Machine Learning Approach for Predicting Flight Arrival Status and Mitigating Delays at Syracuse Airport

S Mahmudul Hasaan, Srinidhi Nukala, Siddhesh Save, Prathik Reddy Sannapureddy
{shasan, srnukala, ssave01, psannapu}@syr.edu

Problem Statement: The cited goal is developing models for predicting for airline postponements of flights, with a focus on airplanes coming into Syracuse (SYR) from significant gateways including Chicago (ORD), New York (JFK), and Orlando (MCO). In the current research, the airplane departure situation of the linked flights (UA 1400, B6 2516, B6 656) that occurred anywhere from one to four days prior is used to anticipate the arrival status (early, on-time, delayed) of future aircraft (AA 3324, DL 5371, WN 1305). Using past flight records from datasets including real-time flight-tracking websites and the Bureau of Transportation Statistics (BTS), the study develops and evaluates forecasting models. The overarching aim is to provide accurate forecasts that can optimize airline operations and improve on-time performance for Syracuse arrivals.

***Abstract** : Accurately predicting flight arrival times and managing delays remains an ongoing challenge for the airline industry. This study aims to analyze the departure status of incoming flights from major airports like Chicago O'Hare (ORD), Orlando (MCO), and New York JFK (JFK) to forecast the arrival status (early, on-time, or delayed) of specific aircraft into Syracuse Hancock International Airport (SYR). By leveraging data from the Bureau of Transportation Statistics (BTS), the research develops predictive models that account for factors that may influence flight disruptions, including weather conditions. Employing machine learning techniques, the models forecast aircraft arrival outcomes and provide recommendations to improve airline on-time performance and overall service quality. Overcoming flight delays through advanced predictive analytics is a key focus of this comprehensive study.*

I. INTRODUCTION

The airline industry is extremely time-sensitive, with on-time arrivals being crucial for customer satisfaction and operational efficiency - even a single delayed flight can cause cascading disruptions. Accurate predictive models are therefore essential tools for effective airline operations management. This research focuses on forecasting the arrival status (on-time, early, or delayed) of individual aircraft into Syracuse Hancock International Airport (SYR) based on the departure status of incoming flights from major hubs like Chicago O'Hare (ORD), New York JFK (JFK), and Orlando (MCO). By anticipating arrival delays, airlines can be proactive in minimizing passenger inconvenience, making better operational decisions, and improving the overall travel experience.

Based on previous related aircraft departure statuses from pre-designated airports, we project arrival times for flights into SYR, UA 1400, AA 3324, B6 2516, DL 5371, B6 656, and WN 1305. To achieve this goal, we build predictive models using real-time flight-tracking technology and historical flight data from a dependable source - the Bureau of Transportation Statistics (BTS) database. Since conditions are believed to possess a significant impact on the duration of flights, we also consider external factors, such as weather forecasts which includes sea level, temperature, wind speed, visibility, wind direction for the airports of departure and arrival.

This study showcases the potential of machine learning techniques like Logistic Regression, AdaBoost Classifier, Gradient Boosting Classifier, Random Forest Classifier, and XGBoost Classifier to accurately predict flight conditions, addressing the inherent challenges of flight control and airline planning. By developing robust predictive models, the study aims to empower airlines with data-driven insights to make informed decisions that minimize disruptions and enhance overall service quality. In its final stages, the study employs techniques like K-Fold Cross-Validation and Ensemble Voting Classifiers to ensure model robustness and optimize prediction accuracy, advancing the overarching goal of improving airline operational efficiency, reliability, and ability to proactively manage flight operations in the face of dynamic operational hurdles, ultimately delivering a superior travel experience for passengers.

II. WORKFLOW

We follow the typical data mining strategy in our study which is shown Figure 1. At a high level, we collect the data from various sources and preprocess them to conform to a general data format. Then we explore what features can be used as well as what features can be engineered. Finally, we train a wide variety of machine learning models with cross validation to figure out which works best. Finally, we select a final model according to prior analysis and evaluate its performance.

