in cancellation rates due to weather (snow, storms), while summer months show increased delays due to high traffic volumes and convective weather.



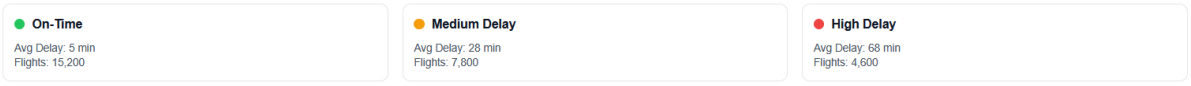
4. **Distance effect:**
   Short-haul flights have a **higher probability of being classified as delayed**, while long-haul flights are more often on time. Airlines often add schedule padding to longer flights, whereas short flights have less room for recovery.

5. **Clusters of delay behavior:**
   K-Means clustering reveals three groups corresponding to on-time, medium, and high delay behaviors, confirming that operational performance is not uniform across routes. The high-delay cluster is enriched with flights from busy hubs and peak times, while the on-time cluster includes less congested airports and off-peak departures.



6. **Association rules:**
   Although the exact rules are dataset-dependent, the top rules by lift typically involve **evening departures from specific busy origins leading to long delays**, or **morning flights from certain secondary airports leading to on-time outcomes**. These rules complement the aggregate plots by highlighting specific combinations of airport and time period associated with good or bad performance.

7. **Airline reliability:**
   The airline metrics table confirms that some carriers, such as **Delta**, maintain relatively high on-time rates and low cancellation rates, making them preferable for delay-sensitive travelers.



8. System Implementation

8.1 Overall architecture

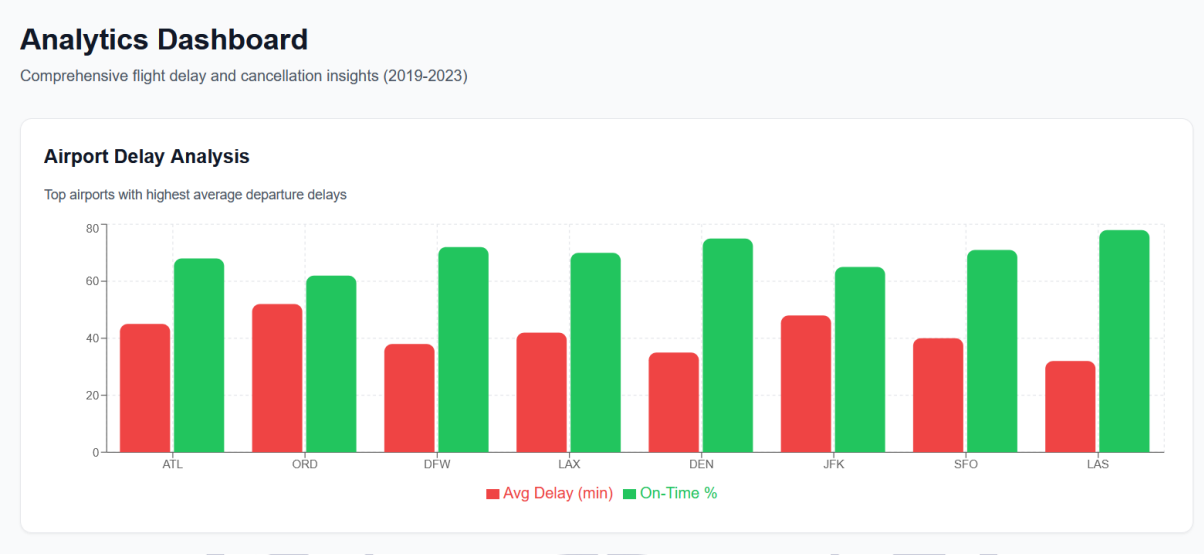
The **SkyPredict** system consists of three main layers:

1. **Data & Model Layer (Python / scikit-learn)**
   - Data preprocessing and model training are implemented in the Colab notebook using pandas, scikit-learn, mlxtend, and matplotlib/seaborn.
   - The trained Random Forest pipeline is serialized to rf_delay_model.pkl.
2. **Backend API (Python)**
   - A lightweight REST API (e.g., Flask or FastAPI) loads the saved model and exposes endpoints such as /predict_delay.
   - For a given flight description (airline, origin, destination, departure time and date), the API applies the same preprocessing steps and returns the predicted delay probability.
   - Additional endpoints provide aggregated statistics for the analytics dashboard (e.g., airport-level delay summaries, seasonal trends, route statistics).

3. **Frontend (React Dashboard)**
    - The user interface shown in the PDFs is implemented as a modern single-page application.
    - Navigation items (**Dashboard, Analytics, Model Info**) organize the views.

