Flight Cancellation Prediction Using Domestic US Flights Data

Nazgul Salikhova, Chulpan Valiullina,
Milyausha Shamsutdinova, Diana Vostrova

n.salikhova@innopolis.university
c.valiullina@innopolis.university
m.shamsutdinova@innopolis.university
d.vostrova@innopolis.university

May 9, 2025

Contents

1 Introduction

This project aims to build a complete end-to-end Big Data pipeline to predict flight cancellations.

The analysis leverages historical flight data — including scheduled and actual departure/arrival times, airline details, origin and destination airports, delay reasons, and limited weather-related indicators — to detect patterns that are commonly associated with cancelled flights.

A machine learning model is developed as part of the predictive analytics stage to classify flights as likely to be cancelled or not (binary classifcation task). These insights can support both airline operations and passenger planning by helping to anticipate disruptions before they occur.

The project implements a distributed processing pipeline comprising the following stages:

- Data ingestion from PostgreSQL into HDFS using Apache Sqoop.
- Data storage and preparation in Hive, using efficient formats (AVRO/Parquet with Snappy compression).
- Exploratory data analysis using HiveQL and Tez execution engine.
- Model training and evaluation using Apache Spark MLlib.
- Visualization of results through interactive dashboards built in Apache Superset.

Business Objectives

The primary business objective of this project is to enhance decision-making in the airline industry by predicting flight cancellations in advance. Early identification of high-risk flights enables:

- **Airlines** to proactively manage operations, reallocate resources, notify passengers, and minimize operational disruption.
- **Passengers** to make more informed travel decisions, choose alternative itineraries, and avoid missed connections.
- **Air traffic controllers and airport authorities** to better manage airport logistics and slot allocation.
- **Customer service teams** to reduce strain during cancellation surges through preemptive communication.

By leveraging historical flight data and scalable Big Data technologies, this project demonstrates how predictive analytics can provide operational resilience and improve the travel experience in the face of uncertain conditions.

2 Data Description

Dataset Overview

The dataset used in this project is titled **"Domestic US Air Flights 2016–2018"** and was obtained from Kaggle. It contains flight-level data reported by certified U.S. air carriers, covering over 18 million domestic flights from 2016 to 2018. The dataset includes 26 features and has an uncompressed size of approximately 1.9 GB. These carriers each account for at least 1% of domestic scheduled passenger revenue and are required to report operational data to the US Department of Transportation's Bureau of Transportation Statistics (BTS).

The dataset is stored in a single CSV file named `combine_files.csv`, with an uncompressed size of approximately 1.8 GB. Additional reference data, such as airport codes and cancellation reasons (with corresponding codes and descriptions), was not provided explicitly but was stored as separate CSV files and used in subsequent processing. To ensure balance, a representative sample of 531,167 rows was selected for training the machine learning models.

Dataset Source and Purpose

The original data was collected by the Office of Airline Information, part of the BTS. It includes scheduled and actual departure/arrival times, flight durations, origin/destination codes, carrier information, and detailed records of delays, diversions, and cancellations.

A 2016 government report estimated that air transportation delays led to a $4 billion loss in US GDP. This dataset enables analysis of such delays and cancellations, providing valuable insights into operational inefficiencies and predictive modeling opportunities.

Dataset Features

The dataset contains **26 columns**, described as follows:

1. **Year**: Year of flight (2016, 2017, or 2018)
2. **Month**: Month of flight (1 to 12)
3. **DayofMonth**: Day of the month (1 to 31)
4. **DayOfWeek**: Day of the week (1 = Monday to 7 = Sunday)
5. **DepTime**: Actual departure time (local, hhmm)
6. **CRSDepTime**: Scheduled departure time (local, hhmm)
7. **ArrTime**: Actual arrival time (local, hhmm)
8. **CRSArrTime**: Scheduled arrival time (local, hhmm)
9. **ActualElapsedTime**: Total flight time in minutes
10. **CRSElapsedTime**: Scheduled elapsed time in minutes
11. **AirTime**: Actual airborne time in minutes
12. **ArrDelay**: Arrival delay in minutes (on time if < 15 minutes late)
13. **DepDelay**: Departure delay in minutes
14. **Origin**: Origin airport IATA code
15. **Dest**: Destination airport IATA code
16. **Distance**: Flight distance in miles
17. **TaxiIn**: Taxi-in time in minutes
18. **TaxiOut**: Taxi-out time in minutes
19. **Cancelled**: Cancellation indicator (1 = cancelled)
20. **CancellationCode**: Reason for cancellation:
    - A = Carrier
    - B = Weather
    - C = NAS (National Airspace System)
    - D = Security
21. **Diverted**: Diversion indicator (1 = flight was diverted)
22. **CarrierDelay**: Delay due to airline issues (e.g. maintenance, crew rest, baggage)
23. **WeatherDelay**: Delay due to weather (extreme conditions at departure, en route, or arrival)
24. **NASDelay**: Delay within control of NAS (non-extreme weather, airport operations, air traffic)
25. **SecurityDelay**: Delay due to security incidents (e.g. screening, terminal evacuation)
26. **LateAircraftDelay**: Delay due to late arrival of the same aircraft from a previous flight

Target Variable

For the purposes of this project, the key variable of interest is `Cancelled`, which serves as the binary label for classification. The goal is to predict this variable based on flight characteristics, times, and delay causes.