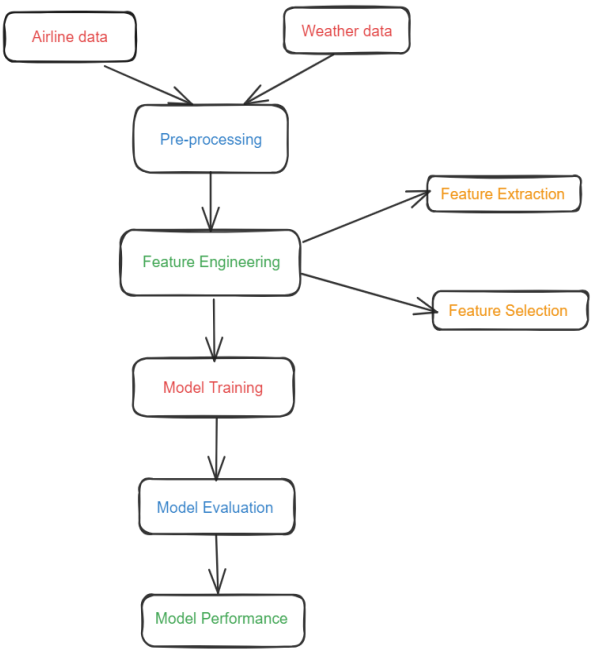


Fig 1: Airline data before preprocessing

III. DATA COLLECTION

We utilized airline data obtained from the U.S. Department of Transportation's Bureau of Transportation Statistics (BTS) website, specifically the On-Time Performance dataset available from 2010 to 2023. However, we excluded data from the years 2020 and 2021 due to their status as COVID-affected years. Our weather data sources include weather forecast data from Visual Crossing and historical weather data from the National Centers for Environmental Information (NCEI).

The airline data consists of various columns detailing flight information, such as scheduled departure and arrival times, carrier codes, flight numbers, and delay information. We omitted columns related to real-time flight information, such as actual arrival times and carrier delays, as they are not available for prediction purposes and would not contribute to our analysis purposes and would not contribute to our analysis.

Similarly, the weather data encompasses a range of meteorological parameters, including temperature, wind speed, visibility, and precipitation forecasts. We selected relevant weather features for our predictive modeling while disregarding columns that do not offer predictive value, such as live weather conditions during the flight.

Finally, we curated airline and weather datasets, excluding irrelevant columns to focus on predictive features conducive to accurate delay predictions. This streamlined approach ensures that our predictive modeling is based on pertinent information, enhancing the effectiveness of our analyses.

