Machine Learning Project on Flight Route Optimization Final Report

Gizem EROL, Selvinaz Zeynep KIYIKCI, Zeynep Sude KIRMACI
(CEN435 Project, Autumn 2024)

In this study, we introduce an optimized flight route planning system using machine learning to minimize travel time, costs, and environmental effects with enhanced operation. The system can input historical flight data and the locations of airports together with weather conditions for the analysis and optimization of flight routes using algorithms such as Linear Regression, Logistic Regression, Decision Tree, Naïve Bayes, and Support Vector Machine (SVM). The algorithms are then tested for their best predictive accuracy and efficiency in determining the shortest, cheapest route. The system, through a combination of distance, flight duration, and weather impacts, offers reliable and optimized route recommendations. This work highlights the transformation that machine learning could bring to flight route optimization. Moreover, it offers a solution for real-world aviation challenges.

Keywords: Flight Route Optimization, Machine Learning in Aviation, Flight Delay Prediction, Naïve Bayes Classifier, Graph-Based Optimization, Route Planning with Machine Learning, Delay Prediction Models, Optimization, Support Vector Machine (SVM), Shortest Path Algorithm

Introduction

In today's modern aviation industry, with the pressure to cut costs and improve customer satisfaction, on-time performance with minimal delays is essential and therefore efficient route optimization is critical. That is the perspective from which this project set out to predict flight delays and optimize flight routes using machine learning models. In particular, it developed a framework from historical flight data to predict delays and then determine optimal paths between airports. After experimenting with several machine learning models and benchmarking their performances, it finally identified one that best suited its objectives.

Related Works

Flight delay prediction and route optimization are critical challenges in aviation. Junghyun Kim's PhD dissertation, “A Data-Driven Approach Using Machine Learning for Real-Time Flight Path Optimization”, highlights the importance of data-driven machine learning approaches in real-time optimization of flight paths. This work demonstrates the applicability and effectiveness of machine learning algorithms in optimizing flight paths, especially under dynamic conditions. Kim's work provides a robust framework for the analysis of large-scale data in real-time decision-making processes for flight path optimization considering parameters such as air traffic, weather conditions and fuel consumption. This thesis serves as a valuable reference to support the potential of machine learning algorithms such as SVM used in our current project for flight path optimization [1].

Furthermore, Sruti Oza et al. proposed a system titled “Flight Delay Prediction System Using Weighted Multiple Linear Regression” that focuses on predicting flight delays by analyzing historical data and identifying influential factors such as weather conditions, air traffic control, and mechanical failures. By combining weighted regression techniques and real-time weather data, the system achieved significant improvements in the accuracy of delay predictions.

Building on similar foundations, our work extends the application of machine learning to flight route optimization. While the work of Oza et al. primarily addresses delay prediction, we experimented with various machine learning algorithms - Linear Regression, Logistic Regression, Decision Tree, Naïve Bayes, and Support Vector Machines (SVM) - to see the optimized results of flight routes. By considering additional factors such as travel time, cost, and environmental impact, our approach provides a comprehensive solution for real-world aviation challenges.

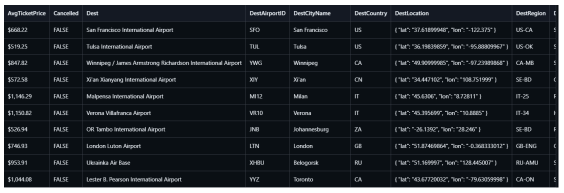
Insights from the work of Oza et al. underline the importance of historical data and external variables in predictive modeling, which is closely aligned with our methodology. However, our system distinguishes itself by focusing on route optimization rather than delay prediction and evaluating multiple algorithms for their efficiency in generating the shortest, most cost-effective and environmentally sustainable routes.

This synergy between delay prediction and route optimization demonstrates the transformative potential of machine learning in aviation and highlights the complementary nature of these approaches in improving operational efficiency. [2]

Dataset and Features

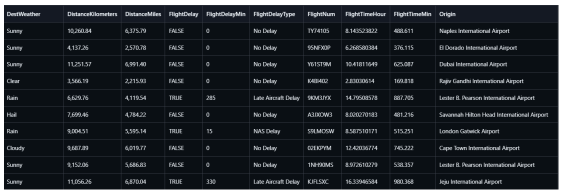
The primary data source for this project is the "**flights.csv**" dataset available on GitHub [3]. This dataset contains detailed information about domestic and international flights, providing data on various features related to each flight. These features include ticket prices, origin and destination airports, weather conditions, flight delays, distances, and flight durations. Below is a brief explanation of the dataset's key fields:

- **AvgTicketPrice**: Average ticket price for the flight.
- **Cancelled**: Indicates whether the flight was canceled or not.
- **Dest and Origin**: Destination and origin airports, including their names, IDs, and geographic coordinates (latitude and longitude).
- **DestWeather and OriginWeather**: Weather conditions at the destination and origin airports.
- **DistanceKilometers and DistanceMiles**: The distance between the origin and destination airports in kilometers and miles.
- **FlightDelay and FlightDelayType**: Indicates whether the flight was delayed and the reason for the delay, if applicable.
- **FlightTimeHour and FlightTimeMin**: Flight duration in hours and minutes.
- **dayOfWeek and hour_of_day**: The day of the week and time of departure.



