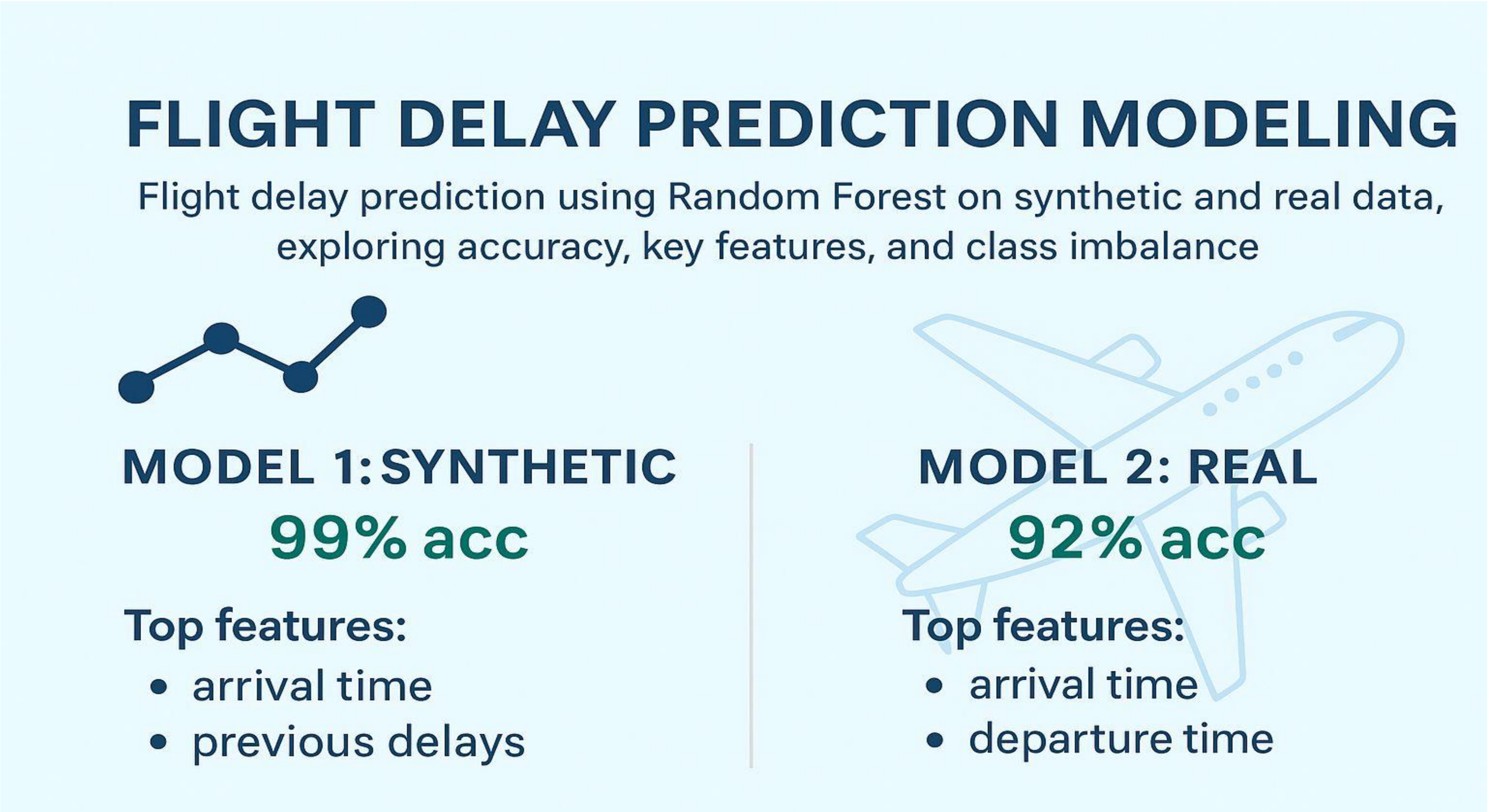


Gordon Andric 2025

After my family was stranded during a London transit meltdown, I wanted to engineer a predictive system that could reduce disruptions like this.