3 Architecture of Data Pipeline

This project follows an end-to-end Big Data architecture to handle, process, and analyze large-scale flight data. The architecture consists of several well-defined stages, with clear input and output formats at each step:

1. **Data Ingestion (PostgreSQL → HDFS)**:
   - **Input:** Raw CSV file `combine_files.csv` (600,000 rows)
   - **Process:** Loaded into PostgreSQL tables using `COPY` and preprocessing scripts
   - **Output:** Structured relational tables in PostgreSQL
2. **Data Transfer (Sqoop)**:
   - **Input:** PostgreSQL tables
   - **Process:** Exported to HDFS using Apache Sqoop
   - **Output:** Files in Parquet format (Snappy-compressed) in HDFS
3. **Storage and Query (Hive)**:
   - **Input:** Parquet files in HDFS
   - **Process:** External tables defined in Hive; partitioned and bucketed
   - **Output:** Queryable Hive tables
4. **Analysis and Modeling (Spark)**:
   - **Input:** Hive tables via Spark
   - **Process:** EDA and classification models using Spark DataFrame and MLlib
   - **Output:** Trained ML models and predictions
5. **Presentation (Apache Superset)**:
   - **Input:** Analytical results and prediction outputs
   - **Process:** Dashboard design and storytelling
   - **Output:** Interactive dashboard with insights and visualizations

4 Data Preparation

ER Diagram

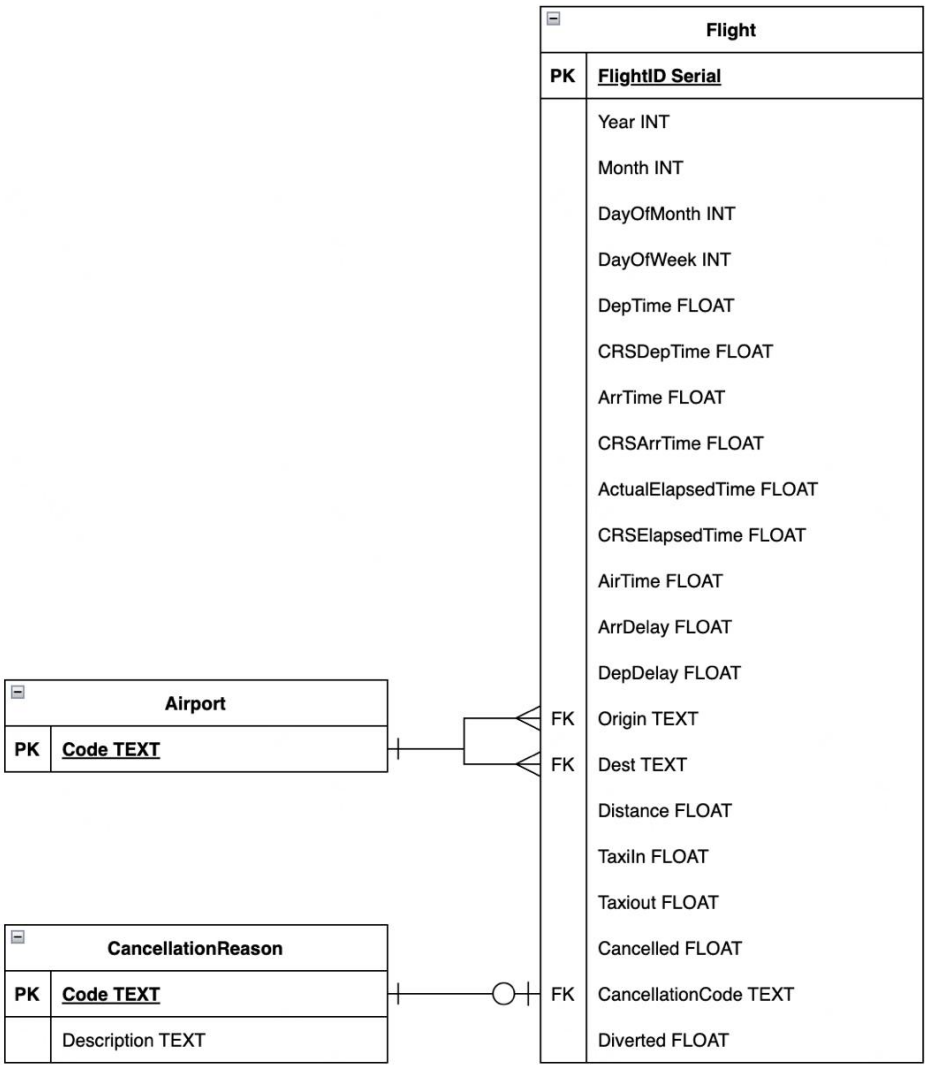
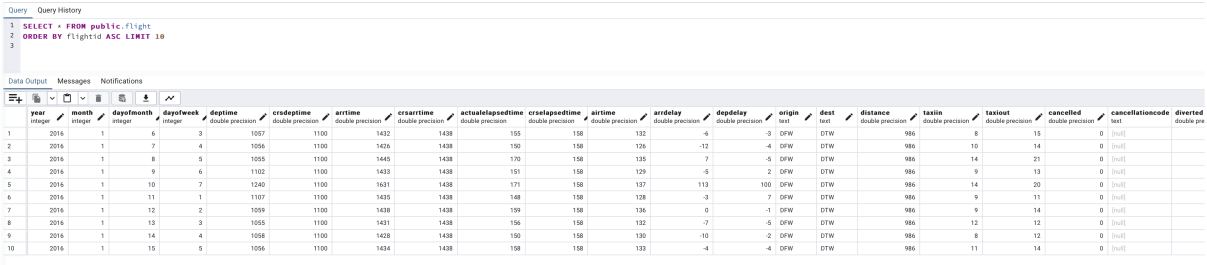