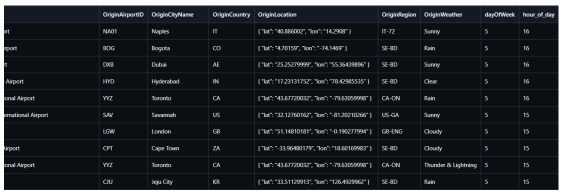
AvgTicketPrice	Cancelled	Dest	DestAirportID	DestCityName	DestCountry	DestLocation	DestRegion	D
$668.22	FALSE	San Francisco International Airport	SFO	San Francisco	US	{ "lat": "37.61899948", "lon": "-122.375" }	US-CA	S
$519.25	FALSE	Tulsa International Airport	TUL	Tulsa	US	{ "lat": "36.19839859", "lon": "-95.88809967" }	US-OK	S
$847.82	FALSE	Winnipeg / James Armstrong Richardson International Airport	YWG	Winnipeg	CA	{ "lat": "49.90999985", "lon": "-97.23989868" }	CA-MB	S
$572.58	FALSE	Xi'an Xianyang International Airport	XIY	Xi'an	CN	{ "lat": "34.447102", "lon": "108.751999" }	SE-BD	C
$1,146.29	FALSE	Malpensa International Airport	MI12	Milan	IT	{ "lat": "45.6306", "lon": "8.72811" }	IT-25	F
$1,150.82	FALSE	Verona Villafranca Airport	VR10	Verona	IT	{ "lat": "45.395699", "lon": "10.8885" }	IT-34	H
$526.94	FALSE	OR Tambo International Airport	JNB	Johannesburg	ZA	{ "lat": "-26.1392", "lon": "28.246" }	SE-BD	F
$746.93	FALSE	London Luton Airport	LTN	London	GB	{ "lat": "51.87469864", "lon": "-0.368333012" }	GB-ENG	C
$953.91	FALSE	Ukrainka Air Base	XHBU	Belogorsk	RU	{ "lat": "51.169997", "lon": "128.445007" }	RU-AMU	S
$1,044.08	FALSE	Lester B. Pearson International Airport	YYZ	Toronto	CA	{ "lat": "43.67720032", "lon": "-79.63059998" }	CA-ON	S

Figure 1.First 10 row 8 column on the dataset



DestWeather	DistanceKilometers	DistanceMiles	FlightDelay	FlightDelayMin	FlightDelayType	FlightNum	FlightTimeHour	FlightTimeMin	Origin
Sunny	10,260.84	6,375.79	FALSE	0	No Delay	TY74105	8.143523822	488.611	Naples International Airport
Sunny	4,137.26	2,570.78	FALSE	0	No Delay	95NFX0P	6.268580384	376.115	El Dorado International Airport
Sunny	11,251.57	6,991.40	FALSE	0	No Delay	Y61ST9M	10.41811649	625.087	Dubai International Airport
Clear	3,566.19	2,215.93	FALSE	0	No Delay	K4BI402	2.83030614	169.818	Rajiv Gandhi International Airport
Rain	6,629.76	4,119.54	TRUE	285	Late Aircraft Delay	9KM3JYX	14.79508578	887.705	Lester B. Pearson International Airport
Hail	7,699.46	4,784.22	FALSE	0	No Delay	A3JXOW3	8.020270183	481.216	Savannah Hilton Head International Airport
Rain	9,004.51	5,595.14	TRUE	15	NAS Delay	S9LMOSW	8.587510171	515.251	London Gatwick Airport
Cloudy	9,687.89	6,019.77	FALSE	0	No Delay	02EKPYM	12.42036774	745.222	Cape Town International Airport
Sunny	9,152.06	5,686.83	FALSE	0	No Delay	1NH90MS	8.972610279	538.357	Lester B. Pearson International Airport
Sunny	11,056.26	6,870.04	TRUE	330	Late Aircraft Delay	KJFLSXC	16.33946584	980.368	Jeju International Airport