AI Model 1 – Synthetic Data : Creating the Artificial Data Set

```
[2]: np.random.seed(42)

num_samples = 50000  # Data size

df_synthetic = pd.DataFrame({
    "FL_DATE": pd.date_range(start="2024-01-01", periods=num_samples, freq="D"),
    "AIRLINE": np.random.choice(["Delta", "American", "United", "Southwest", "Alaska", "JetBlue", "Spirit", "Frontier"], num_samples),
    "ORIGIN": np.random.choice(["JFK", "LAX", "ORD", "ATL", "SEA", "DFW", "MIA", "SFO", "DEN", "PHX"], num_samples),
    "DEST": np.random.choice(["BOS", "LAS", "IAH", "MSP", "CLT", "DTW", "EWR", "FLL", "BWI", "SLC"], num_samples),
    "CRS_DEP_TIME": np.random.randint(500, 2359, num_samples),
    "NUM_PREVIOUS_FLIGHTS_LATE": np.random.randint(0, 10, num_samples),  # More variability
    "AVG_GATE_WAIT_TIME": np.random.uniform(0, 60, num_samples),
})

# Simulate delays based on broader conditions
df_synthetic["DELAYED"] = (
    (df_synthetic["NUM_PREVIOUS_FLIGHTS_LATE"] >= 5) |
    (df_synthetic["AVG_GATE_WAIT_TIME"] > 50)
).astype(int)

# Save for future use
df_synthetic.to_csv("expanded_flight_delays.csv", index=False)

print("New Dataset Size:", df_synthetic.shape)

New Dataset Size: (50000, 8)
```

Output Results

AI Model 1 – Synthetic Data : Data Set

expanded_flight_delays.csv ×

C: > my_github_repos > Flight-Delay-Prediction-Modeling > expanded_flight_delays.csv > data

```
FL_DATE,AIRLINE,ORIGIN,DEST,CRS_DEP_TIME,NUM_PREVIOUS_FLIGHTS_LATE,AVG_GATE_WAIT_TIME,DELAYED
2024-01-01,Spirit,PHX,CLT,917,9,7.478868696583549,1
2024-01-02,Southwest,PHX,IAH,1345,8,49.40366927292514,1
2024-01-03,Alaska,JFK,MSP,1981,3,26.901369518243758,0
2024-01-04,Spirit,ORD,EWR,2330,1,46.92140320678622,0
2024-01-05,United,ATL,IAH,2285,3,39.61700657449153,0
2024-01-06,Frontier,MIA,BOS,1352,5,48.41399442100398,1
2024-01-07,Alaska,DEN,BOS,1201,2,34.008907082564754,0
2024-01-08,Alaska,SEA,MSP,1963,0,29.368711297202715,0
2024-01-09,Spirit,JFK,BOS,1386,8,59.708911934701675,1
2024-01-10,American,LAX,DTW,1804,6,0.6857796628601309,1
2024-01-11,United,PHX,MSP,2018,1,32.36797136871213,0
2024-01-12,Spirit,LAX,CLT,1944,8,54.198175226692946,1
2024-01-13,United,DEN,DTW,1195,3,2.527770840189254,0
2024-01-14,United,PHX,LAS,538,5,47.04697676305268,1
2024-01-15,Frontier,MIA,BOS,2034,7,19.59865872678699,1
2024-01-16,Alaska,LAX,SLC,1217,3,33.84299776359828,0
2024-01-17,Southwest,JFK,LAS,824,4,54.001670272365274,1
2024-01-18,Frontier,SEA,DTW,716,6,41.1699310043518,1
2024-01-19,Frontier,JFK,SLC,2080,4,7.671910249935614,0
2024-01-20,United,DEN,BWI,1845,3,22.73021841835738,0
2024-01-21,JetBlue,ORD,FLL,1123,8,34.35691656051302,1
2024-01-22,Alaska,LAX,SLC,647,6,26.897561787442495,1
2024-01-23,American,DFW,BOS,1405,1,16.303029373108217,0
2024-01-24,Frontier,ORD,CLT,1115,0,36.76633961378634,0
```

50,000 rows

AI Model 1 – Synthetic Data : Balancing Out the Synthetic Data Set

```
[5]: smote_tomek = SMOTETomek(random_state=42)
     X_train_balanced, y_train_balanced = smote_tomek.fit_resample(X_train, y_train)

     print("Balanced class distribution:", np.bincount(y_train_balanced))
```

```
Balanced class distribution: [17031 17031]
```

Output Results

AI Model 1 – Synthetic Data : Fitting the RandomForestClassifier Model to the Data

```
[6]: model = RandomForestClassifier(n_estimators=200, class_weight="balanced_subsample", random_state=42)
     model.fit(X_train_balanced, y_train_balanced)
```

```
[6]: RandomForestClassifier
RandomForestClassifier(class_weight='balanced_subsample', n_estimators=200,
                       random_state=42)
```

