PREDICTING FLIGHT DELAY

Flight delays have become so common that people almost expect them. But behind every late takeoff, there's more than just inconvenience — there's lost money, frustrated passengers, missed connections, and disrupted operations.

We explored the historical flight data with two key goals:

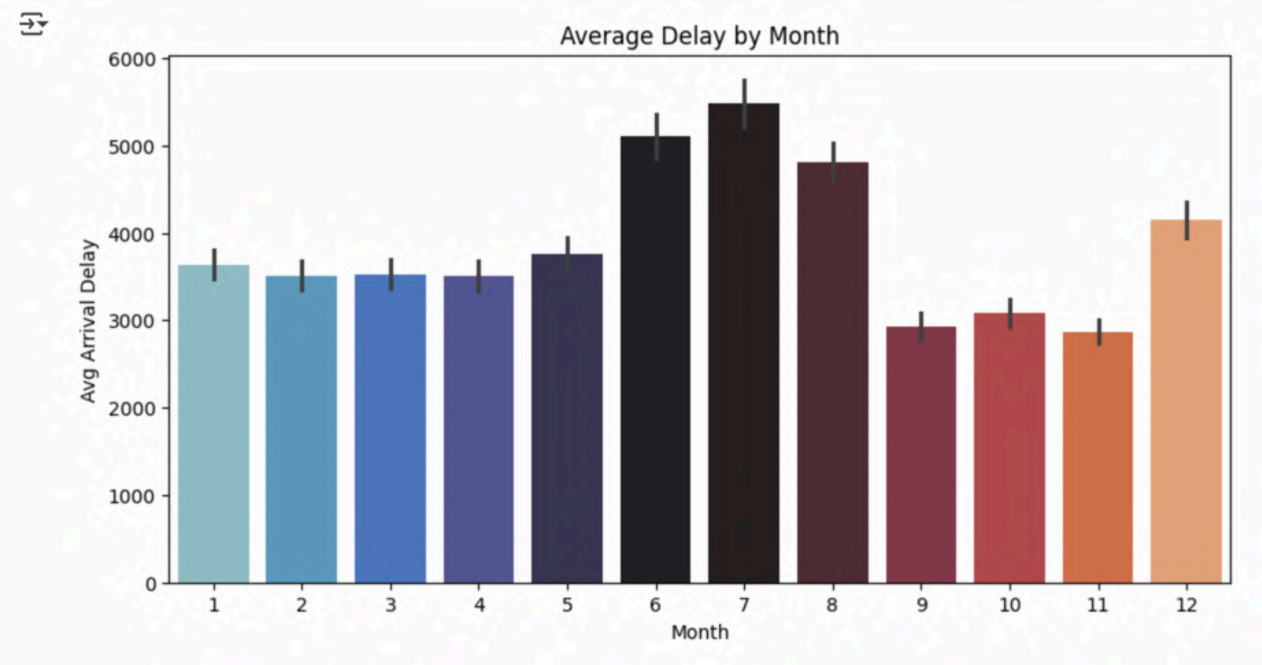
- Understand what really causes delays — not just weather or air traffic, but deeper patterns
- Build models that can flag risky flights early — both as Yes/No and by estimating the delay time
- We created a custom metric to focus more on fixable delays (like those caused by carriers or aircraft rotation)
- And we used explainable ML (SHAP) to show why a flight is predicted to be delayed