Figure 1.1: ER Diagram of the Flight Dataset

Samples from the database



```
SELECT * FROM public.flight
ORDER BY flightid ASC LIMIT 10
```

	year integer	month integer	dayofmonth integer	dayofweek integer	deptime double precision	crsdeptime double precision	arrtime double precision	crsarrtime double precision	actualelapsedtime double precision	crselapsedtime double precision	airtime double precision	arrdelay double precision	depdelay double precision	origin text	dest text	distance double precision	taxiin double precision	taxiout double precision	cancelled double precision	cancellationcode text	diverted double pre
1	2016	1	6	3	1057	1100	1432	1438	155	158	132	-6	-3	DFW	DTW	986	8	15	0	[null]	
2	2016	1	7	4	1056	1100	1426	1438	150	158	126	-12	-4	DFW	DTW	986	10	14	0	[null]	
3	2016	1	8	5	1055	1100	1445	1438	170	158	135	7	-5	DFW	DTW	986	14	21	0	[null]	
4	2016	1	9	6	1102	1100	1433	1438	151	158	129	-5	2	DFW	DTW	986	9	13	0	[null]	
5	2016	1	10	7	1240	1100	1631	1438	171	158	137	113	100	DFW	DTW	986	14	20	0	[null]	
6	2016	1	11	1	1107	1100	1435	1438	148	158	128	-3	7	DFW	DTW	986	9	11	0	[null]	
7	2016	1	12	2	1059	1100	1438	1438	159	158	136	0	-1	DFW	DTW	986	9	14	0	[null]	
8	2016	1	13	3	1055	1100	1431	1438	156	158	132	-7	-5	DFW	DTW	986	12	12	0	[null]	
9	2016	1	14	4	1058	1100	1428	1438	150	158	130	-10	-2	DFW	DTW	986	8	12	0	[null]	
10	2016	1	15	5	1056	1100	1434	1438	158	158	133	-4	-4	DFW	DTW	986	11	14	0	[null]	

Figure 1.2: Sample Flight Records (Screenshot is taken form PgAdmin 4)

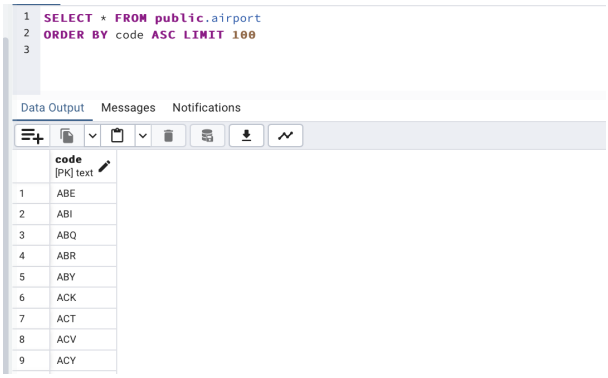


Figure 1.3: Sample Airport Records (Screenshot is taken form PgAdmin 4)

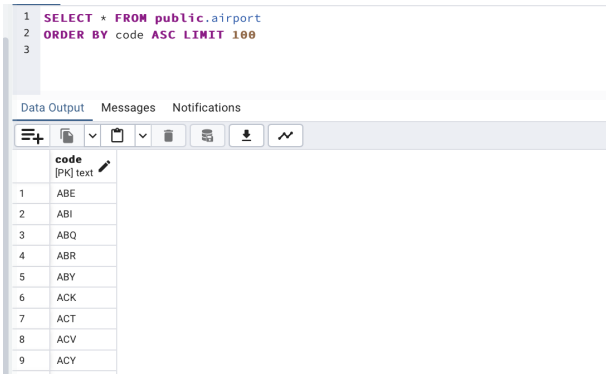


Figure 1.4: Sample CancellationReason Records (Screenshot is taken form PgAdmin 4)

Data Preparation and Hive Table Creation

The initial data setup involves preparing HDFS and defining Hive structures. AVRO schemas (`.avsc` files) are first placed in a designated HDFS location. Then, within Hive, the `team15_projectdb` database is created. External tables (`airport`, `flight`, `cancellationreason`) are subsequently defined, mapping to AVRO-serialized data in HDFS and referencing their respective schemas. Key HiveQL statements from `sql/db.hql` for this process include dropping and reacreating database, creating external tables and verifying data loading:

```
DROP DATABASE IF EXISTS team15_projectdb CASCADE;
CREATE DATABASE team15_projectdb LOCATION "project/hive/warehouse";
USE team15_projectdb;

CREATE EXTERNAL TABLE airport
STORED AS AVRO
LOCATION 'project/warehouse/airport'
TBLPROPERTIES ('avro.schema.url'='project/warehouse/avsc/airport.avsc');

SELECT * FROM airport LIMIT 5;
```

Script Execution Workflow

The entire setup and data processing pipeline is orchestrated by a main bash script. This script handles HDFS directory management, uploads AVRO schemas, and executes a sequence of HiveQL scripts using `beeline`. Example commands from the execution script are:

```
hdfs dfs -mkdir -p project/warehouse/avsc
hdfs dfs -put -f output/*.avsc project/warehouse/avsc

password=$(head -n 1 secrets/.hive.pass)

beeline -u jdbc:hive2://hadoop-03.uni.innopolis.ru:10001 -n team15 -p $password -f sql/db.hql
↪  > output/hive_results.txt

beeline -u jdbc:hive2://hadoop-03.uni.innopolis.ru:10001 -n team15 -p "$password" -f
↪  sql/optimization.hql

bash scripts/eda_queries.sh "$password"
```

The bash script first prepares HDFS by creating necessary directories and uploading AVRO schema files. It then executes `sql/db.hql` to set up the database and tables, redirecting output to `output/hive_results.txt`. Subsequent `beeline` calls execute further HiveQL scripts for data optimization and exploratory data analysis.

5 Data Analysis

Exploratory data analysis was performed in Hive. Insights include patterns by month, time of day, and carrier. Key findings were visualized in Superset.

Analysis Results. Main insights

- **Overall Flight Cancellation Rate**

  This analysis reveals that only 1.43% (265,138) of flights were cancelled, with 98.57% (18,240,587) operating as scheduled. While this cancellation rate seems low, when applied to the millions of flights in the US domestic market, it represents significant disruption to passengers and operational challenges for airlines. This baseline metric helps us understand the scale of the problem we're addressing with our predictive model.