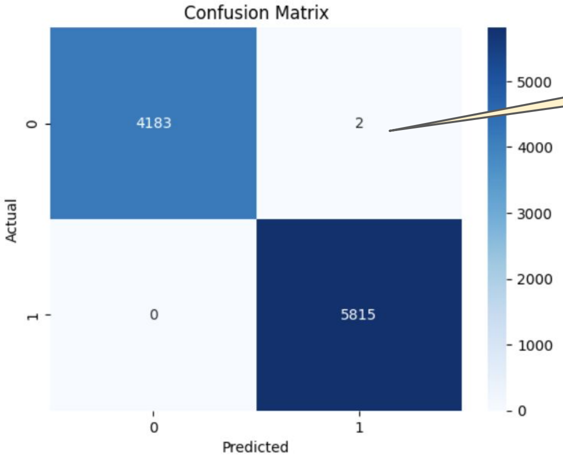
Output Results

AI Model 1 – Synthetic Data : Confusion Matrix and Accuracy

```
[7]: y_pred = model.predict(X_test)

     print("Accuracy:", accuracy_score(y_test, y_pred))
     print("Classification Report:\n", classification_report(y_test, y_pred))

     # Confusion Matrix
     cm = confusion_matrix(y_test, y_pred)
     sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
     plt.xlabel("Predicted")
     plt.ylabel("Actual")
     plt.title("Confusion Matrix")
     plt.show()
```

```
Accuracy: 0.9998
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00      4185
           1       1.00      1.00      1.00      5815

    accuracy                           1.00     10000
   macro avg       1.00      1.00      1.00     10000
weighted avg       1.00      1.00      1.00     10000
```

Output Results



Only two flights were predicted wrong

AI Model 1 – Synthetic Data : F1 Score

[9]:
```
F1 = f1_score(y_test, y_pred, average = 'macro')
print(F1)
```

```
0.9997945273932791
```

Output Results

AI Model 1 – Synthetic Data : Feature Importance

[10]:
```
# Create a series with feature names and their importance scores
importances = pd.Series(model.feature_importances_, index=X_train.columns)

# Sort and show the top 20 features
top_features = importances.sort_values(ascending=False).head(20)
print(top_features)
```

```
NUM_PREVIOUS_FLIGHTS_LATE    0.732326
AVG_GATE_WAIT_TIME           0.208496
FL_DATE                      0.010131
HOUR                         0.006916
AIRLINE_Frontier             0.003505
AIRLINE_Delta                0.003491
ORIGIN_ATL                   0.002148
DEST_IAH                     0.002068
ORIGIN_PHX                   0.002025
AIRLINE_United               0.001991
AIRLINE_Southwest            0.001810
AIRLINE_Alaska               0.001808
DEST_SLC                     0.001743
AIRLINE_JetBlue              0.001670
AIRLINE_American             0.001561
DEST_EWR                     0.001515
ORIGIN_DEN                   0.001383
AIRLINE_Spirit               0.001372
ORIGIN_JFK                   0.001232
DEST_MSP                     0.001182
dtype: float64
```

Top Features

AI Model 2 – Real-World Data : Preprocessing Data

Imported and cleaned 1.6M real flight records, filling missing values

```
# Load dataset
df = pd.read_csv('flights_sample_3m.csv')

# Fill missing categorical columns with mode
for col in df.select_dtypes(include='object').columns:
    df[col] = df[col].fillna(df[col].mode()[0])

# Fill missing numeric columns with median
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    df[col] = df[col].fillna(df[col].median())

# --- Encoding ---

le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    if col != 'status':
        df[col] = le.fit_transform(df[col])

# Create binary target column for delay: 1 if delayed, 0 otherwise
df['status_Delayed'] = (df['status'] == 'Delayed').astype(int)

# --- Prepare features and target ---

X = df.drop(columns=['status', 'status_Delayed'])
y = df['status_Delayed']
```