Prachi Sawant

23112073

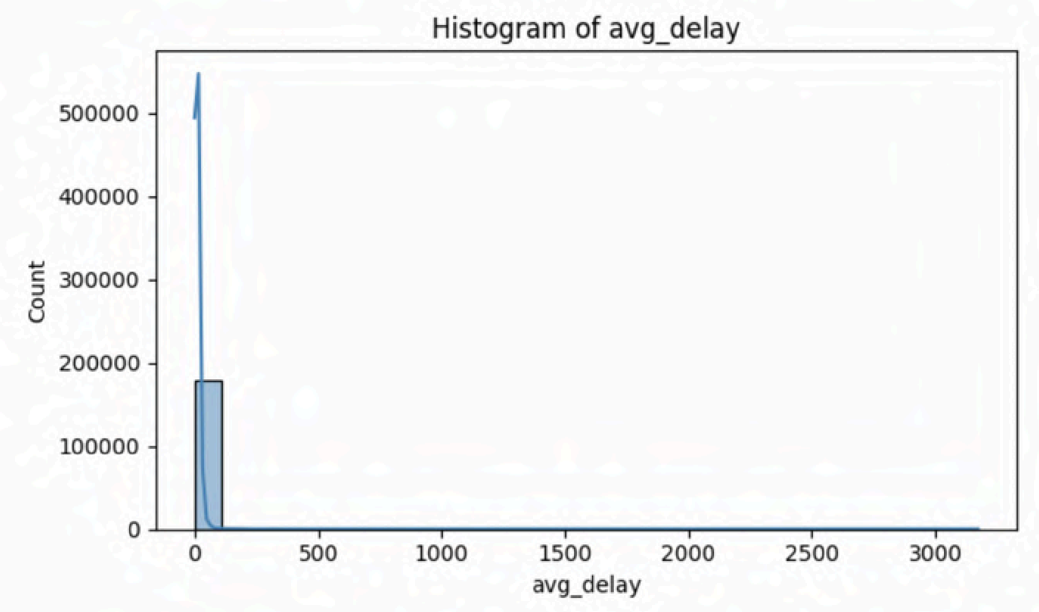
Exploratory Data Analysis

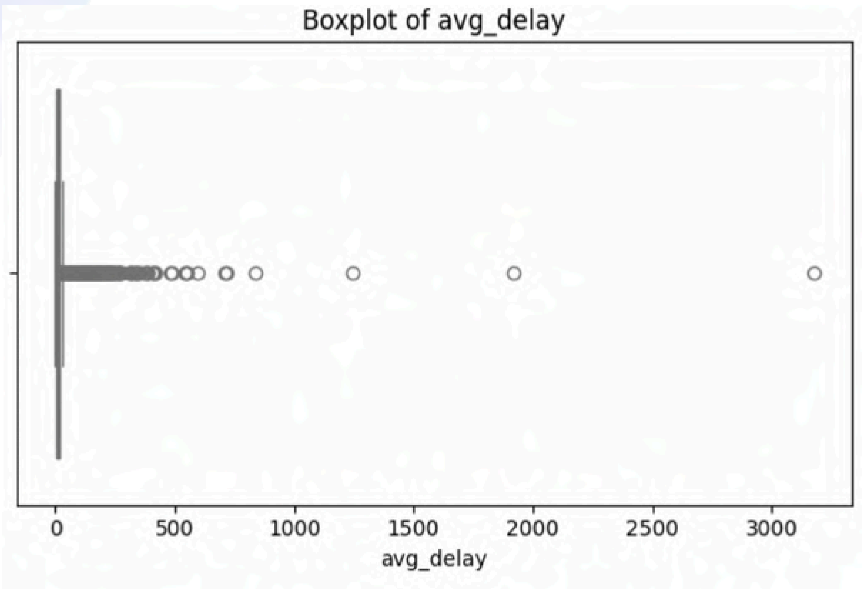
Distribution of Average delay per Month



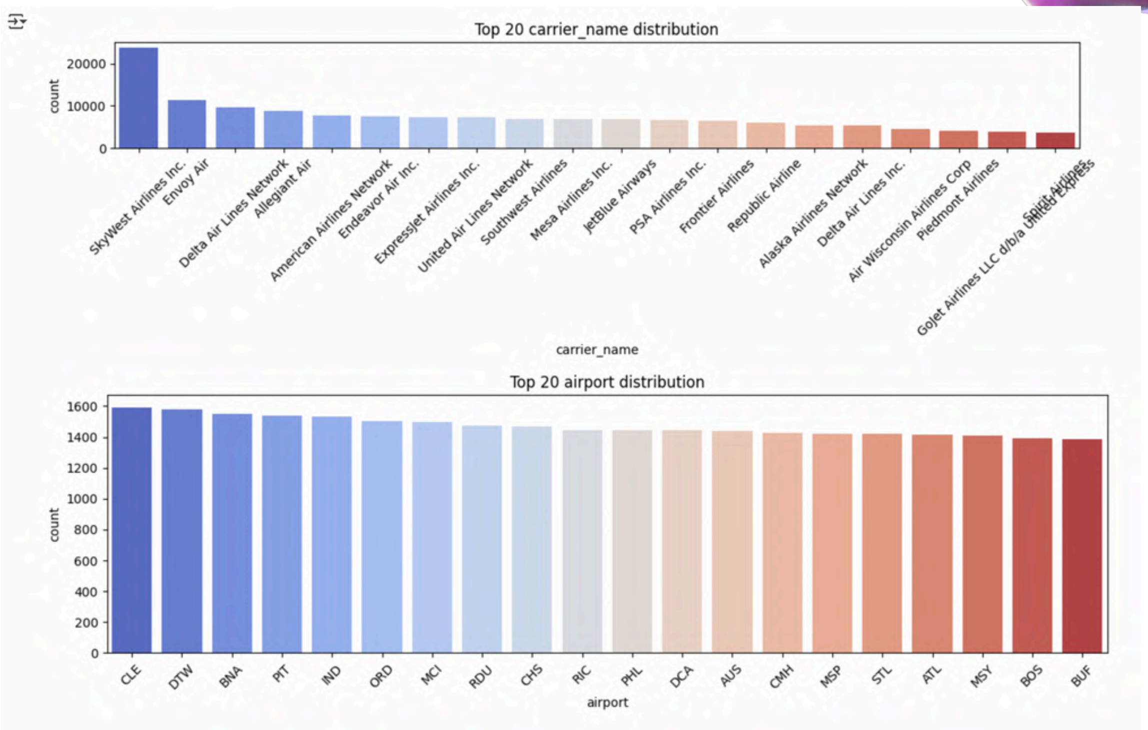
Month and Year

- July, August, and December have peak frequencies.
- these months align with holiday travel seasons in the US.