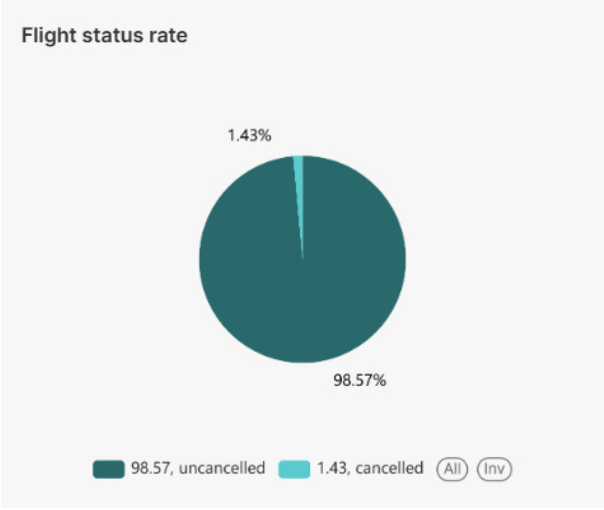


Figure 1.5: Insight 1: Overall Flight Cancellation Rate

- **Monthly Flight Cancellation Trends**

  The data shows clear seasonal patterns in flight cancellations, peaking at approximately 40,000 cancellations in the first month of the year (January), because of the winter Holidays and dropping to as low as 8,980 towards the end of the year. Airlines can use this information to anticipate

high-risk periods and allocate additional resources accordingly. Metrics such as cost per thousand cancellations could also highlight the significant financial impact during high-cancellation months.

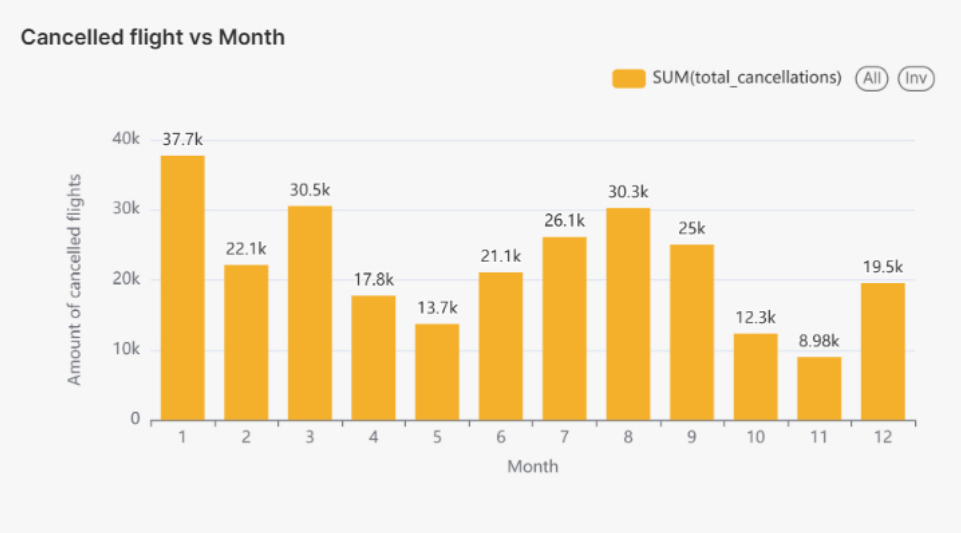


Figure 1.6: Insight 2: Monthly Flight Cancellation Trends

- **Airport Cancellation Hotspots**

Certain airports, such as ORD (Chicago) and LGA (New York), appear as cancellation hotspots, exhibiting particularly high cancellation counts. Airlines should investigate the factors contributing to these higher rates and explore what makes other airports more resilient to cancellations.

Figure 1.7: Insight 3: Airport Cancellation Hotspots

- **Primary Reasons for Flight Cancellations**

Weather issues (54.65%) are the leading cause of flight cancellations, responsible for significantly more disruptions than Carrier-related issues (25.78%), which represent internal airline problems like maintenance or crew issues. Problems within the National Airspace System (NAS, 19.43%), such as air traffic control limitations, contribute the third most cancellations. Security reasons account for a very small fraction of cancellations (0.15%).

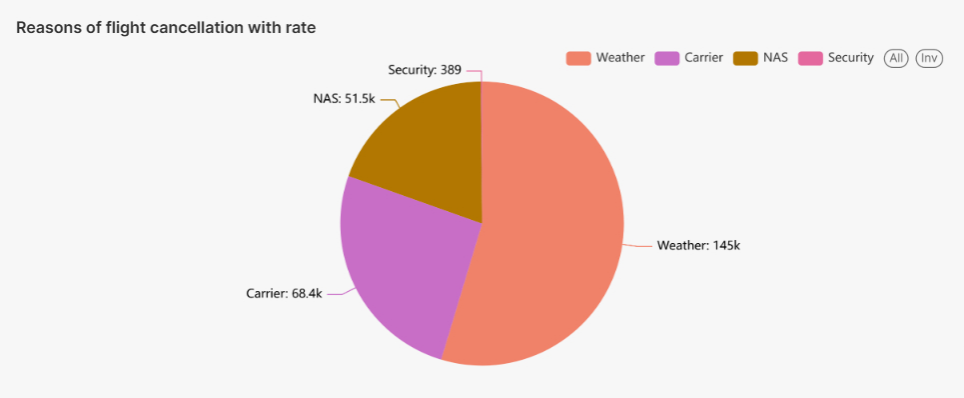


Figure 1.8: Insight 4: Primary Reasons for Flight Cancellations

- **Cancellations and Flight Volume by Day of the Week**

  The analysis reveals the distribution of cancellation rates and total flight volumes across the days of the week (where 1=Monday, 7=Sunday). Interestingly, Saturdays experience the lowest total number of flights compared to other days, because business flights and vacation beginnings or endings are mostly on sundays and fridays. The cancellation rate generally follows the pattern of total flights, indicating that higher flight volumes may correlate with slightly higher absolute numbers of cancellations, though the rate itself might vary.