AI Model 2 – Real-World Data : Data Set

flights_sample_3m.csv > data

```
flight_id,flight_no,scheduled_departure,scheduled_arrival,departure_airport,arrival_airport,status,aircraft_code,actual_departure,actual_arrival,seat_no,fare_conditions
1185,PG0134,2017-09-10 09:50:00+03,2017-09-10 14:55:00+03,DME,BTK,Scheduled,319,,,
3979,PG0052,2017-08-25 14:50:00+03,2017-08-25 17:35:00+03,VKO,HMA,Scheduled,CR2,,,
4739,PG0561,2017-09-05 12:30:00+03,2017-09-05 14:15:00+03,VKO,AER,Scheduled,763,,,
5502,PG0529,2017-09-12 09:50:00+03,2017-09-12 11:20:00+03,SVO,UFA,Scheduled,763,,,
6938,PG0461,2017-09-04 12:25:00+03,2017-09-04 13:20:00+03,SVO,ULV,Scheduled,SU9,,,
7784,PG0667,2017-09-10 15:00:00+03,2017-09-10 17:30:00+03,SVO,KRO,Scheduled,CR2,,,
9478,PG0360,2017-08-28 09:00:00+03,2017-08-28 11:35:00+03,LED,REN,Scheduled,CR2,,,
11085,PG0569,2017-08-24 15:05:00+03,2017-08-24 16:10:00+03,SVX,SCW,Scheduled,733,,,
11847,PG0498,2017-09-12 10:15:00+03,2017-09-12 14:55:00+03,KZN,IKT,Scheduled,319,,,
12012,PG0621,2017-08-26 16:05:00+03,2017-08-26 17:00:00+03,KZN,MQF,Scheduled,CR2,,,
13113,PG0612,2017-08-18 16:25:00+03,2017-08-18 20:05:00+03,ROV,KZN,Scheduled,CN1,,,
14806,PG0676,2017-09-06 07:05:00+03,2017-09-06 07:45:00+03,PEE,CEK,Scheduled,CR2,,,
16837,PG0010,2017-09-05 12:25:00+03,2017-09-05 14:35:00+03,JOK,VKO,Scheduled,CN1,,,
17173,PG0059,2017-09-14 12:25:00+03,2017-09-14 14:45:00+03,SCW,NBC,Cancelled,CN1,,,
19807,PG0035,2017-09-11 06:35:00+03,2017-09-11 09:25:00+03,MJZ,CNN,Scheduled,CN1,,,
23609,PG0648,2017-08-31 11:35:00+03,2017-08-31 13:00:00+03,UUA,SVO,Scheduled,CR2,,,
23695,PG0388,2017-08-26 10:55:00+03,2017-08-26 11:25:00+03,UUA,REN,Scheduled,CR2,,,
23780,PG0098,2017-09-02 06:50:00+03,2017-09-02 10:30:00+03,SWT,CEK,Scheduled,CN1,,,
23945,PG0076,2017-09-05 09:15:00+03,2017-09-05 11:50:00+03,EYK,DME,Scheduled,CR2,,,
24705,PG0632,2017-08-26 15:00:00+03,2017-08-26 17:35:00+03,TJM,PES,Scheduled,CR2,,,
25382,PG0556,2017-08-31 09:05:00+03,2017-08-31 10:55:00+03,IKT,MJZ,Scheduled,CR2,,,
26057,PG0467,2017-08-26 15:45:00+03,2017-08-26 17:00:00+03,ULY,RTW,Scheduled,CN1,,,
27580,PG0483,2017-09-12 07:20:00+03,2017-09-12 11:20:00+03,KEJ,DME,Scheduled,SU9,,,
29272,PG0334,2017-09-05 16:20:00+03,2017-09-05 19:05:00+03,KGD,ESL,Scheduled,CR2,,,
29440,PG0065,2017-09-02 12:15:00+03,2017-09-02 18:05:00+03,UUD,VKO,Scheduled,319,,,
32658,PG0674,2017-08-19 09:35:00+03,2017-08-19 12:00:00+03,KRO,KJA,Scheduled,CR2,,,
1,PG0405,2017-07-16 09:35:00+03,2017-07-16 10:30:00+03,DME,LED,Arrived,321,2017-07-16 09:44:00+03,2017-07-16 10:39:00+03,,
2,PG0404,2017-08-05 19:05:00+03,2017-08-05 20:00:00+03,DME,LED,Arrived,321,2017-08-05 19:06:00+03,2017-08-05 20:01:00+03,,
3,PG0405,2017-08-05 09:35:00+03,2017-08-05 10:30:00+03,DME,LED,Arrived,321,2017-08-05 09:39:00+03,2017-08-05 10:34:00+03,,
```