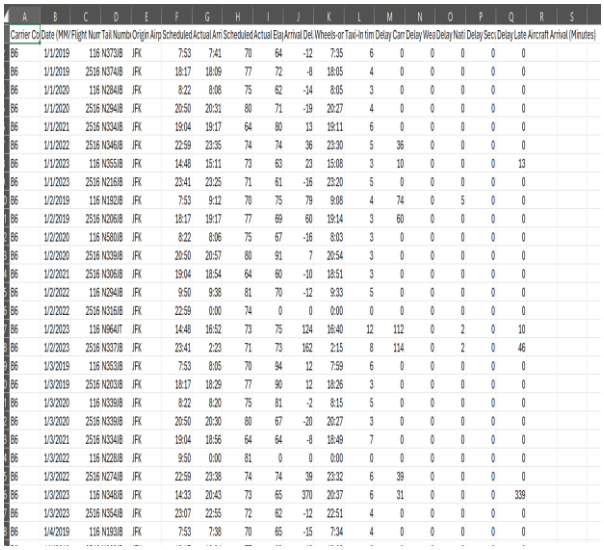


Fig 2: Airline data before preprocessing

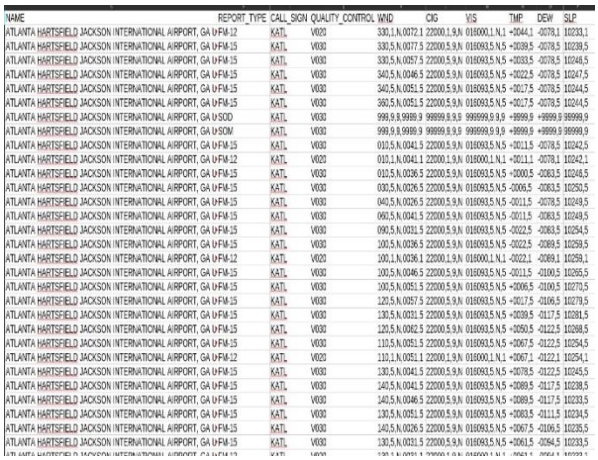


Fig 3: Weather data before preprocessing

IV. DATA PRE-PROCESSING AND CLEANING

In the data preprocessing and cleaning phase, we performed rigorous feature selection by eliminating irrelevant attributes from the original dataset. This process aimed to retain only the essential features crucial for subsequent analysis and modeling tasks. Furthermore, we transformed certain columns to more usable formats and units, aligning them with the requirements of our analytical methodologies. This meticulous approach served two key purposes: reducing the dataset's dimensionality and converting disparate data types into standardized formats conducive for further analysis. By curating the dataset through judicious feature

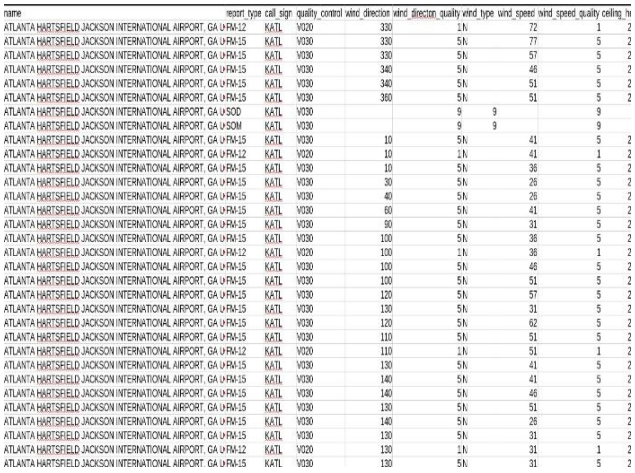
selection and transformation, we ensured a lean, optimized feature space tailored for robust predictive modeling.

We developed a Python solution to comprehensively manage weather data sourced from NOAA, automating key processes like data retrieval, parsing, cleaning, and preprocessing. Initially, we initiated data acquisition by fetching weather data files from NOAA's repository, targeting specific airports and years, utilizing *urllib.request* to download files via HTTP requests and stored them in a designated directory. The solution then handled parsing and cleaning the downloaded data files, extracting relevant weather information while handling inconsistencies and missing values, and performed preprocessing tasks like data formatting and normalization to prepare the weather data for integration with other datasets and subsequent analysis, streamlining the entire workflow for efficient and consistent handling of weather data from NOAA.

Moving into the parsing phase, we systematically extracted pertinent information from the downloaded weather data files. Through custom parsing functions tailored to NOAA's data format, we deciphered crucial weather parameters such as wind direction, speed, visibility, temperature, dew point, and sea level pressure. This parsing transformed the raw, unstructured data into a structured format suitable for analysis, facilitating the extraction of meaningful insights. Subsequently, we transitioned to cleaning and preprocessing, essential steps to ensure data quality and reliability. Addressing missing values was paramount, as they could significantly affect analytical outcomes. To remedy this, we employed an interpolation technique to fill missing values in the dataset. By averaging neighboring non-null values, we enhanced data completeness and integrity, preparing it for further analysis with increased confidence.

Following preprocessing, the processed and cleaned weather data was saved into CSV files for easy access and utilization in subsequent analyses. We organized the data into individual CSV files for each airport, simplifying data management and integration with various analytical tools. By storing the processed data in CSV format, we ensured its accessibility and usability across different analytical workflows and applications.