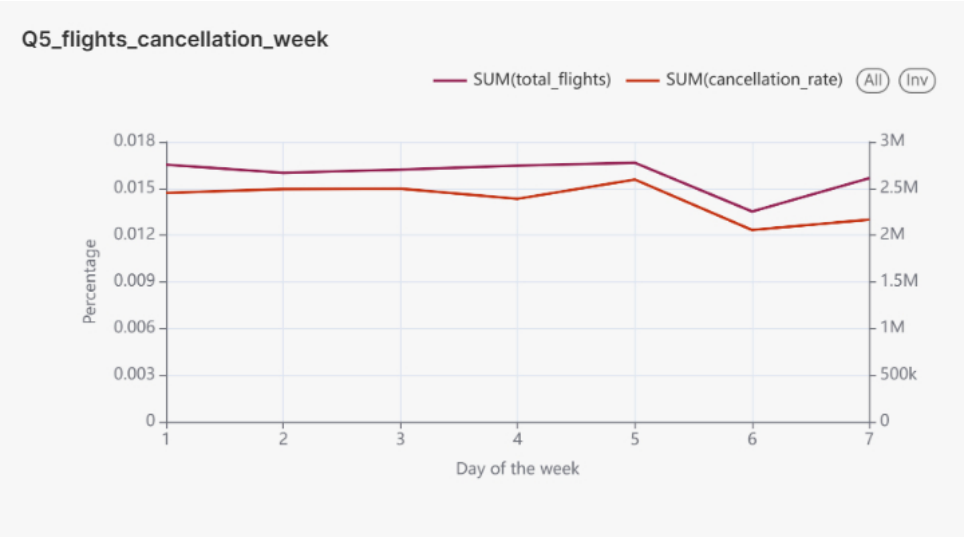


Figure 1.9: Insight 5: Cancellations and Flight Volume by Day of the Week

- **Impact of Departure Delay on Cancellation Rate**

  The analysis shows a clear positive correlation between the length of a departure delay and the likelihood of flight cancellation. Flights departing on time or early have a minimal cancellation rate (approximately 0.025%). This rate increases progressively with longer delays, reaching over 0.55% for flights delayed by more than 4 hours. This underscores the importance of on-time performance in mitigating cancellations.

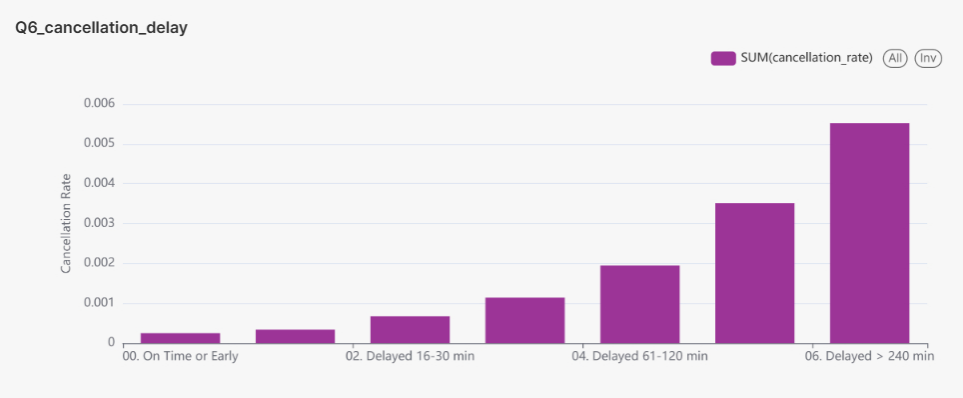


Figure 1.10: Insight 6: Impact of Departure Delay on Cancellation Rate

6 Modeling and Prediction

The primary objective was to develop, train, and evaluate machine learning models using PySpark ML to predict flight cancellations. The process involved comprehensive data preparation, model selection, hyperparameter tuning, and evaluation, all executed on a YARN cluster leveraging data stored in Hive and HDFS.

Dataset overview

- Source: Data was loaded from Hive tables within the team15_projectdb database, primarily utilizing the flight table.
- Target variable: The binary column "cancelled" from the flight table (0: Not cancelled, 1: Cancelled).
- Initial size: The flight table contained 18,505,725 records.

Data preparation pipeline

A robust preprocessing pipeline was constructed using PySpark to prepare the data for ML modeling:

1. *Feature selection*:

   Relevant features selected from the flight table included: year, month, dayofmonth, dayofweek, crsdeptime (scheduled departure time), crsarrtime (scheduled arrival time), crselapsedtime (scheduled elapsed time), origin, dest (destination), and distance. The cancelled column was designated as the label. Columns with high null counts directly related to cancellation outcomes (e.g., cancellationcode) or high cardinality without direct predictive value post-processing were excluded.

2. *Handling missing values*: Rows with null values in the selected features were dropped. This reduced the dataset size marginally from 18,505,725 to 18,505,702 records.

3. *Time features decomposition*:

   - Cyclical features **month, dayofmonth, and dayofweek** were transformed using sine and cosine functions to capture their cyclical nature.
   - Scheduled time features **crsdeptime, crsarrtime** were first split into hour and minute components, which were then also transformed using sine and cosine functions.

4. *Categorical features encoding*: Categorical string features **origin, dest, year** were converted to numerical representations using StringIndexer followed by OneHotEncoder.

5. *Numerical features scaling*: Continuous numerical features **crselapsedtime, distance** and the newly engineered cyclical features were scaled using StandardScaler.