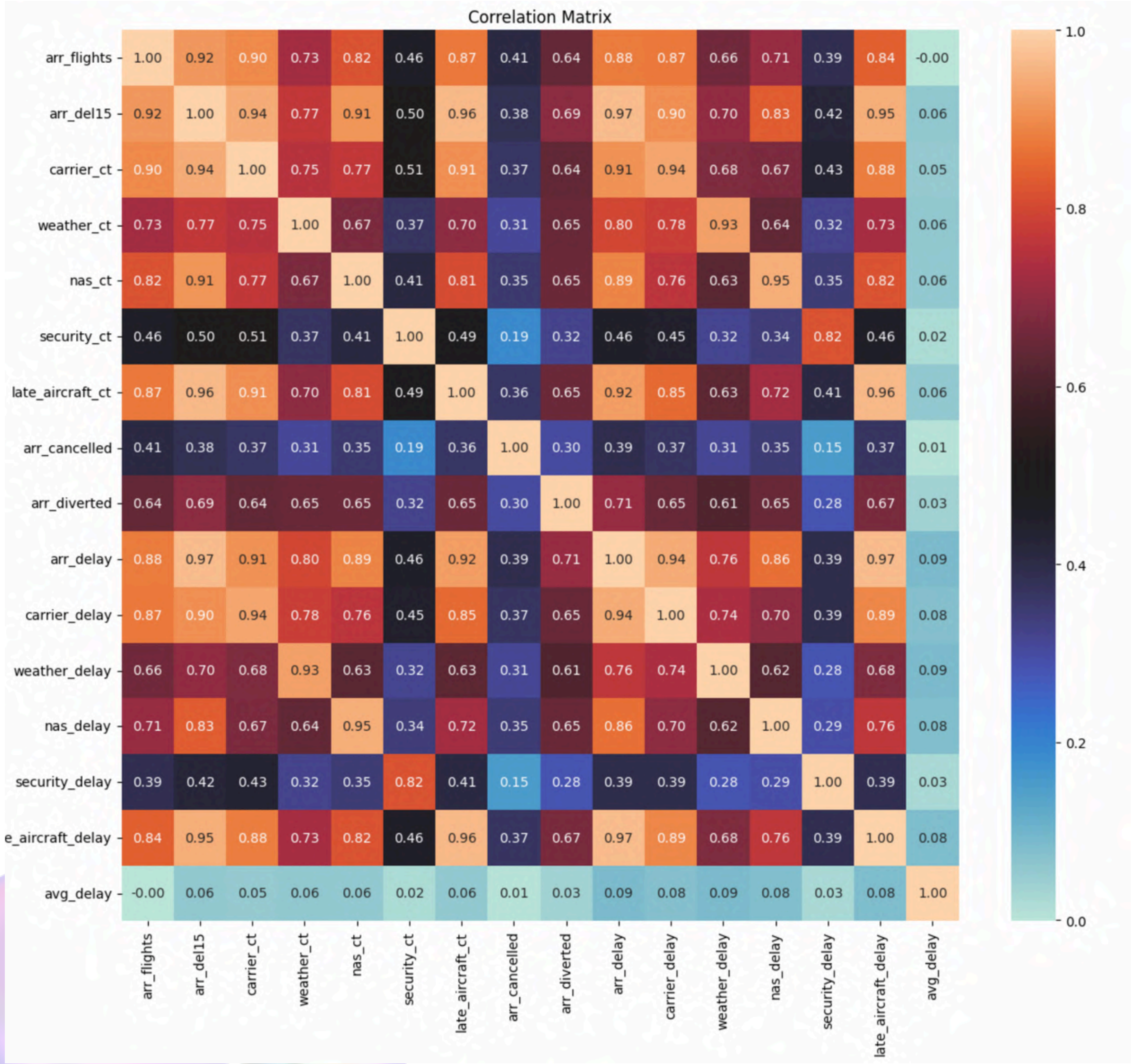


From the plots we can see that avg_delay has outliers beyond 75th percentile and is highly right skewed.