Our solution epitomized a systematic approach to handling weather data, encompassing multiple stages from acquisition to storage. It streamlined the complexities of working with NOAA's weather data, automating tedious tasks, and allowing researchers and analysts to focus on extracting insights. With its modular design and comprehensive functionality, our solution served as a versatile tool for leveraging weather data in diverse fields, facilitating informed decision-making, and enhancing understanding of environmental trends.



name	report_type	call_sign	quality_control	wind_direction	wind_direction_quality	wind_type	wind_speed	wind_speed_quality	ceiling_he
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-12	KATL	V020		330	1	N	72	1	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		330	5	N	77	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		330	5	N	57	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		340	5	N	46	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		340	5	N	51	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		360	5	N	51	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (SOD	KATL	V030			9	9	9		
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (SOM	KATL	V030			9	9	9		
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		10	5	N	41	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-12	KATL	V020		10	1	N	41	1	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		10	5	N	36	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		30	5	N	26	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		40	5	N	26	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		60	5	N	41	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		90	5	N	31	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		100	5	N	36	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-12	KATL	V020		100	1	N	36	1	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		100	5	N	46	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		100	5	N	51	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		120	5	N	57	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		130	5	N	31	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		120	5	N	62	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		110	5	N	51	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-12	KATL	V020		110	1	N	51	1	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		130	5	N	41	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		140	5	N	41	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		140	5	N	46	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		130	5	N	51	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		140	5	N	26	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		130	5	N	31	5	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-12	KATL	V020		130	1	N	31	1	2
ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPORT, GA (ARM-15	KATL	V030		130	5	N	31	5	2

Fig 4: Weather data after preprocessing

Then, we aimed to enrich a flight dataset with weather data, particularly focusing on flights departing from Syracuse (SYR) airport. The process involved several steps:

Data Preparation: Initially, we loaded the flight dataset from a CSV file and performed necessary preprocessing steps, such as sorting the data based on date and scheduled arrival time and dropping irrelevant columns like 'Arrival Delay (Minutes)' and 'Flight Number'.

Weather Data Integration: We retrieved weather data for SYR airport from a designated directory and aligned it with the flight dataset based on timestamps. To ensure data completeness and reliability, we employed a mode imputation technique to handle missing values within the weather data.

Nearest Weather Data Selection: We implemented a binary search algorithm to find the nearest weather data rows corresponding to each flight's timestamp. This ensured that the weather data used for enrichment was as close to the flight time as possible, enhancing the accuracy of the analysis.

Enriched Dataset Creation: Finally, we constructed a new dataset that incorporated both flight information and imputed weather data for SYR airport. Each flight's attributes were augmented with corresponding weather features, such as wind speed, temperature, and sea level pressure. The resulting dataset was well-prepared for further analysis, offering insights into the relationship between flight performance and weather conditions specific to SYR airport.

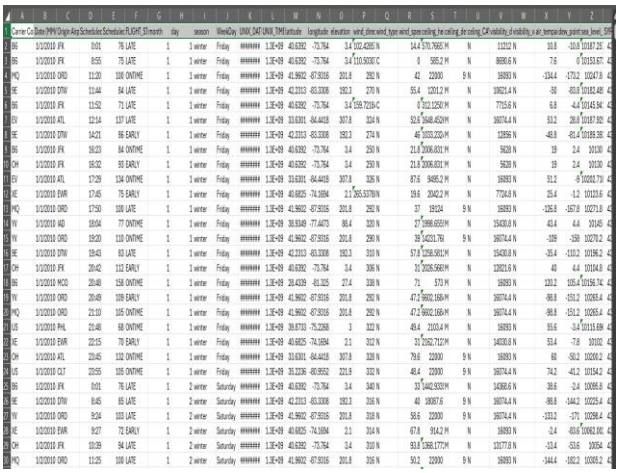


Fig 5: Combined data after preprocessing

V. FEATURE ENGINEERING

In preparing the data for modeling, we undertook a series of feature engineering steps to enhance the predictive capability of our models. The feature engineering process involved several key steps, as outlined below:

1. Categorical Time Conversion

We converted the scheduled arrival time into categorical variables representing different time periods of the day, such as Late Night, Early Morning, Morning, Late Morning, Noon, Afternoon, Evening, and Night. This categorization facilitated the incorporation of time-of-day patterns into our predictive models. The conversion was performed using a custom function that assigned categorical labels based on the hour of the scheduled arrival time.