6. *Feature assembling*: All processed features (scaled numerical, OHE categorical, engineered cyclical) were combined into a single features vector column using VectorAssembler.

7. *Dataset balancing*: The dataset exhibited significant class imbalance for the cancelled label. To address this, the majority class (not cancelled) was down-sampled using sampleBy, resulting in balanced dataset of **531,167 records** for modeling.

8. *Train-test split*: The balanced dataset was split into training (80%) and validation (20%) sets, stratified by the cancelled label to ensure proportional representation in both sets.

   - Training set size: **425,472 records**.
   - Validation set size: **105,695 records**.

9. *Data saving*: The final processed training and validation datasets were saved to HDFS in JSON format (project/data/train.json and project/data/val.json) and subsequently copied to the local filesystem (data/train.json and data/val.json).

ML model training with hyperparameters tuning

Two classification models were trained and tuned using 3-fold cross-validation on the training data. The BinaryClassificationEvaluator with the areaUnderROC (AUC) metric was used to guide hyperparameter selection.

Models chosen for training:

1. Logistic Regression

   - Hyperparameters tuned: regParam ([0.01, 0.1, 0.5]), elasticNetParam ([0.0, 0.5, 1.0]), maxIter ([10, 50]).
   - Best parameters found: elasticNetParam = 0.0, maxIter = 50, regParam = 0.01.

2. Random Forest

   - Hyperparameters tuned: numTrees ([20, 50]), maxDepth ([5, 10, 15]), featureSubsetStrategy ([”sqrt”, ”log2”]).
   - Best parameters found: featureSubsetStrategy = ’sqrt’, maxDepth = 15, numTrees = 50.

Model evaluation and comparison

Models were evaluated on the validation set using AUC as the primary metric, supplemented by Accuracy, Precision, Recall, and F1-score for the positive class (cancelled=1).

Model	**AUC**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Logistic Regression	69.58%	64.16%	63.79%	65.18%	64.48%
Random Forest	**76.49%**	**69.65%**	**69.39%**	**69.78%**	**69.59%**

Table 1.1: Model Comparison

As shown in the Table 1.2, The Random forest classifier demonstrated superior performance across all evaluated metrics, achieving an AUC of 76.49%.

Model and prediction outputs

- Models saved: The best trained Logistic Regression and Random Forest models were saved to HDFS (in project/models/ folder) and also copied to the local models/ directory.
- Predictions saved: Predictions made by each best model on the validation set, containing the actual cancelled label and the prediction, were saved to HDFS as single-partition CSV files (in project/output/ folder). These were also copied to the local output/ directory.
- Evaluation results saved: The comparison table of model performance metrics was saved to HDFS (project/output/evaluation.csv) and locally (output/evaluation.csv).

Outcomes

The stage 3 successfully demonstrated the end-to-end process of building predictive models for flight cancellation using PySpark ML. Key achievements include:

- Comprehensive data preprocessing, including handling of temporal and categorical features, and dataset balancing. Training and rigorous hyperparameter tuning of Logistic Regression and Random Forest models.
- Comparative evaluation, identifying Random Forest as the better-performing model with an AUC of 76.49% on the validation set.
- All project requirements for this stage were met, including the use of PySpark ML on a YARN cluster, data sourcing from Hive, HDFS storage for models and outputs, and adherence to specified output formats. The process was automated via scripts (stage3.sh) and code quality was maintained (using pylint).

7 Data Presentation

To present the results of our analysis and model predictions, we developed an interactive dashboard using Apache Superset. The dashboard is publicly available at link

It consists of three main tabs:

1. **Data Description**: Presents a summary of the dataset, including its source, format, number of records, and sample rows with cancellations. It also describes how the data was stored in PostgreSQL and later processed into optimized Hive tables using AVRO and Snappy compression.

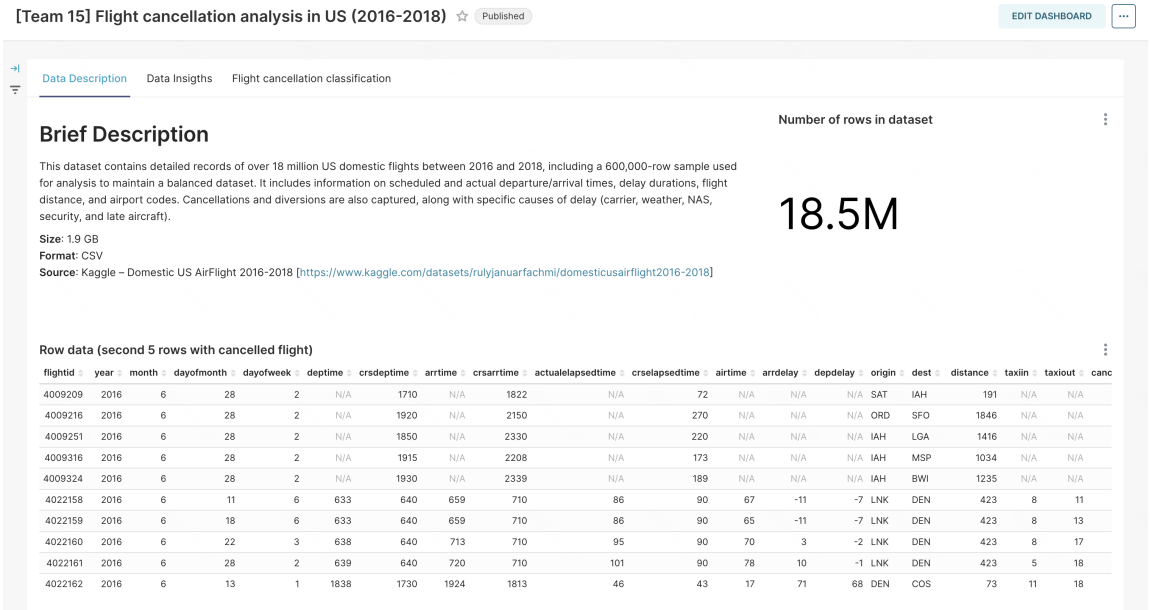


Figure 1.11: Dashboard tab – Data Description

2. **Data Insights**: Includes six visual charts supporting the exploratory data analysis, such as cancellation trends by month, airport hotspots, cancellation reasons, day-of-week patterns, and delay correlation with cancellations. Each insight is also documented and interpreted in detail in Section 5 of the report.