Airport and carrier distributions

Correlation Analysis



- Individual delay types such as carrier-related, weather-based, and others show a weak direct link with overall average delay (correlation values stay under 0.07).
- Interestingly, many of these delay types show strong inter-dependence among each other, indicating they often happen together rather than in isolation.
- Key Takeaway: Instead of analyzing delay factors in silos, we need multivariate models that can account for these interacting effects for better predictions.

- Among all categories, weather-related delays emerge as the most time-consuming on average.
- Delays caused by late-arriving aircraft also significantly affect overall schedules.
- Security issues, on the other hand, contribute minimally to overall delay time.
- Action Plan: To make meaningful improvements, efforts should be strategically targeted at weather disruptions and aircraft turnaround delays.

Data Preprocessing

1. Removing outliers

```
[ ]  # Removing outliers
     q1 = df["avg_delay"].quantile(0.25)
     q3 = df["avg_delay"].quantile(0.75)
     iqr = q3 - q1

     upper_bound = q3 + 1.5 * iqr
     df = df[df["avg_delay"] <= upper_bound].copy()
```

3. Encoded Columns

- airport_encoded (from airport)
- carrier_encoded (from carrier)
- season_encoded (winter, summer, fall)

```
le_carrier = LabelEncoder()
le_airport = LabelEncoder()
le_season = LabelEncoder()

df["carrier_encoded"] = le_carrier.fit_transform(df["carrier"])
df["airport_encoded"] = le_airport.fit_transform(df["airport"])
df['season_encoded'] = le_season.fit_transform(df['season'])
```

2. Feature Engineering

1. **is_delayed** is a binary column (1 if arravival flight is delayed by more than 15 minutes else 0)
   - this will help in classification task
2. **total_delay_ct** aggregates the number of delay incidents, giving a simple measure of overall disruption for a given flight or time window.
3. Splitting delays into controllable vs. external helps model real-world operational vs. uncontrollable risks. For example, weather is not under airline control, but carrier delays are.
   - controllable_delay = carrier_delay + late_aircraft_delay
   - external_delay = weather_delay + nas_delay + security_delay
4. **delay_rate** gives the proportion of flights delayed, a measure thatnis useful when comparing months with different traffic volumes.
5. **peak_season** Binary indicator for high-traffic travel months (e.g., summer, holidays) suggesting increased risk during these periods.
6. **monthly_traffic**

4. Splitting of data and scaling

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

X_train = pd.DataFrame(X_train).fillna(0)
X_test = pd.DataFrame(X_test).fillna(0)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Regression Model

Metrics obtained of regression models

```
Linear Regression performance on target:
MAE:  2.59
RMSE: 3.74
R²:   0.7014
```

```
XGBoost performance on target:
MAE:  0.60
RMSE: 0.93
R²:   0.9814
```

```
Random Forest performance on target:
MAE:  0.49
RMSE: 0.97
R²:   0.9801
```

```
LightGBM performance on target:
MAE:  0.67
RMSE: 1.05
R²:   0.9767
```