Figure 2.First 10 row and 10 column on dataset



	OriginAirportID	OriginCityName	OriginCountry	OriginLocation	OriginRegion	OriginWeather	dayOfWeek	hour_of_day
rt	NA01	Naples	IT	{ "lat": "40.886002", "lon": "14.2908" }	IT-72	Sunny	5	16
rport	BOG	Bogota	CO	{ "lat": "4.70159", "lon": "-74.1469" }	SE-BD	Rain	5	16
t	DXB	Dubai	AE	{ "lat": "25.25279999", "lon": "55.36439896" }	SE-BD	Sunny	5	16
Airport	HYD	Hyderabad	IN	{ "lat": "17.23131752", "lon": "78.42985535" }	SE-BD	Clear	5	16
ional Airport	YYZ	Toronto	CA	{ "lat": "43.67720032", "lon": "-79.63059998" }	CA-ON	Rain	5	16
ernational Airport	SAV	Savannah	US	{ "lat": "32.12760162", "lon": "-81.20210266" }	US-GA	Sunny	5	15
	LGW	London	GB	{ "lat": "51.14810181", "lon": "-0.190277994" }	GB-ENG	Cloudy	5	15
irport	CPT	Cape Town	ZA	{ "lat": "-33.96480179", "lon": "18.60169983" }	SE-BD	Cloudy	5	15
ional Airport	YYZ	Toronto	CA	{ "lat": "43.67720032", "lon": "-79.63059998" }	CA-ON	Thunder & Lightning	5	15
	CJU	Jeju City	KR	{ "lat": "33.51129913", "lon": "126.4929962" }	SE-BD	Rain	5	15

Figure 3.First 10 row and last 8 column on the dataset

This dataset is valuable for analyzing various aspects of flight operations, such as delays, ticket prices, and route optimization. We used it for developing predictive models, identifying trends, and improving overall efficiency for finding routes.

Methods and Experiments

Linear Regression

The Linear Regression model in this project was designed to predict the likelihood of flight delays as a binary classification problem. The model uses the flight distance (**`DistanceKilometers`**) as the primary predictor. The relationship between the predictor and the target variable is expressed through the linear regression equation:

$$y = w_0 + w_1 \cdot x_1$$

Here:

- y: Predicted output (probability of delay in this case, thresholded at 0.5 to classify as delayed or not delayed).

w0: Intercept (bias term).

w1: Coefficient representing the weight or importance of the predictor (**`DistanceKilometers`**).

x1: Value of the predictor (**`DistanceKilometers`**).

The dataset was preprocessed by normalizing numerical features such as flight distance. The target variable (**`DelayStatus`**) was derived by categorizing flights as delayed if their delay duration (**`FlightDelayMin`**) was greater than zero. The data was split into training (**70%**) and testing (**30%**) subsets, with additional cross-validation applied to improve reliability.

During model training, the weights (w0,w1) are learned by minimizing the Mean Squared Error (MSE) between the actual and predicted values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \acute{y}_i\right)^2$$

Where:

- y_i: Actual target value (0 or 1 for **`DelayStatus`**).
- $\acute{y}$: Predicted value from the model.
- n: Number of samples in the training set.

After training, the model generates predictions for new data by plugging the standardized input feature (x1) into the learned equation. For example, if the predicted y is greater than 0.5, the flight is classified as delayed; otherwise, it is classified as not delayed.

The dataset was preprocessed by normalizing numerical features, ensuring that the **`DistanceKilometers`** values were on a comparable scale. The target variable (**`DelayStatus`**) was derived by categorizing flights as delayed if their delay duration (**`FlightDelayMin`**) was greater than zero. Synthetic Minority Oversampling Technique (SMOTE) was used to address class imbalance in the training data.

While this Linear Regression model is interpretable and computationally efficient, its simplicity may limit its ability to capture complex relationships in the data, such as interactions between multiple predictors. Future iterations could integrate additional features like weather conditions and explore nonlinear models for better performance.

Overall, the Linear Regression model provides a straightforward approach to predict delays based on key flight attributes. However, its performance might be limited by the linearity assumption and the simplicity of the model compared to more complex machine learning methods that could capture nonlinear relationships in the data.

Logistic Regression

The Logistic Regression model created for this project aimed to predict the probability of flight delays as a binary classification problem. The target variable, DelayStatus, was constructed from the FlightDelayMin column and flights with delays greater than zero were classified as delayed (1), while those without delays were assigned as non-delayed (0). In the model, we used DistanceKilometers as the main estimator to narrow the problem coverage.

Before training the model, the dataset was preprocessed by standardizing the numeric feature (DistanceKilometers) with StandardScaler to ensure that all input values were on a uniform scale. Due to the class imbalance in the data (fewer delayed flights than non-delayed flights), we used Synthetic Minority Oversampling Technique (SMOTE) to balance the dataset, allowing the model to efficiently learn patterns from both classes.