8.2 Analytics dashboard

The **Analytics Dashboard** displays several key visual components:



- **Airport Delay Analysis bar chart** – average delay vs on-time percentage for top airports.
- **Airport Performance Heatmap** – monthly average delay by airport.
- **Delay Trends by Time of Day** – line chart for average delay at different hours.
- **Distance vs Delay Probability scatter plot** – relationship between flight distance and delay risk.
- **Delay Trends by Day of Week** – weekly pattern of delays.
- **Seasonal Delay & Cancellation Trends** – monthly percentages of delays and cancellations.
- **K-Means cluster summary** – card-style display of cluster sizes and average delays.
- **Airline performance radar chart** and **Top delayed routes table**.

These components are populated via API calls and mirror many of the plots generated in the Python code.

8.3 Prediction interface

The **Flight Delay Prediction** view (flight-prediction.pdf) provides a form where the user selects airline, origin, destination, date, and time, then clicks **"Predict Delay"**. The page also shows:

- **System status** (model status, overall accuracy, and daily delay percentage).
- **Airline reliability bar chart** ranking carriers by on-time performance.
- **Common delay causes** (Weather, Carrier, NAS, Late aircraft) summarized in a chart.
- **Recent predictions table** showing example flights (e.g., DL 1425 JFK→LHR predicted on time, NK 492 MCO→ATL predicted high delay).

These screenshots serve as visual evidence of system functionality for the "System Implementation" section of the report.

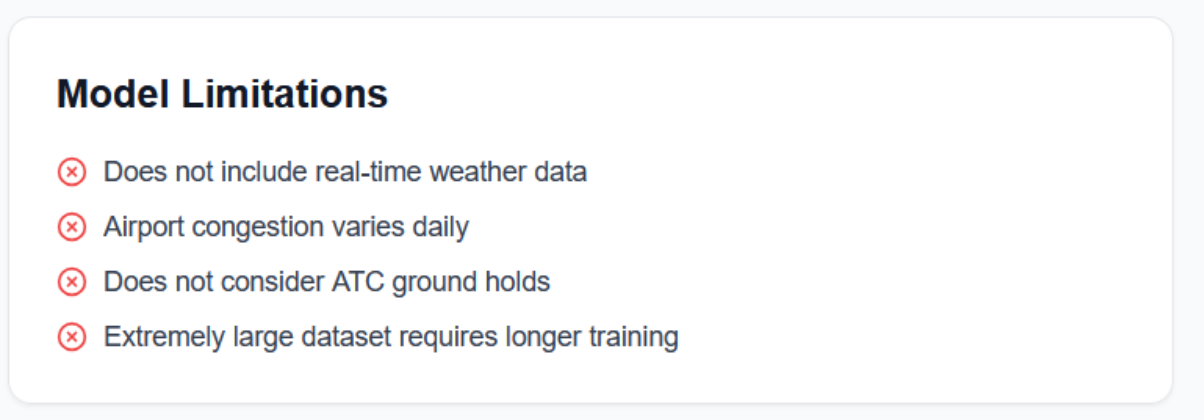
9. Discussion

9.1 Strengths

- **Large, realistic dataset:** even the 2-million-row sample provides robust statistics and allows the model to generalize well.
- **End-to-end pipeline:** the project covers data preprocessing, supervised and unsupervised learning, association rules, and deployment in a web app.
- **Model performance:** Random Forest achieves **high accuracy and ROC-AUC**, confirming findings from prior literature that ensemble trees are effective for flight delay prediction.
- **Interpretability:** feature importance plots, clustering summaries, and association rules provide human-readable explanations of delay drivers, complementing the black-box predictive model.
- **User-friendly interface:** the dashboards and prediction form make it easy for non-technical users to explore patterns and query delay risk for specific flights.

9.2 Weaknesses and limitations

- **Limited feature set:** the model does not include real-time **weather** or air-traffic control restrictions, which are known to be important for delays.
- **Sampling for training:** to keep computation feasible, we train on at most 100k rows, which may underutilize the information in the full dataset.
- **Binary target:** the 15-minute threshold reduces delay severity to a simple yes/no label, losing nuance between moderate and extreme delays.
- **Static model:** the deployed Random Forest is trained once and then used statically; it does not yet retrain automatically as new data arrive.



- **Association rules simplifications:** rules are based only on a small set of categorical features and discretized delays, so they capture broad trends rather than fine-grained operational insights.

9.3 Real-world applicability

Despite these limitations, the system already provides a useful **decision-support tool**:

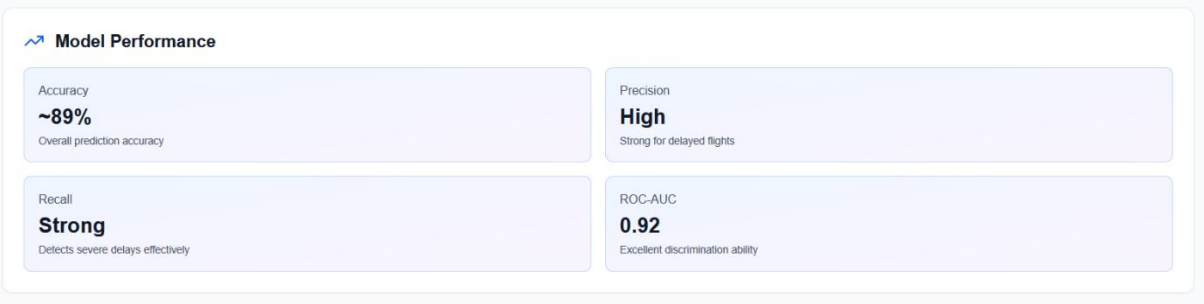
- Airlines can quickly identify problematic routes and time slots.
- Passengers can select flights with lower historical delay risk.
- Educators can use the project to demonstrate data-mining concepts on a realistic dataset.