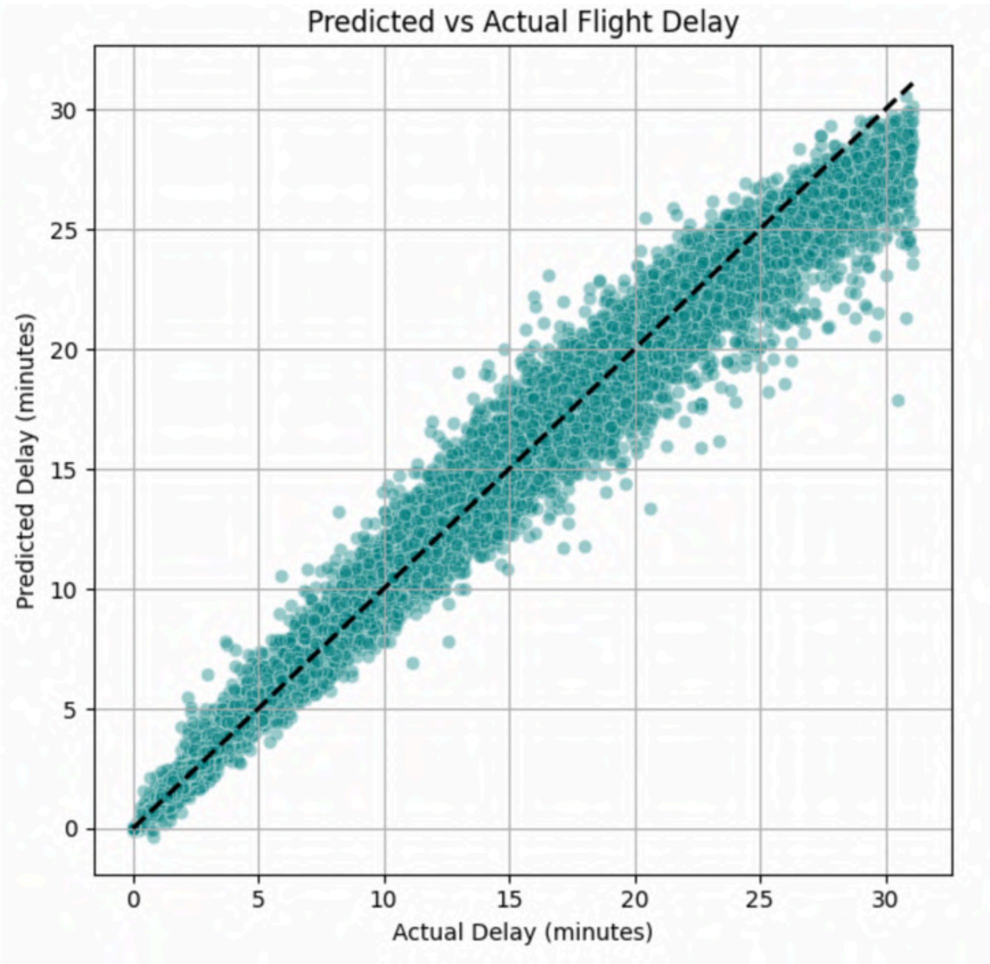
Checking for overfitting of our best model

```
Best Model: XGBoost
MAE : 0.596
RMSE: 0.933
R²  : 0.981
```

```python
y_train_pred = xgb.predict(X_train_scaled)
y_test_pred = xgb.predict(X_test_scaled)
print("\n XGBoost Regressor Overfitting Check")
print("-" * 50)
print("Train R²: ", round(r2_score(y_train, y_train_pred), 4))
print("Train RMSE:", round(np.sqrt(mean_squared_error(y_train, y_train_pred)), 4))
print("Train MAE: ", round(mean_absolute_error(y_train, y_train_pred), 4))
print('-' * 50)
print("Test  R²: ", round(r2_score(y_test, y_test_pred), 4))
print("Test  RMSE:", round(np.sqrt(mean_squared_error(y_test, y_test_pred)), 4))
print("Test  MAE: ", round(mean_absolute_error(y_test, y_test_pred), 4))
```

```
 XGBoost Regressor Overfitting Check
--------------------------------------------------
Train R²:  0.9849
Train RMSE: 0.8366
Train MAE:  0.5492
--------------------------------------------------
Test  R²:  0.9814
Test  RMSE: 0.933
Test  MAE:  0.5965
```

Since our test and test scores are very close, especially R2. It implies our XGBoost model is not overfitting and performs well on unseen data.

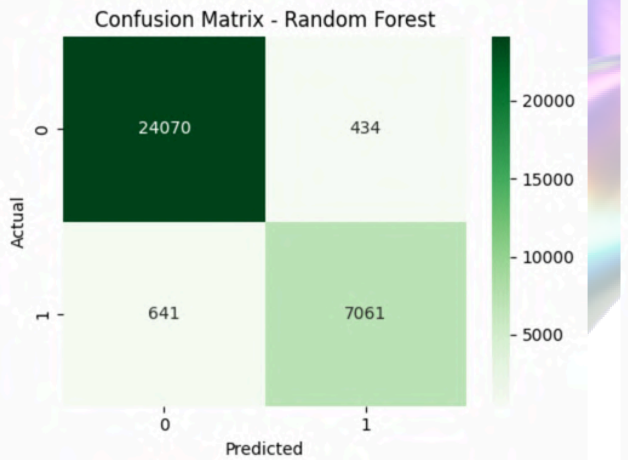


- The points in the plot closely follow the diagonal line, indicating good predictive performance.
- Slight spread around the line suggests some prediction error, but it's generally tight.
- The plot shows the model captures the overall trend well, especially for delays up to ~30 minutes.

Classification Model

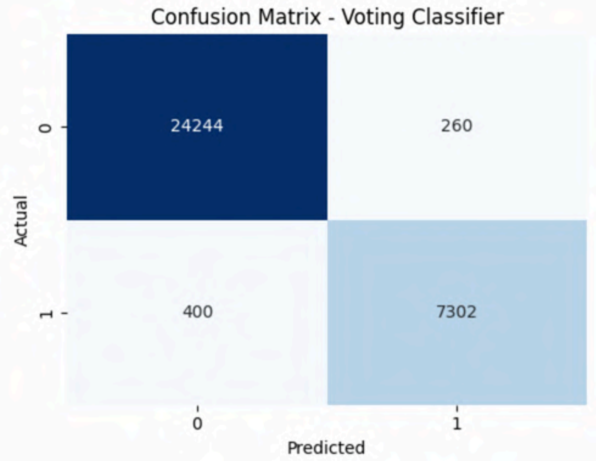
Classification Report - Random Forest:

	precision	recall	f1-score	support
0	0.97	0.98	0.98	24504
1	0.94	0.92	0.93	7702
accuracy			0.97	32206
macro avg	0.96	0.95	0.95	32206
weighted avg	0.97	0.97	0.97	32206