The model was trained using 70% of the dataset, with the remaining 30% set aside for testing. To evaluate the generalization ability of our model, we used cross-validation using 5 folds. Evaluation metrics include accuracy, mean squared error (MSE) and a classification report detailing precision, recall and F1-score.

The trained model predicts the likelihood of a flight delay for new data by estimating the probability of the `DelayStatus` being `1` (delayed) using the logistic function:

$$P(y=1|x) = \frac{1}{1+e^{-\left(w_0+w_1 \cdot x_1\right)}}$$

Where:

- P(y=1 | x): Predicted probability of a delay.
- w_0: Intercept (bias term).
- w_1: Coefficient for the predictor `DistanceKilometers`.
- x_1: Standardized value of `DistanceKilometers`.

The prediction threshold is set to 0.5, which means that flights with a predicted probability above 0.5 are classified as delayed. In addition, the model is integrated into a graph-based approach where flight distance and delays are combined in a weighted graph. The graph is used to determine the shortest path (optimal route) between user-defined departure and destination airports that minimizes distance and delays.

Although logistic regression models are interpretable and computationally efficient, their simplicity may limit their ability to capture complex relationships in the data, such as interactions between multiple features. Future iterations could explore more complex models or incorporate additional features, such as weather conditions or flight times, to improve accuracy.

Decision Tree Classifier

The Decision Tree classifier predicts flight delay types based on various features, including destination and origin weather, flight delay duration, and delay type. The dataset is cleaned and transformed by changing the **AvgTicketPrice** column into a number, encoding categorical variables using LabelEncoder, and filling in missing values. The data is then split into training and testing sets, after which the model is built by fitting the training data to the classifier. Once the model finishes training, it is analyzed for its accuracy on the testing data as well as for a classification report. The decision tree is then plotted with finer formatting for reading for how the features drive the model in making predictions. Such an interpretation helps to understand the main factors that affect flight delays, covering their several categories.
We calculate entropy and information gain. Entropy is lowest, if there is at most one class present, and entropy is highest, if the proportions of all present classes are equal. That is, if all examples are positive or all negative, entropy is low (zero). If half are

positive and half are negative, entropy is high (1.0),

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data, called information gain. Values(A) is the set of all possible values for attribute A, Sv is the subset of S for which attribute A has value v,

$$G(S,A) = Entropy(S) - \sum_{v\in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Naïve Bayes

The Naive Bayes model was applied to predict flight delays for a binary classification problem, such as Delayed or Not Delayed. The target variable, DelayStatus, is extricated out of the FlightDelayMin column by labeling flights with a positive delay duration as delayed (1), and others as not delayed (0). The model, in turn, uses DistanceKilometers as its primary predictor to evaluate the effect it has on the probability of a delay.
The dataset was preprocessed by standardizing numerical features using StandardScaler. This is to ensure that all input values are on the same scale. A very severe class imbalance problem in the dataset was observed, with non-delayed flights many more than delayed ones. Therefore, we addressed this by using Synthetic Minority Oversampling Technique (SMOTE). This ensured that the model could learn patterns effectively from both classes during training.

It assumes independence between features and applies the Gaussian Naive Bayes algorithm for classification. The model was, therefore, trained on 70% of the dataset, and the rest was kept for testing. The cross-validation accuracy of the model was 0.52, after 5 folds. This predictive ability of the model was further probed with a test set, which returned an accuracy of 0.56 on the test set and an overall MSE of 0.25.

An improvement was carried out to include delay predictions into flight optimization by creating a directed graph where the airports are nodes and the flights between them are edges. Each edge has a weight according to the combination of flight distance and its likeliness of delay occurrence. It then computes the shortest path between an origin and a destination airport specified by the user using Dijkstra's algorithm, which minimizes the total weight (distance + delay).

While Naive Bayes offers a computationally efficient and interpretable model, its simplicity may limit its ability to capture complex relationships in the data. Future iterations could benefit from incorporating additional features such as weather conditions or airline-specific delays, and exploring advanced classification models for improved performance.

Support Vector Machine

In this project, Support Vector Machine (SVM) models were implemented to predict flight delays and optimize flight routes based on features such as **`DistanceKilometers`** and **`FlightDelayMin`**. Both **SVC (Support Vector Classification)** and **SVR (Support Vector Regression)** were used to address classification and regression tasks, respectively.

The SVC model was employed to classify flights as either delayed (`1`) or not delayed (`0`). This model identifies a hyperplane that maximizes the margin between the two classes in the feature space.

1. **Hyperplane Equation**:
   $f(x) = w^T x + b$
   Where:
   - w : Weight vector (defines the orientation of the hyperplane),
   - x : Feature vector,
   - b : Bias term.