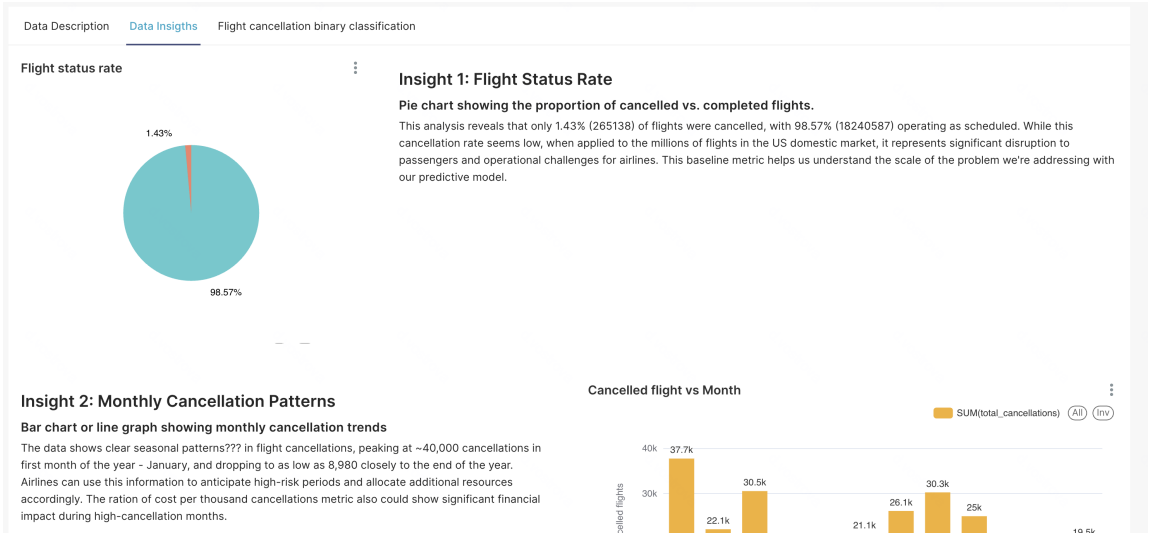


Figure 1.12: Dashboard tab – Data Insights

3. **Flight Cancellation Binary Classification**: This tab presents the final stage of the project — model evaluation and binary prediction of flight cancellations. It includes:

   - **Overview panel**: Summarizes the goal of building a binary classifier using PySpark ML to predict the target variable `Cancelled`, where 1 indicates a cancelled flight. It provides the final dataset distribution: 265,138 cancelled flights and 334,862 non-cancelled, totaling 600,000 records after balancing.
   - **Model training summary**: Outlines the hyperparameter tuning process for both models — Logistic Regression and Random Forest. For each model, the grid search configuration and the best selected parameters are listed (e.g., maxIter = 50, regParam = 0.01 for Logistic Regression; numTrees = 50, maxDepth = 15 for Random Forest).
   - **Performance metrics**: Evaluation results for the best models are shown, including AUC, accuracy, precision, recall, and F1-score. As previously described in Section 6, Random Forest achieved the highest AUC of 76.49% and outperformed Logistic Regression across all metrics.
   - **Prediction examples**: Two separate tables visualize predictions made by the models on sample records. These tables compare actual vs. predicted cancellation labels to illustrate typical classification behavior and model tendencies.