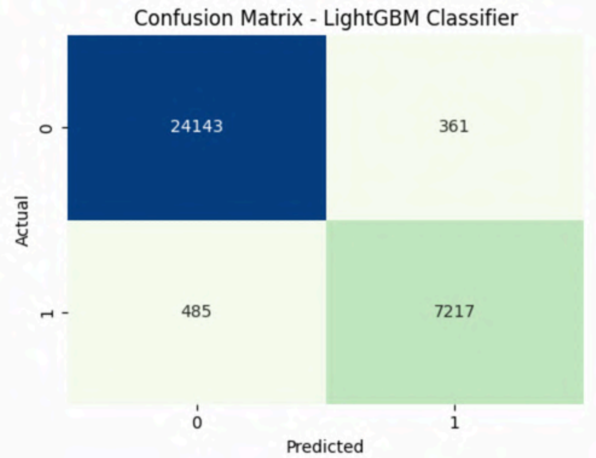
Classification Report - Voting Classifier:

	precision	recall	f1-score	support
0	0.984	0.989	0.987	24504
1	0.966	0.948	0.957	7702
accuracy			0.980	32206
macro avg	0.975	0.969	0.972	32206
weighted avg	0.979	0.980	0.979	32206



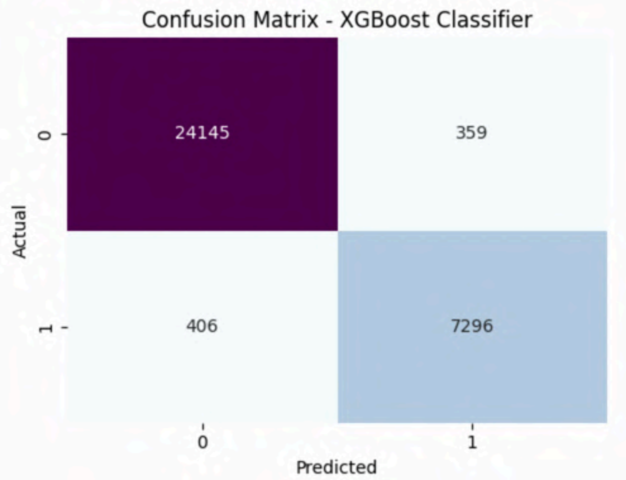
Classification Report - LightGBM Classifier:

	precision	recall	f1-score	support
0	0.980	0.985	0.983	24504
1	0.952	0.937	0.945	7702
accuracy			0.974	32206
macro avg	0.966	0.961	0.964	32206
weighted avg	0.974	0.974	0.974	32206



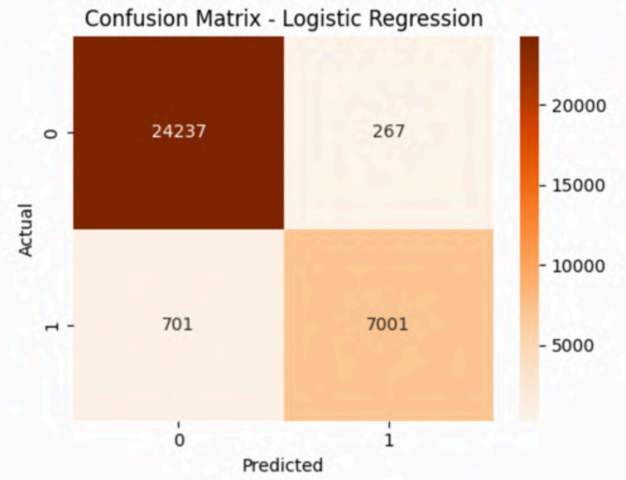
Classification Report - XGBoost Classifier:

	precision	recall	f1-score	support
0	0.983	0.985	0.984	24504
1	0.953	0.947	0.950	7702
accuracy			0.976	32206
macro avg	0.968	0.966	0.967	32206
weighted avg	0.976	0.976	0.976	32206