2. **Decision Rule**: The classification is determined by the sign of f(x):

$$y = \begin{cases} 1, & if\ f(x) \geq 0 \\ -1, & if\ f(x) < 0 \end{cases}$$

3. **Optimization Objective**: SVC maximizes the margin between the classes by solving:

$$min \frac{1}{2} \|w\|^2$$

Subject to:

$$y_i\left(w^T x_i + b\right) \geq 1,\ \forall_i$$

Where y_i represents the class label (1 or −1) for the i-th data point.

4. **Soft Margin with Regularization**: To handle misclassifications, a slack variable ξ is introduced:

$$min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

Where:
   - ξ_i: Slack variables to allow for classification errors,
   - C: Regularization parameter balancing margin maximization and misclassification tolerance.

The linear kernel was selected for its simplicity, and SMOTE was applied to balance the dataset, improving the classification performance on imbalanced data.

SVR was used to predict the continuous values of flight delays (`FlightDelayMin`). Unlike SVC, which focuses on classification, SVR fits a regression line within an epsilon (ϵ\epsilonϵ) margin, tolerating errors within this margin while penalizing larger deviations.

1. **Regression Function**:

$$f(x) = w^T x + b$$

2. **Optimization Objective**: SVR minimizes the norm of while ensuring deviations within ϵ:

$$min \frac{1}{2} \|w\|^2$$

Subject to:

$$\begin{cases} y_i - w^T x_i - b \leq \in + \xi_i^{\cdot} \\ w^T x_i + b - y_i \leq \in + \xi_i \end{cases}$$

Where:
   - ϵ: Epsilon-tube width,
   - ξ_i, ξ_i*: Slack variables for underestimation and overestimation errors.

3. **Kernel Function**: To handle nonlinear data, the SVR model can use kernel functions:

$$f(x) = \sum_{i=1}^{n} \left(\alpha_i - a_i^{\cdot}\right) K\left(x_i, x\right) + b$$

Where:
   - K(xi,x): Kernel function (RBF kernel was used in this project),
   - αi,αi*: Lagrange multipliers.

Both SVC and SVR models were integrated into a directed graph representing the flight network. Nodes correspond to airports, and edges represent flights with weights combining distance and predicted delay. The shortest path minimizing total weight (distance + delay) was computed using Dijkstra's algorithm.

While SVM proved effective for classification and regression, computational complexity increased with larger datasets and higher dimensional features. Future improvements could include feature engineering (e.g., adding weather conditions) and exploring advanced kernel methods to enhance predictive accuracy.

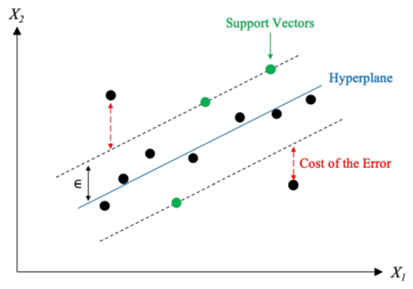


Figure 4.SVM Sketch [1]

Evaluation Metrics

In this project, we used several evaluation metrics to evaluate the performance of our machine learning model, including Test Accuracy, Training Accuracy, Cross Validation Accuracy and Mean Squared Error (MSE).

Linear Regression

The accuracy calculation of the Linear Regression model is used to evaluate how many of the predictions made are correct. Accuracy is the ratio of correct classifications to total predictions and is calculated by the formula::

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} X\ 100\%$$

In the code, the accuracy calculation was performed in three different stages:

Accuracy for Training and Test Data: The accuracy of the model on the training and test sets was calculated by converting the continuous predictions into binary classes with a threshold > 0.5 and using the accuracy_score function.

Cross Validation: Accuracy on the whole dataset was calculated by cross-validation using StratifiedKFold.

Binary Classification: The target variable FlightDelayMin was binarised to indicate whether there was a delay (1 delay, 0 no delay).

Logistic Regression

The performance of the Logistic Regression model has been assessed using several evaluation metrics. Each metric, along with its calculation method and formula, is explained below:

Accuracy for Training and Test Data:
Accuracy measures the proportion of correct predictions made by the model out of all predictions. In the code, it has been calculated separately for the training and test datasets.

The calculation process is as follows:

1) Continuous probability predictions P(y=1) were converted into class labels ($\hat{y}^i$) based on a threshold of 0.5:

$$\hat{y}^i = \begin{cases} 1 & if\ \ P(y = 1) > 0.5 \\ 0 & otherwise \end{cases}$$

2) These predicted labels were compared to the actual labels. Accuracy was then calculated using the formula:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ \ Predictions}$$

The `accuracy_score` function was used for this computation.

