This code is part of a data analysis and predictive modeling project, specifically focused on a dataset related to flights and their arrival times. Here's an explanation of each section of the code and how I would present the work to a potential client or employer, highlighting the value and results of the work.

**1. DATA LOADING AND PREPARATION**

The first section focuses on loading and preprocessing the data.

* **Removing unnecessary columns**: Unused columns are dropped from the dataset.
* **Handling missing values (NaN)**: There are 49,510 cancelled flights, with no ARR\_TIME. We proceed to delete these  flights as will not have delete time
* **Converting categorical variables**: The day of the week and the month are converted into numerical values to make them more suitable for machine learning models.

These transformations are crucial for ensuring that machine learning models work with clean and well-structured data.

**2. TRAIN TEST-SPLIT**

Next, the data is split into a training set (80%) and a testing set (20%) using train\_test\_split. This is a standard practice in machine learning to evaluate model performance on data that wasn't seen during training.

This split is essential to prevent overfitting and to ensure that the model generalizes well on unseen data.

**3. VARIABLES (Categorical and Numerical)**

Each variable in the DataFrame is classified based on its type:

* **Categorical variables** (ORIGIN\_CITY\_NAME,DEST\_CITY\_NAME ).
* **Numerical variables** (MONTH, DAY\_OF\_MONTH, DAY\_OF\_WEEK, CRS\_DEP\_TIME,CRS\_ARR\_TIME ).

A dictionary of categorical and numerical variables is created to make further processing and transformation easier.

This classification helps apply appropriate preprocessing techniques, such as encoding or normalization, based on the type of variable.

**4. ENCODING**

The code creates a mapping of the origin and destination cities to unique numerical values. This process, called **encoding**, transforms categorical variables into values that machine learning models can understand.

Additionally, a **logarithmic transformation** is applied to variables like "Month" and "Day of the Week" to improve the data distribution and avoid bias.

These transformations enhance the model’s ability to handle categorical variables and ensure better handling of numeric distributions.

**5. NORMALIZATION**

The next step is to **normalize** the numerical variables (such as delays, flight times) using StandardScaler and the categorical using OneHotEncoder. This is essential to ensure that the models, work efficiently without some variables dominating others due to their scale.

Normalization improves the convergence and accuracy of models, especially for algorithms like logistic regression or neural networks.

**6. MACHINE LEARNING MODELS**

The code implements three machine learning models:

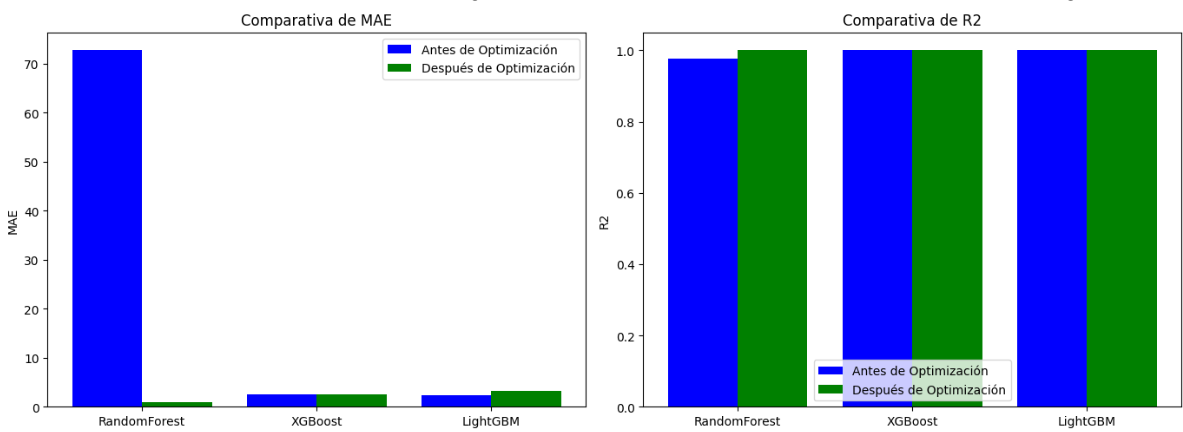
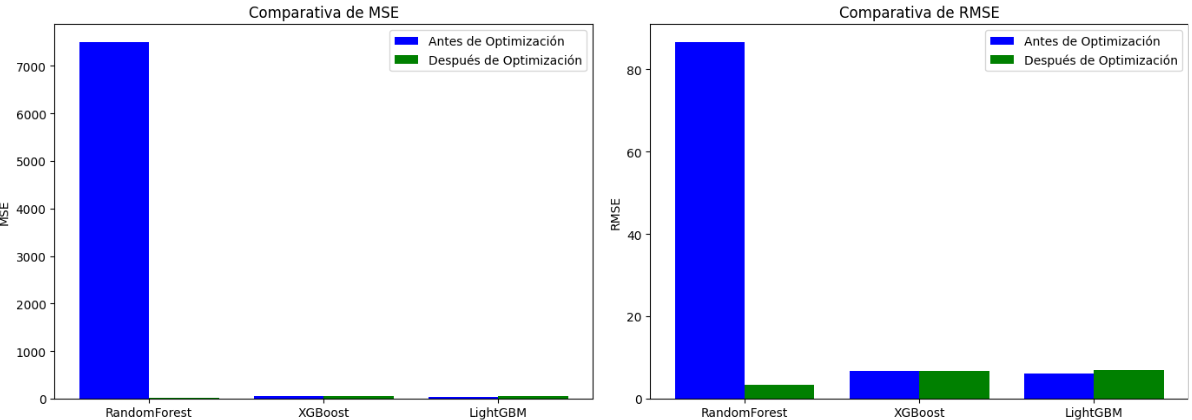
* **Random Forest Regressor**: A decision tree-based model used to predict flight arrival times (ARR\_TIME).
* **XGBoost Regressor**: A boosting algorithm that uses decision trees and is popular for handling large datasets and providing strong predictions.
* **LightGBM Regressor**: Similar to XGBoost but more efficient in terms of computational time and resources, especially for large datasets.