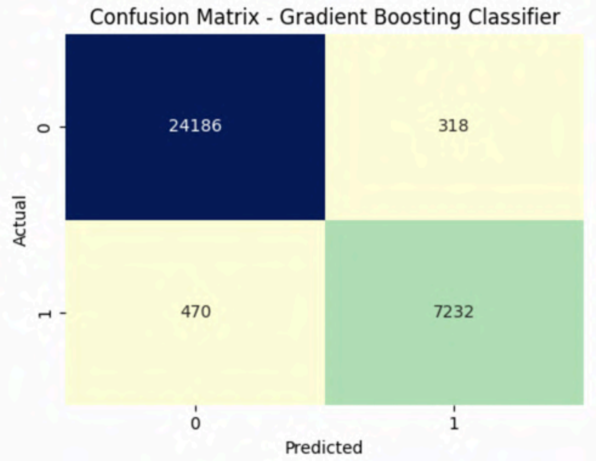
Classification Report - Logistic Regression:

	precision	recall	f1-score	support
0	0.972	0.989	0.980	24504
1	0.963	0.909	0.935	7702
accuracy			0.970	32206
macro avg	0.968	0.949	0.958	32206
weighted avg	0.970	0.970	0.970	32206



Classification Report - Gradient Boosting Classifier:

	precision	recall	f1-score	support
0	0.981	0.987	0.984	24504
1	0.958	0.939	0.948	7702
accuracy			0.976	32206
macro avg	0.969	0.963	0.966	32206
weighted avg	0.975	0.976	0.975	32206




```
Best Model: Voting Classifier
Accuracy: 0.98
F1 Score: 0.957
ROC AUC: 0.998
```

```
[ ] delay_output['Predicted Delay Status'].value_counts()
```

Predicted Delay Status	count
No	24644
Yes	7562

dtype: int64

```
[ ] delay_output['Actual Delay Status'].value_counts()
```

Actual Delay Status	count
No	24505
Yes	7701

dtype: int64

Insights

1. Very Close Distribution:
   - The predicted class distribution is extremely close to the actual one.
   - The classifier predicted 7,562 delayed flights, and 7,701 were truly delayed.
   - it predicted 24,644 as on-time, while 24,505 were actually on-time.
   - this shows the model isn't biased toward one class and is well-balanced.
2. Low Misclassification Rate:
   - Total samples = 32,206, aproxx ~150 predictions differ between actual and predicted counts.
   - Indicates low false positives/negatives.

OAI and SHAP Explanation

Operational Adjustability Index is a custom regression model that focuses on predicting delays caused by controllable factors. While traditional models predicted overall delay (avg_delay), OAI emphasizes delays that an airline can actually act upon and reduce.

```
[ ] # assigning higher weights to controllable delays
    def compute_oai(row):
        controllable = row['carrier_delay'] + row['late_aircraft_delay']
        uncontrollable = row['weather_delay'] + row['nas_delay'] + row['security_delay']

        return 0.7 * controllable + 0.3 * uncontrollable
```

We created a custom delay score (OAI) that assigns:

70% weight to controllable delays (carrier, late aircraft)

30% weight to uncontrollable delays (weather, NAS, security)

```
y_pred_oai = model_oai.predict(X_test_oai)

# Evaluation metrics
mae_oai = mean_absolute_error(y_test_oai, y_pred_oai)
rmse_oai = np.sqrt(mean_squared_error(y_test_oai, y_pred_oai))
r2_oai = r2_score(y_test_oai, y_pred_oai)

# Print evaluation
print("OAI Prediction Performance:")
print(f"MAE   : {mae_oai:.3f}")
print(f"RMSE  : {rmse_oai:.3f}")
print(f"R²    : {r2_oai:.4f}")

OAI Prediction Performance:
MAE   : 94.535
RMSE  : 1087.115
R²    : 0.9776
```

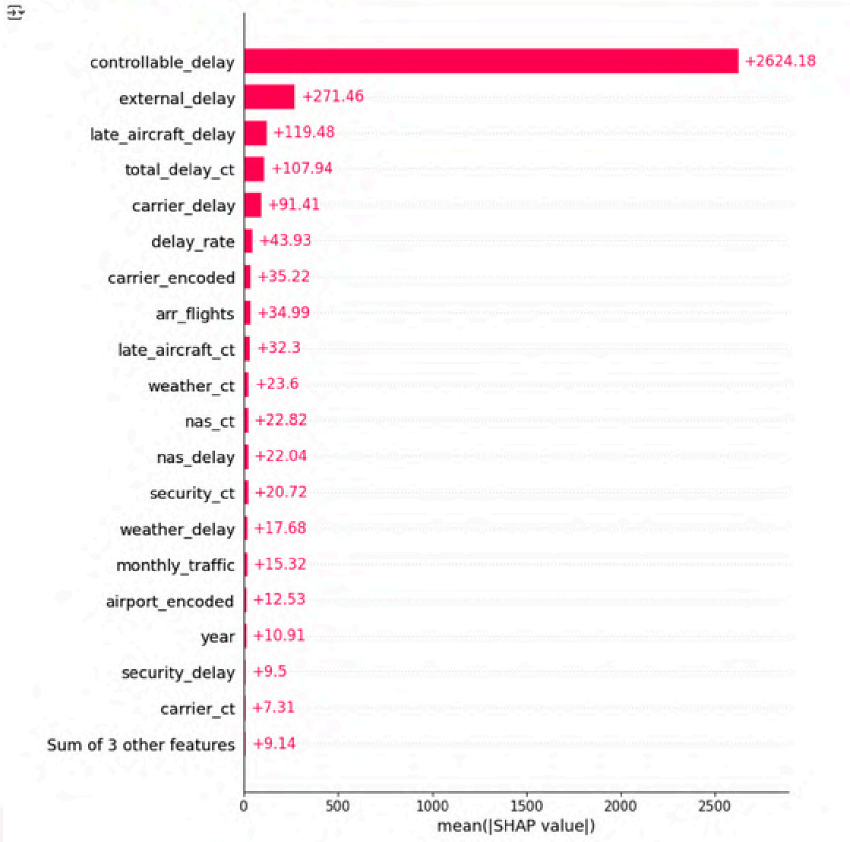
```
Training Performance (OAI):
R²    : 0.9996
RMSE  : 148.0692
MAE   : 46.3042
------------------------------
Test Performance (OAI):
R²    : 0.9776
RMSE  : 1087.1148
MAE   : 94.5349
```