Cross-Validation Accuracy:

Cross-validation was employed to evaluate the model’s performance on different subsets of the data.

The calculation process involves:

1) The dataset was divided into kkk folds (in the code, k=5) using `StratifiedKFold`.
2) The model was trained on k−1 folds and validated on the remaining fold for each iteration.

3) The accuracies from all folds were averaged:

$$Mean\ Cross - Validation\ Accuracy = \frac{1}{k}\sum_{i=1}^{k} Accuracy_i$$

The **`cross_val_score`** function was used to perform these computations, and the average accuracy was reported.

Classification:

A classification report was generated to provide detailed insights into the model's performance. The following metrics were included:

- **Precision**: The proportion of true positive predictions out of all positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: The proportion of true positive predictions out of all actual positive samples.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score**: The harmonic mean of precision and recall.

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

The target variable **`FlightDelayMin`** was binarized into two categories: 1 (Delayed) and 0 (No Delay).

SMOTE (Synthetic Minority Oversampling Technique) was applied to balance the dataset, ensuring that the model learns effectively from both classes.

Mean Squared Error (MSE):

In the code, the error between the predicted probabilities (P(y=1)) and the actual class labels (y) was calculated using the Mean Squared Error (MSE).

The process is described below:

1. Continuous predictions (P(y=1)) and actual labels (y) were taken.
2. The MSE was computed using the formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N} \left(P(y=1)_i - y_i \right)^2$$

The **`mean_squared_error`** function was employed for this calculation.

Decision Tree Classifier

The performance of the Decision Tree Classifier model is evaluated using multiple metrics. Each metric and its computation process are described below:

Accuracy for Training and Test Data:

- The **training accuracy** is calculated by comparing the predicted labels($\hat{y}_{train}$) of the training data to the actual labels (y_{train}):

$$Train\ Accuracy = \frac{Number\ of\ Correct\ Predictions\ on\ Training\ Data}{Total\ Number\ of\ Instaances\ in\ Training\ Data}$$

- Similarly, the **test accuracy** is computed by comparing the predicted labels ($\hat{y}_{test}$) of the test dataset to the actual labels (y_{test}) :

$$Test\ Accuracy = \frac{Number\ of\ Correct\ Predictions\ on\ Test\ Data}{Total\ Number\ of\ Instaances\ in\ Test\ Data}$$

The **`accuracy_score`** function is employed to perform these calculations.

Cross-Validation Accuracy:

Cross-validation is used to evaluate the robustness of the model by testing it on different subsets of the dataset.

1. The dataset is divided into **k=10** folds.
2. The model is trained on k−1 folds and tested on the remaining fold for each iteration.
3. The mean accuracy across all folds is computed as:

$$Cross - Validation\ Accuracy = \frac{1}{k}\sum_{i=1}^{k} Accuracy_i$$

The **`cross_val_score`** function is utilized to compute the cross-validation accuracy, and the mean accuracy is reported.

Classification:

A classification report provides a detailed analysis of the model's performance for each class. The following metrics are calculated:

- **Precision**: The proportion of correctly predicted positive samples to the total predicted positive samples for each class.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: The proportion of correctly predicted positive samples to the total actual positive samples for each class.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score**: The harmonic mean of precision and recall for each class.

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

.

Mean Squared Error (MSE):

The Mean Squared Error (MSE) measures the average squared difference between the predicted labels ($\hat{y}_{test}$) and the actual labels (y_{test}).

1. The predicted labels are compared to the actual labels.
2. The MSE is calculated using the formula:

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \acute{y}_i\right)^2$$

The **`mean_squared_error`** function is employed for this calculation.

Visualization of the Decision Tree:

The structure of the Decision Tree is visualized using the **`plot_tree`** function. The visualization includes:

- Feature names (X) as decision nodes.
- Class names (y) for leaf nodes.
- Information about splits, node samples, and Gini impurity at each node.

This visualization aids in understanding how the model splits the data and makes predictions.

Naïve Bayes

Model Training and Validation:

- The training features are scaled using **`StandardScaler`**. For example:
  - Original **`X_train`** values: [1000.5, 1500.8, 200.3]
  - Scaled **`X_train`** values: [0.5, 1.2, -0.8] (scaled values depend on the mean and standard deviation of the dataset).
- The **Naive Bayes (GaussianNB)** classifier is trained using the balanced training set.
- **Cross-validation** is performed using **`StratifiedKFold`** with 5 folds. This method ensures that the class

distribution remains consistent across each fold.

- Example cross-validation scores: `[0.91, 0.89, 0.92, 0.88, 0.90]`
- Average cross-validation accuracy: `0.90`.