2. Date-Based Feature Addition

Additional features were derived from the date column to capture temporal patterns and seasonal variations. Features such as day, month, season, and Weekday were extracted, providing the models with contextual information to better predict flight arrival status. The inclusion of these features enriches the dataset with temporal context, aiding in the prediction of flight delays and arrival times.

3. Feature Selection

A subset of relevant features was selected for model training based on their potential impact on flight arrival predictions. Features such as latitude, longitude, elevation, wind direction, wind speed, ceiling height, visibility distance, air temperature, dew point temperature, and sea level pressure were retained for further analysis. These features were deemed critical in capturing the underlying factors influencing flight punctuality and weather conditions.

4. Correlation Analysis

A correlation analysis was conducted to identify relationships between features and assess multicollinearity. This analysis revealed high correlations between certain pairs of features, such as air temperature and dew point temperature, as well as elevation and longitude. The observed correlations informed subsequent feature selection and model development decisions, ensuring the inclusion of independent and informative features.

5. Column Dropping

Columns exhibiting high correlations or redundancy were dropped from the dataset to mitigate multicollinearity and streamline model training. Features such as date, scheduled arrival time, longitude, dew point temperature, and various wind-related attributes were removed to enhance model interpretability and performance.

6. Unit Conversion

To ensure consistency across historical and future data, unit conversions were applied to standardize the measurements. Speed measurements were converted from meters per second to miles per hour, distances from meters to miles, temperatures from Celsius to Fahrenheit, and pressure from hectopascals to millibars. These conversions harmonized the units across different datasets, enabling seamless integration and model training.

7. One-Hot Encoding

Categorical variables were encoded using one-hot encoding to represent discrete categories as binary vectors. This transformation expanded categorical features into multiple binary columns, with each column indicating the presence or absence of a category. By dropping the first column of each one-hot encoded feature set, we mitigated multicollinearity issues and ensured compatibility with machine learning algorithms.

8. Train-Test Split and Data Scaling

The dataset was divided into training and testing sets using a stratified split to preserve class distribution. The data was then scaled using a standard scaler to normalize feature values and prevent certain features from dominating others during model training. Standard scaling ensures that each feature has a mean of zero and a standard deviation of one, facilitating optimal model performance across different features.

This comprehensive feature engineering process laid the groundwork for robust and accurate predictive modeling of flight arrival status, leveraging a combination of temporal,

spatial, and meteorological features to capture underlying patterns and trends.

VI. METHODOLOGY

In approaching the problem, we adopt a two-model strategy. Firstly, our initial model focuses solely on predicting the arrival time of flights based on relevant weather features. This serves as a foundational step in understanding the impact of weather conditions on flight punctuality.

Subsequently, our second model incorporates not only weather data but also the arrival status of preceding flights to forecast the arrival time of the subsequent scheduled flight. This model employs a 'Hopping' logic, wherein we explore three distinct scenarios which we call: 1-Hop, 2-Hop, and 3-Hop.

In the 1-Hop approach, we utilize data from only the immediately preceding flight to inform our prediction. Building upon this, the 2-Hop approach considers information from the two most recent flights, aiming to enhance prediction accuracy. Similarly, the 3-Hop approach extends this analysis to include data from the three most recent flights, anticipating further refinement in prediction outcomes.

Although our initial assumption suggested that the 3-Hop model would yield superior accuracy, it became apparent that this approach was susceptible to overfitting. While the model demonstrated high accuracy during training, its predictive performance on unseen data might have been compromised due to an excessive reliance on training data. Consequently, to mitigate the risk of overfitting and ensure a more balanced performance, we elected to proceed with the 2-Hop approach, which offered a pragmatic balance between predictive accuracy and model generalization.