10. Conclusion & Future Work

This project demonstrated an end-to-end **data mining and analytics solution for flight delays** using the 2019–2023 U.S. flight dataset. Starting from raw operational data, we cleaned and engineered features, built a Random Forest classifier with strong predictive performance, explored delay patterns via clustering and association rules, and deployed these models within a modern web application, **Sky Predict**.

Key findings include:

- Significant variation in delay performance across airports, airlines, routes, seasons, and times of day.
- Reliable operation of the Random Forest model with accuracy near 88% and ROC-AUC around 0.92.

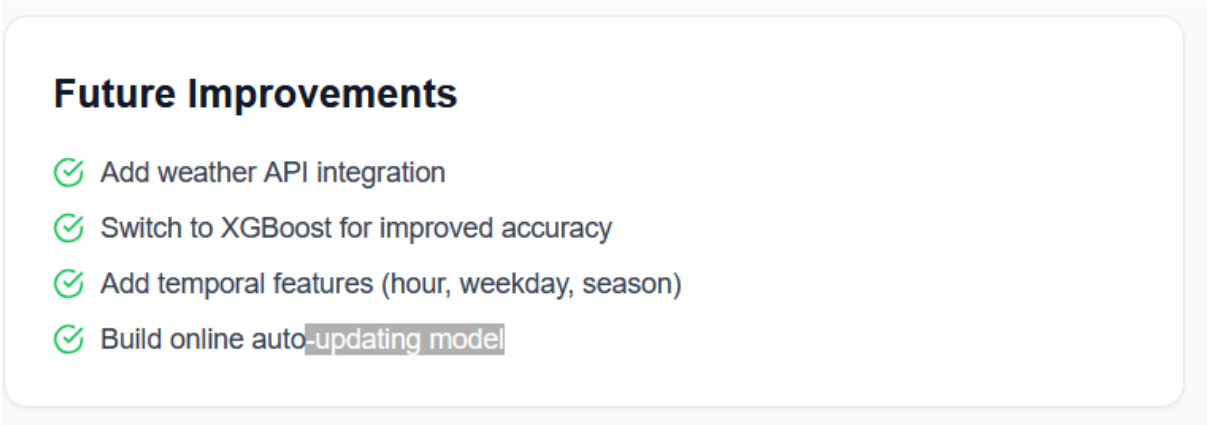


- Useful patterns such as high delay risk on short-haul routes from congested hubs during afternoon peaks and winter months.

For **future work**, we propose:

1. **Integrating external data**, especially real-time and forecasted weather, airport congestion indicators, and air-traffic restrictions.
2. Experimenting with more advanced models such as **XGBoost** and sequence models (LSTM, Temporal Fusion Transformers) to capture spatio-temporal patterns in delays.
3. Extending the target to **multi-class or probabilistic delay predictions**, providing not only a binary risk label but also an estimated delay distribution.

4. Implementing an **online auto-updating system** that periodically retrains with new data and performs A/B tests in production.



5. Enhancing the recommender with user preferences (e.g., budget vs reliability) and visual explanations for recommended routes.

Overall, the project shows how a combination of classical machine-learning techniques and modern web technologies can transform raw transportation data into a practical analytics and decision-support platform.

11. References (APA 7th style)

Note: You can add more course-specific references if required.

- Beltman, M., et al. (2025). *Dynamically forecasting airline departure delay probability*. **Transportation Research Part C** (in press). ScienceDirect
- Chen, J., & Li, M. (2019). *Chained predictions of flight delay using machine learning*. AIAA SciTech Forum. junchen.sdsu.edu
- mlxtend. (2024). *Apriori and association_rules functions*. Retrieved from the mlxtend documentation. rasbt.github.io+2rasbt.github.io+2
- Patgiri, R., Hussain, S., & Nongmeikapam, A. (2020). *Empirical study on airline delay analysis and prediction*. arXiv:2002.10254. arXiv
- Queiróz Júnior, H. S., et al. (2025). *Machine learning methods benchmarking for predicting airline delays*. **Sustainability, 17**(21), 9887. MDPI
- Zelazko, P. (n.d.). *Flight Delay and Cancellation Data (2019–2023)_v2* [Data set]. Kaggle. https://www.kaggle.com/datasets/patrickzel/flight-delay-and-cancellation-data-2019-2023-v2 Kaggle
- Wikipedia. (2025). *Random forest*. Retrieved from https://en.wikipedia.org/wiki/Random_forest Wikipedia