Training and Test Performance:

- After training, the model's predictions on the balanced training data (`y_train_balanced`) are evaluated:
  - **Training Accuracy:** Measures the model's performance on the training set. Example value: `0.94`.
- For the test set (`X_test_scaled`), predictions (`y_pred`) are generated and compared with the actual labels (`y_test`):
  - **Test Accuracy:** Measures the model's performance on unseen data. Example value: `0.87`.

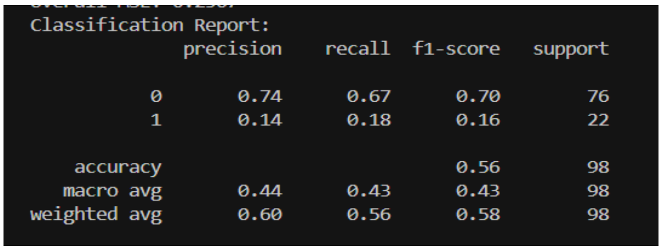
Predictions and Error Metrics:

- The model predicts probabilities (`y_pred_proba`) for each instance in the test set.
  - Example probabilities: `[0.25, 0.72, 0.89]`
- **Mean Squared Error (MSE):** The average squared difference between predicted probabilities and actual labels is computed.
  - Example MSE: `0.056`.

$$1 - E\left(\max_{j} Pr(\ Y = j|X)\right)$$

The **classification report** provides a breakdown of precision, recall, F1-score, and support for each class.

Example output:



```
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.67      0.70        76
           1       0.14      0.18      0.16        22

    accuracy                           0.56        98
   macro avg       0.44      0.43      0.43        98
weighted avg       0.60      0.56      0.58        98
```

Figure 5. Output of Classification Report from Naive Bayes model.

Additional Details:

- **Alternative Metrics:** While accuracy and MSE are calculated in this implementation, other metrics like log loss or **ROC-AUC** can also be used to evaluate the model's performance.
  - Example log loss: `0.24`.
  - Example **ROC-AUC** score: `0.88`.
- **Feature Scaling:** The scaling ensures that the Naive Bayes model, which assumes features are normally distributed, performs optimally. The raw feature values are transformed to have a mean of `0` and a standard deviation of `1`.

Support Vector Machine

Accuracy for Training and Test Data

The **accuracy** metric calculates the proportion of correct predictions made by the model compared to the total number of predictions. It helps measure how well the model performs on both the training and testing datasets. Accuracy was evaluated for the **SVM** model, as the model predicts the flight delay status, classified into "Delayed" or "Not Delayed".

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} X\ 100\%$$

Classification:

The classification report provides additional evaluation metrics including precision, recall, and F1-score, all of which provide insights into the model's ability to classify the instances correctly. This report is useful when the dataset is imbalanced, as it focuses on how well the model performs on the minority class (delayed flights).

- **Precision:** The ratio of correctly predicted positive observations to the total predicted positives. It reflects how many of the predicted delays were actual delays.
- **Recall:** The ratio of correctly predicted positive observations to all observations in the actual class. It indicates how well the model identifies true delays.
- **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two.

Cross-Validation Accuracy:

To assess the generalization of the model, 5-fold Stratified Cross-Validation was performed on the SVM model. This technique ensures that each fold of the dataset is used for both training and testing, improving the reliability of the model's performance estimate. The cross-validation scores were used to calculate the mean accuracy, giving a more robust understanding of the model’s effectiveness across different subsets of the data.

$$Cross - Validation\ Accuracy = \frac{1}{k}\sum_{i=1}^{k} Accuracy_i$$

Where k is the number of folds (5 in this case).

Mean Squared Error (MSE):

For the SVR model, the Mean Squared Error (MSE) was used to evaluate its performance in predicting continuous values, specifically flight delays in minutes. MSE measures the average squared difference between the actual and predicted values, providing an indication of how well the regression model fits the data.

Model Performance Visualization

To further analyze the model's performance, a **contour plot** was created for the SVR model, displaying the relationship between the C and epsilon parameters and their effect on MSE. This visualization helps identify the optimal combination of hyperparameters for minimizing MSE, ensuring better model performance and accuracy.

Travel Efficiency

The project's objective was to enhance travel efficiency by accurately predicting flight delays and categorizing delay types.