The models are trained using the training set (X\_train and y\_train) and evaluated using the test set. Various evaluation metrics are calculated, including **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, **Mean Absolute Error (MAE)**, and **R² coefficient**.

These three models are highly effective for predicting continuous values like flight arrival times. Comparing their results gives a clear picture of which algorithm performs best for the dataset.

**7. HYPERPARAMETER**

Using **RandomizedSearchCV**, a randomized search is performed to find the optimal hyperparameters for each model. This ensures the models are fine-tuned for maximum performance.



- Random Forest achieves significantly better performance on all metrics after optimization compared to the default values ​​shown in the first code. Is the bhe best model among the three, with the best MSE, RMSE, MAE, and a near-perfect R².

- XGBoost is still a solid choice, but it does not outperform Random Forest on any of the metrics.

- LightGBM, while efficient, has shown somewhat inferior performance compared to XGBoost and Random Forest after optimization.

Conclusion

- **Random Forest is the best model** for this dataset, as it has the best performance on all key metrics after optimization.

- XGBoost is a second competitive option, but if you are looking for the most accurate model, Random Forest is the optimal choice.

- LightGBM, despite its good ability to handle large volumes of data, does not offer the best fit in this case.

**8. RESULTS**

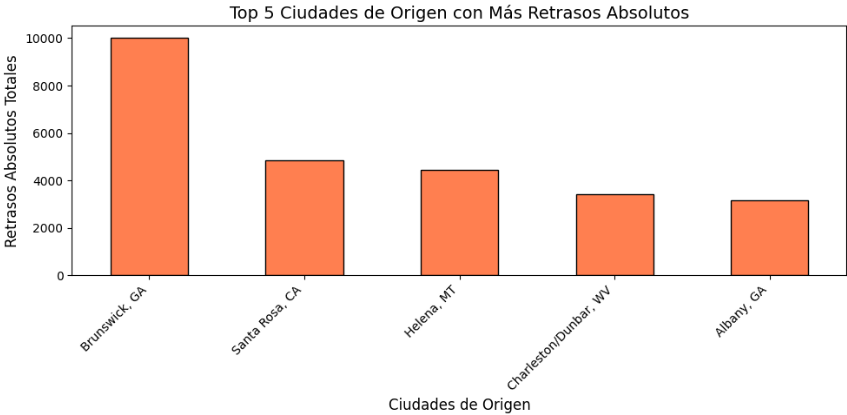
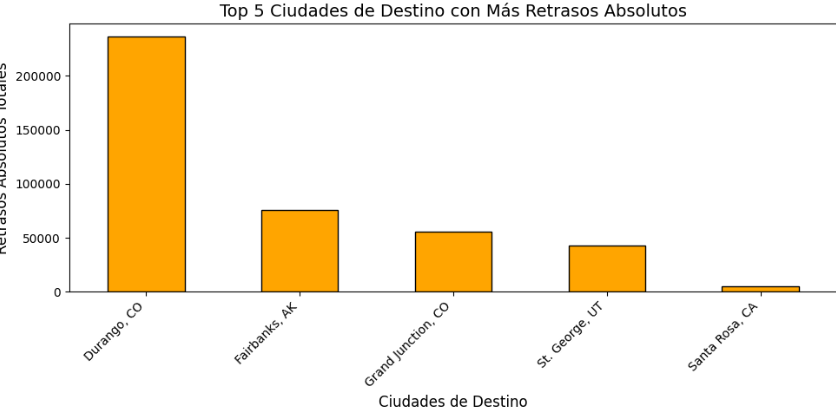
After calculating the ARR\_TIME again, we will proceed to do some analysis in order to see the delays across the country.

For that analysis we will create the following columns:

-flight duration

-absolute delay

-relative delay



- The month with the most delays is February

* The day of the week with the most delays is Saturday
* The flight with the longest delay is Phoenix, AZ - Durango, CO
* The flight with the shortest delay is Chicago, IL - New York, NY