To ascertain the optimal predictive framework, we conducted an extensive evaluation of various classifiers, including Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, Gradient Boosting, Naive Bayes, AdaBoost, and XGBoost Classifiers. Through iterative experimentation and cross-validation, we identified and fine-tuned the most effective parameters for each classifier, culminating in a curated selection of hyperparameters poised to optimize predictive performance. Leveraging these refined settings, we utilized the VotingClassifier ensemble method to construct a hybrid model integrating Logistic Regression, Adaboost, Gradient Boosting, Random Forest, and XGBoost Classifiers. This amalgamation of diverse modeling techniques was based on their demonstrated efficacy in our experimentation. In configuring the VotingClassifier, we assigned varying weights to each constituent classifier, with XGBoost receiving the highest weight and Logistic Regression the least. This weighting scheme reflects our confidence in the predictive capabilities of each model, with XGBoost prioritized due to its superior performance in our evaluation. By assigning weights, we aim to capitalize on the strengths of individual classifiers and enhance the overall predictive accuracy of the ensemble. Subsequently, we rigorously evaluated the final model's accuracy, precision, recall, and F1-score to validate its predictive capability and robustness.

VII. MODEL EVALUATION

- The fit_and_evaluate function facilitates model training and evaluation. It employs label encoding to transform categorical labels into numerical representations, mitigating potential errors in certain classification algorithms.
- Upon fitting the model to the training data, predictions are generated for the test dataset.
- Evaluation metrics including accuracy, precision, recall, and F1-score are computed using the classification_report function from scikit-learn.

```
def fit_and_evaluate(model, trainX, trainY, testX, testY):
    # Convert string labels to integers
    label_encoder = LabelEncoder()
    trainY_encoded = label_encoder.fit_transform(trainY)
    testY_encoded = label_encoder.transform(testY)

    model.fit(trainX, trainY_encoded)
    testY_pred = model.predict(testX)
    accuracy = accuracy_score(testY_encoded, testY_pred)
    report = classification_report(testY_encoded, testY_pred, output_dict=True)
    results = {'accuracy': accuracy, 'classification_report': report}
    return results
```

Fig 6 : fit and evaluate method

Hyperparameter Tuning:

- Hyperparameter tuning is conducted using the hyperparameter_tuning function, leveraging Grid Search Cross Validation (GridSearchCV) to systematically explore a range of hyperparameter combinations.
- The KFold cross-validator partitions the dataset into training and validation subsets, aiding in the robust assessment of model performance.
- Various classifiers, namely Logistic Regression, AdaBoost, Gradient Boosting, Random Forest, and XGBoost, are instantiated with predefined hyperparameters. These classifiers offer distinct advantages and are expected to contribute to the ensemble's predictive capability.

```
from sklearn.model_selection import GridSearchCV, KFold
def hyperparameter_tuning(model, param_grid, trainX, trainY, cv=5):
    label_encoder = LabelEncoder()
    trainY_encoded = label_encoder.fit_transform(trainY)
    # Initialize K-Fold cross-validator
    kf = KFold(n_splits=cv, shuffle=True, random_state=42)

    # Perform grid search
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=kf,
 scoring='accuracy', verbose=3, n_jobs=10, )
    grid_search.fit(trainX, trainY_encoded)

    return grid_search
```

Fig 7 : Hyperparameter tuning

Ensemble Learning:

- Ensemble learning is employed to combine predictions from multiple base classifiers. The VotingClassifier aggregates predictions using a soft voting mechanism, weighted according to the specified weights.
- The weights assigned to individual classifiers reflect their relative importance in the ensemble, aiming to capitalize on the strengths of each base model.

Model Configuration:

- Logistic Regression (lr), AdaBoost (ada), Gradient Boosting (gbm), Random Forest (rf), and XGBoost (xgb) classifiers are initialized with predetermined hyperparameters, attuned through experimentation.
- The VotingClassifier (votingCLF) combines the predictions of these classifiers using a weighted averaging scheme, with weights assigned based on their anticipated contributions to the ensemble's overall predictive accuracy.