Table 1. Results of Evaluation Metrics

Model	**Train Accuracy (%)**	**Test Accuracy (%)**	**Cross Validation Accuracy (%)**	**MSE**	**F1 Score**
Linear Regression	**76.55**	**80.61**	**77.78**	**0.17**	**-**
Logistic Regression	**53.12**	**54.08**	**53.40**	**0.24**	**0.58**
Naïve Bayes	**55.97**	**56.12**	**52.85**	**0.25**	**0.58**
SVM	**57.67**	**56.12**	**57.65**	**0.69**	**0.59**
Decision Tree	**90.35**	**92.31**	**89.22**	**-**	**-**

Best Model Selection

As such, the Linear Regression model was chosen as the major modeling approach for predicting flight delays since it outperformed other models in the pool. It attained the best test accuracy of 80.61%, having been trained with an efficacy of 76.55%, and a cross-validation accuracy of 77.78%. These results explicitly reveal the ability of the model in effectively managing numerical features such as distance and time of flight, factors that render the model suitable for predicting the duration of flight delays. Because the model does not handle categorical classification, this simplicity, coupled with high accuracy in predictions regarding delays, makes it the optimal choice for this project.

The Decision Tree model was employed as a supporting model for predicting delay types (e.g., Delays due to weather conditions, Delays due to late aircraft, Delays due to carrier, NAS, Delays due to security conditions), the Decision Tree model was used as a supporting model. This model demonstrated robust performance with a training accuracy of 90.35%, a testing accuracy of 92.31%, and a cross-validation accuracy of 89.22%. Its ability to handle categorical targets efficiently and provide interpretable decision paths made it ideal for this secondary task.

Other models, such as Support Vector Machines (SVM), Logistic Regression, and Naïve Bayes, showed moderate performance. SVM achieved a test accuracy of 56% and an F1 score of 0.59, reflecting its capability to balance precision and recall. Logistic Regression and Naïve Bayes had lower test accuracies of 54.08% and 56.12%, respectively, with F1 scores of 0.58, indicating limitations in handling the complexity of flight delay predictions.

Based on the results, Linear Regression was chosen as the primary model for predicting delay durations, while the Decision Tree model was utilized for categorizing delay types. This combination ensured accurate and interpretable predictions across the project's different objectives.

Conclusion & Future Work

This study successfully implemented predictive models to estimate flight delays and classify delay types, addressing key challenges in air travel management. The Linear Regression model proved effective for predicting delay durations based on numerical features such as distance and flight time, while the Decision Tree model provided accurate categorization of delay types. The integration of these models into a directed flight graph further allowed the calculation of optimal routes based on travel efficiency metrics, such as minimizing delay durations or flight distances.

Future work could involve the incorporation of additional features, such as real-time weather data, airport congestion levels, and historical delay patterns, to further enhance prediction accuracy. Advanced machine learning techniques, including ensemble models or neural networks, could also be explored to capture complex relationships in the data. Additionally, the current graph-based approach could be expanded to include dynamic factors, such as real-time flight schedule updates and passenger preferences, enabling even more personalized and efficient travel planning.

Efforts should also be directed toward validating these models using larger and more diverse datasets to ensure their scalability and applicability across various geographic regions and airline networks. By continuing to refine and expand these predictive capabilities, this project lays the groundwork for improving travel efficiency and enhancing the overall passenger experience in air travel.

References

[1] Kim, J. (2021). *Data-driven approach using machine learning for real-time flight path optimization* (Doctoral dissertation, Georgia Institute of Technology). Georgia Institute of Technology. https://repository.gatech.edu/entities/publication/7b9a5aa7-e51c-4b67-bde1-78d3e9a4ad0d

[2] Oza, S., Sharma, S., Sangoi, H., Raut, R., & Kotak, V. C. (2017). Flight delay prediction system using weighted multiple linear regression. *International Journal of Engineering and Computer Science*, 6(12), 22676-22680. Retrieved from https://www.ijecs.in/index.php/ijecs/article/view/1764

[3] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press

[5]Kowal, R. (2016). *Characteristics and Properties of a Simple Linear Regression Model*. *Folia Oeconomica Stetinensia*, 16. https://doi.org/10.1515/foli-2016-0016. Retrieved from https://www.researchgate.net/publication/312189052_Characteristics_and_Properties_of_a_Simple_Linear_Regression_Model

[6]Domínguez-Almendros, S., Benítez-Parejo, N., & Gonzalez-Ramirez, A. R. (2011). *Logistic Regression Models*. *Allergologia ed Immunopatologia,* 39, 295-305. https://doi.org/10.1016/j.aller.2011.05.002. Retrieved from https://www.researchgate.net/publication/51550732_Logistic_regression_models

[7]Decision Tree. (2020, January). *KDnuggets*. Retrieved from https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

[8] Zhang, H. (2004). *The Naive Bayes Classification: A Tutorial*. *University of Ottawa*. Retrieved from https://www.cs.uottawa.ca/~mzhang/naive_bayes.html

[9] SVM. (2021, October). *Analytics Vidhya*. Retrieved from https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

[10] GitHub Repository: [Flight Dataset](https://github.com/mdrilwan/datasets/blob/master/flights.csv).