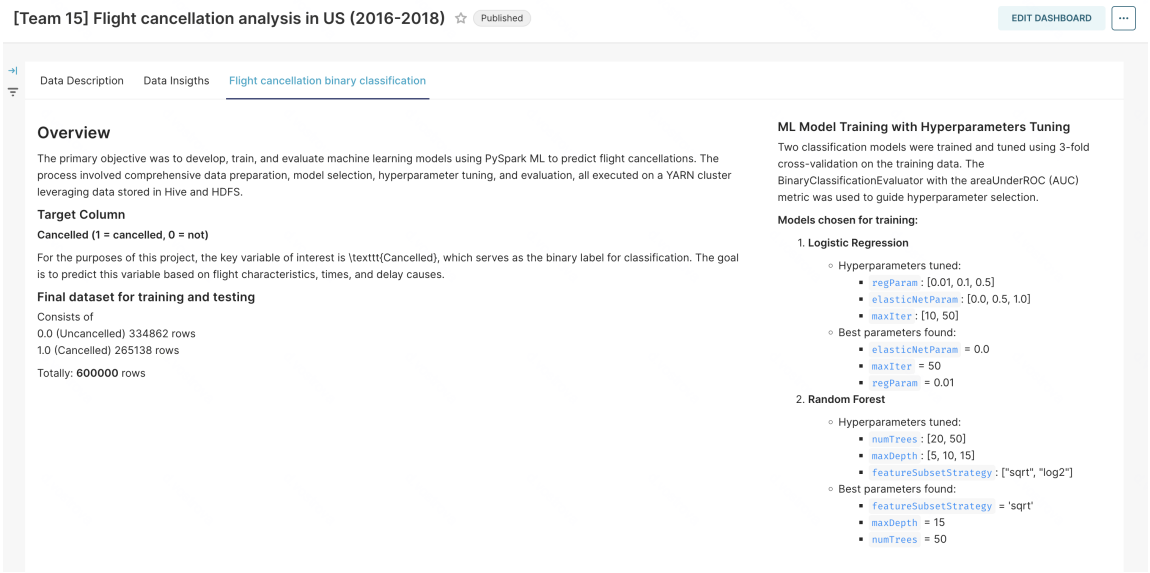


Figure 1.13: Model overview, training settings, and dataset distribution

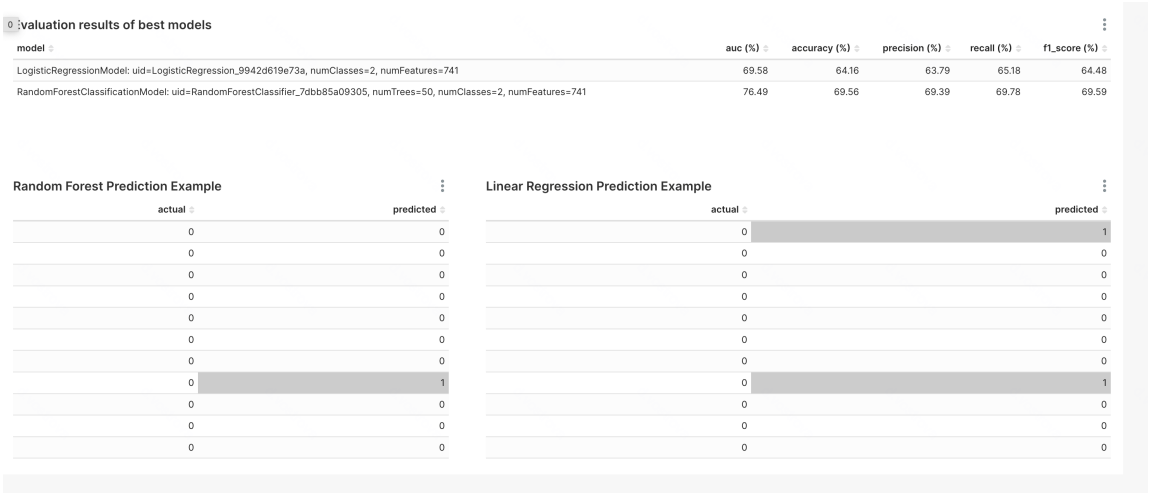


Figure 1.14: Evaluation results and example predictions for both models

8 Conclusions

In this project, we successfully designed and implemented a complete end-to-end Big Data pipeline to predict flight cancellations using domestic US flight data from 2016–2018. The workflow covered data ingestion, storage optimization, analysis, machine learning, and visualization.

We were able to identify meaningful patterns in cancellation behavior and train predictive models that achieved solid classification performance, with the Random Forest model reaching an AUC of 76.49%.

Throughout the project, we learned to:

- Ingest large-scale datasets into HDFS using PostgreSQL and Apache Sqoop.
- Store and optimize data in Hive using AVRO format with SNAPPY compression.
- Perform scalable exploratory data analysis using Hive and Apache Spark.
- Train and evaluate machine learning models using PySpark MLlib.
- Create clear, interactive visualizations using Apache Superset.

9 Reflections

Challenges and difficulties

- Hive tables optimization
- Long model training time
- Working in cluster in the first time
- Superset is not intuitively understable
- Cluster is down several times

Recommendations

- Use additional data sources (weather, traffic)
- Improve model interpretability
- Enable real-time inference

Team Contributions

Project Task	Description	Nazgul	Chulpan	Milyausha	Diana	Deliverables	Hours
[Stage 1] Data extraction	Download dataset from Kaggle and extract 600k sample	100%				`combined_data.csv`, `airport.csv`, `cancellationreason.csv`	2
[Stage 1] ER diagram and relational model	Create schema in Draw.io, write SQL scripts to create and populate PostgreSQL tables	100%				`data_collection.sh`, `data_preprocess.py`, `create_tables.sql`, etc.	2
[Stage 1] Sqoop ingestion to HDFS	Configure Sqoop and transfer data from PostgreSQL to HDFS; finalize with reproducible `stage1.sh` script	90%		10%		`stage1.sh`, AVRO files in `/output`	2
[Stage 2] Hive table creation	Create external Hive tables, define schemas, apply partitioning and bucketing	25%	75%			Hive schema and AVRO definitions	9
[Stage 2] Data preparation	Clean, filter, and transform dataset in Hive for analysis	15%	75%	10%		Cleaned Hive tables	10
[Stage 2] Exploratory Data Analysis	Run HiveQL queries and extract insights	10%	80%	5%	5%	Superset-ready query outputs, insight summaries	8
[Stage 2] Superset charting	Create and configure charts in Superset for Data Insights tab		100%			Superset features and visualizations	14
[Stage 3] Feature engineering	Transform, encode, and prepare features for ML			100%		`data/train.json`, `data/val.json`	4
[Stage 3] Model training	Train ML models with hyperparameter tuning (LogReg, RF)			100%		Trained models in `models/` folder	6
Model evaluation	Evaluate trained models and save predictions and metrics			100%		Predictions and `evaluation.csv`	1
[Stage 4] Dashboard creation	Build Superset dashboard with 3 tabs, add descriptive text and organize charts	70%	15%	5%	10%	Final published dashboard	3
Final report writing	Write the full report and presentation slides	10%	10%	10%	70%	`team15_report.pdf`, `presentation.pdf`	10
Code quality checking	Run Pylint, document scripts, validate reproducibility of pipeline stages				100%	Fixed files	2
Totally spent hours							73

Table 1.2: Individual contributions of team members and associated deliverables

10 References

- Our GitHub repository
- Kaggle Dataset: Domestic US Air Flights 2016–2018
- US DOT BTS (Flight Delay Causes): BTS Delay Explanation