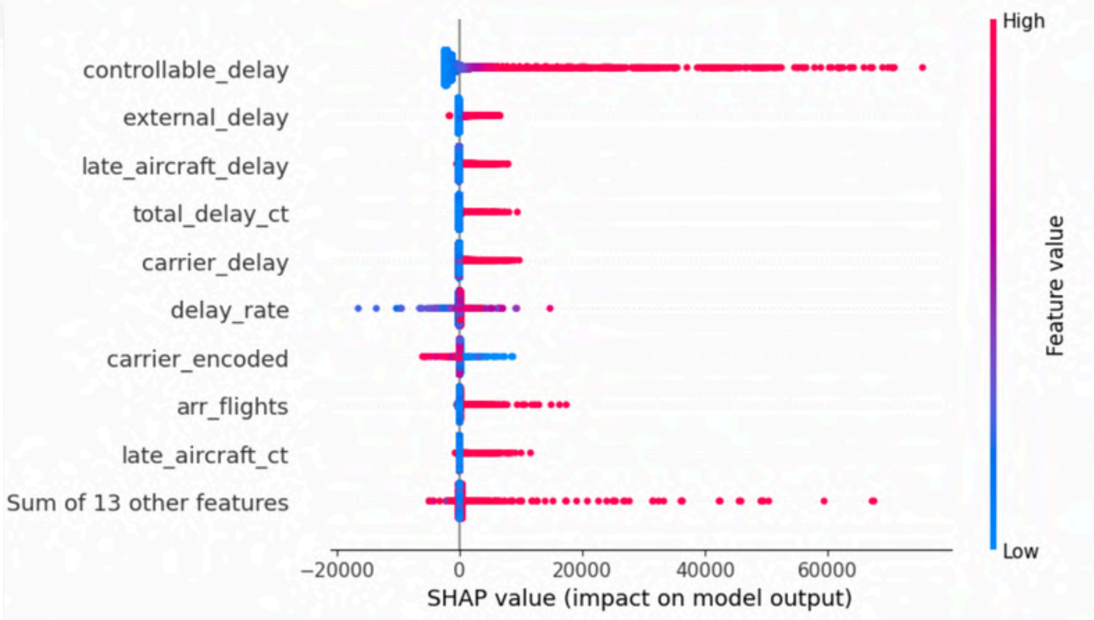
Checking for overfitting of our OAI model

the gap in RMSE—suggests:

- The model may be memorizing patterns in training data too closely.
- there might be mild overfitting
- we will remove this overfitting through shap plots and identify and remove noisy/low-impact features.

```
shap.plots.bar(shap_values, max_display=20)
```





- After breaking down the model's logic using SHAP, we found that delays caused by aircraft arrival issues and carrier handling had the strongest influence on predictions

Airline benefits:

- Plan Flights Smarter: Flag flights that usually end up delayed and tweak buffer times — especially for early-morning slots where delays often snowball.
- On-Ground Fixes at Trouble Spots: Some airports cause more problems than others, smarter gate assignments and live coordination can really reduce ground time and chaos.
- Upgrade the Way Performance is Measured: move beyond tracking raw delay time. Instead, judge teams by how they manage the parts of delay they can actually control — using our OAI-based insights.
- Rethink Internal Workflows: For routes consistently hit by late aircraft or airline-side slowdowns, fine-tune crew shifts, gate usage, and turnaround workflows to tighten the loop.

THANK YOU!

Prachi Sawant | 23112073 | Chemical Engineering