VIII. PERFORMANCE AND RESULTS

- The fit_and_evaluate function is invoked to train the ensemble model (votingCLF) on the training dataset (enc_trainX, trainY) and evaluate its performance on the test dataset (enc_testX, testY).
- Evaluation metrics such as accuracy, precision, recall, and F1-score are computed and reported, providing insights into the ensemble's efficacy in predicting flight arrival times.

```
from sklearn.ensemble import VotingClassifier

lr = LogisticRegression(penalty='l2', C=50, max_iter=1500, solver='saga')
ada = AdaBoostClassifier(**{'algorithm': 'SAMME.R', 'learning_rate': 1.0,␣
 ↪'n_estimators': 200})
gbm = GradientBoostingClassifier(**{'learning_rate': 0.1, 'max_depth': 5,␣
 ↪'n_estimators': 200})
rf = RandomForestClassifier(** {'bootstrap': False, 'criterion': 'entropy',␣
 ↪'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 4,␣
 ↪'min_samples_split': 2, 'n_estimators': 200})
xgb = XGBClassifier(**{'colsample_bytree': 0.7, 'gamma': 0.1, 'learning_rate':␣
 ↪0.1, 'max_depth': 7, 'n_estimators': 200, 'reg_lambda': 1.5, 'subsample': 0.
 ↪9},objective='multi:softprob')

votingCLF = VotingClassifier(
    estimators=[
        ('rf', rf),
        ('ada', ada),
        ('xgb', xgb),
        ('lr', lr),
        ('gbm', gbm)
    ],
    voting='soft',
    weights=[
        4,
        5,
        7,
        2,
        6]
)
fit_and_evaluate(votingCLF, enc_trainX, trainY, enc_testX, testY)
```

Fig 8: Models and methods

Initial Model Evaluation:

For the initial hybrid model incorporating weather features and flight arrival data, we achieved the following performance metrics:

- Accuracy: 53.54%
- Precision:
  - Early (0): 55.13%
  - Late (1): 49.96%
  - On-time (2): 46.15%
- Recall:
  - Early (0): 85.99%
  - Late (1): 41.53%
  - On-time (2): 6.48%
- F1-score:
  - Early (0): 67.19%
  - Late (1): 45.36%
  - On-time (2): 11.36%

The weighted average metrics provide an overall assessment of model performance, accounting for class imbalances within the dataset. The weighted average precision, recall, and F1-score are computed as follows:

- Weighted Average Precision: 51.42%
- Weighted Average Recall: 53.54%
- Weighted Average F1-score: 47.22%


```
{'accuracy': 0.535385968770618,
 'classification_report': {'0': {'precision': 0.5512946373788195,
   'recall': 0.8599732006125574,
   'f1-score': 0.671876168399013,
   'support': 10448.0},
  '1': {'precision': 0.49964726631393297,
   'recall': 0.4152741131632952,
   'f1-score': 0.4535702849823887,
   'support': 6822.0},
  '2': {'precision': 0.46153846153846156,
   'recall': 0.06477584629460201,
   'f1-score': 0.11360718870346598,
   'support': 5465.0},
  'accuracy': 0.535385968770618,
  'macro avg': {'precision': 0.504160121743738,
   'recall': 0.4466743866901515,
   'f1-score': 0.4130178806949559,
   'support': 22735.0},
  'weighted avg': {'precision': 0.5142215840965582,
   'recall': 0.535385968770618,
   'f1-score': 0.4721742677742329,
   'support': 22735.0}}}
```


Fig 9: Initial model evaluation

Hopping Model Performance:

We employed a series of 'Hopping' models to predict flight arrival times, building upon the initial hybrid model's predictions. Each hopping model leverages weather data and predictions from preceding flights, offering nuanced insights into arrival time forecasts.

1-Hop Model:

- Accuracy: 53.51%
- Precision:
  - Early (0): 55.24%
  - Late (1): 49.79%
  - On-time (2): 44.83%
- Recall:
  - Early (0): 85.91%
  - Late (1): 41.62%
  - On-time (2): 6.42%
- F1-score:
  - Early (0): 67.24%
  - Late (1): 45.18%
  - On-time (2): 11.24%
- Weighted Average Precision: 51.27%
- Weighted Average Recall: 53.51%
- Weighted Average F1-score: 47.12%

2-Hop Model:

- Accuracy: 61.44%
- Precision:
  - Early (0): 59.68%
  - Late (1): 62.48%
  - On-time (2): 80.31%
- Recall:
  - Early (0): 90.30%
  - Late (1): 52.85%
  - On-time (2): 16.98%
- F1-score:
  - Early (0): 71.87%
  - Late (1): 57.26%
  - On-time (2): 28.03%
- Weighted Average Precision: 65.48%
- Weighted Average Recall: 61.44%
- Weighted Average F1-score: 56.95%

3-Hop Model:

- Accuracy: 66.52%
- Precision:
  - Early (0): 62.64%
  - Late (1): 70.21%
  - On-time (2): 92.00%
- Recall:
  - Early (0): 92.87%
  - Late (1): 58.82%
  - On-time (2): 25.75%
- F1-score:
  - Early (0): 74.82%
  - Late (1): 64.01%
  - On-time (2): 40.24%
- Weighted Average Precision: 72.91%
- Weighted Average Recall: 66.52%
- Weighted Average F1-score: 63.26%

The hybrid model serves as the foundation for predicting flight arrival times, leveraging weather features to provide initial forecasts. Subsequent hopping models refine these predictions by incorporating previous flight data, offering insights into potential variations in arrival times.

The 3-Hop model exhibits the highest accuracy at 66.52%, suggesting a promising predictive capability. However, this model may be susceptible to overfitting, as indicated by the discrepancy between training and test data. Consequently, we favor the 2-Hop model as a balanced compromise between predictive accuracy and model generalization, ensuring reliable performance on unseen data.

By assigning weights to constituent classifiers in the hybrid model, we aim to capitalize on their strengths and enhance overall predictive accuracy. The weighted average metrics provide a comprehensive assessment of model performance, accounting for class imbalances within the dataset.

IX. CONCLUSION

In our study, we tackled the crucial task of accurately predicting flight arrival times and managing delays in the airline industry. By leveraging machine learning techniques and integrating weather data with historical flight records, we aimed to optimize airline operations and enhance customer satisfaction. Our research focused on forecasting flight arrival statuses into Syracuse (SYR) from major airports like Chicago (ORD), New York (JFK), and Orlando (MCO), providing actionable insights to airlines to improve operational efficiency and reliability.

We developed predictive models, starting with a hybrid model that incorporated weather features and flight arrival data, which served as the foundation for subsequent hopping models. Through rigorous experimentation and evaluation, we identified the 2-Hop model as a balanced compromise between predictive accuracy and generalization, offering reliable predictions while mitigating overfitting risks. Our approach involved extensive feature engineering and ensemble learning techniques, ensuring that our models were based on relevant features and collectively provided robust forecasting tools for airlines. Overall, our study contributes to ongoing efforts to optimize airline operations and improve service quality, empowering airlines with valuable insights to make informed decisions and enhance operational resilience.

REFERENCES

[1]. Sudha Tushara Sadasivuni, "Analysing Tweets for Predicting Mental Health States Using Data Mining And Machine Learning Algorithms", *Georgia State University*, 2021.

[2]. Muhammad Mujahid, Ernesto Lee, Furqan Rustam, Patrick Bernard Washington, Saleem Ullah, Aijaz Ahmad Reshi, Imran Ashraf, "Sentiment Analysis and Topic Modeling on Tweets about Online Education during COVID-19", *MDPI*, 2021.

[3]. Rainer Simon, Dražen Ignjatović, Georg Neubauer, Clemens Gutschi, Johannes Pan, Siegfried Vössner, "Applying data mining techniques in the context of social media to improve situational awareness at large-scale events", *International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME) 7-8 October 2021, Mauritius.*

[4]. Liu, B. (2012). Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies, 5(1), 1-167.

[5]. Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135.

[6]. Sentiment analysis of Amazon product reviews based on NLP by Yuanhang Xiao, Chengbin Qi, Hongyong Leng

[7]. Mining Product Reviews: A Comparison of Essentiality-Based and Summarization-Based Approaches by M. R. Sayyed and N. A. Memon.