AI Model 2 – Real-World Data : Train/test splitting and Balancing Classes

```python
# --- Train/test split ---

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y)

# --- Handle class imbalance by undersampling majority class ---

# Combine X_train and y_train for easier resampling
train_data = pd.concat([X_train, y_train], axis=1)

# Separate majority and minority classes
majority = train_data[train_data.status_Delayed == 0]
minority = train_data[train_data.status_Delayed == 1]

print("Before undersampling:", majority.status_Delayed.value_counts(), minority.status_Delayed.value_counts())

# Downsample majority class
majority_downsampled = resample(
    majority,
    replace=False,
    n_samples=len(minority),
    random_state=42
)

# Combine minority class with downsampled majority class
undersampled = pd.concat([majority_downsampled, minority])

print("After undersampling:", undersampled.status_Delayed.value_counts())

# Split back into X and y
X_train_bal = undersampled.drop('status_Delayed', axis=1)
```

Applied resampling to address class imbalance

AI Model 2 – Real-World Data : Training Model and Accuracy

```python
# --- Train RandomForestClassifier model ---

clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_bal, y_train_bal)

# --- Evaluate model ---

y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, digits=4))
```

RandomForestClassifier achieved 92% accuracy but F1 score for delays remained low (0.03), highlighting real-world challenges

Output Results

```
Before undersampling: status_Delayed
0    24093
Name: count, dtype: int64 status_Delayed
1    29
Name: count, dtype: int64
After undersampling: status_Delayed
0    29
1    29
Name: count, dtype: int64
Accuracy: 0.9235828980460438
              precision    recall  f1-score   support

           0     1.0000    0.9235    0.9602     10326
           1     0.0150    1.0000    0.0295        12

    accuracy                         0.9236     10338
   macro avg     0.5075    0.9617    0.4949     10338
weighted avg     0.9989    0.9236    0.9591     10338
```

F1 score for “Not Delayed” class: 0.9602

F1 score for “Delayed” class: 0.0295

AI Model 2 – Real-World Data : Top 10 Important Features

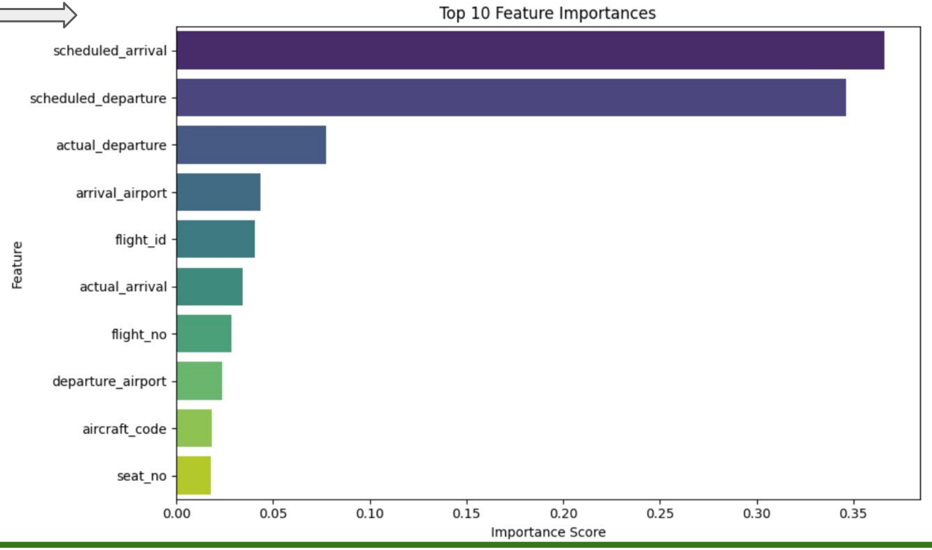
[5]:

```python
# --- Feature Importance from trained model ---
importances = clf.feature_importances_
feature_names = X_train.columns

# Create DataFrame of features and their importances
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Print top 10 most important features
print("Top 10 Important Features:")
print(feature_importance_df.head(10))

# --- Plotting ---
plt.figure(figsize=(10, 6))
sns.barplot(
    data=feature_importance_df.head(10),
    x='Importance',
    y='Feature',
    hue='Feature',
    dodge=False,
    palette='viridis',
    legend=False
)
plt.title('Top 10 Feature Importances')
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

Top 10 Important Features:

	Feature	Importance
3	scheduled_arrival	0.366114
2	scheduled_departure	0.346360
7	actual_departure	0.077389
5	arrival_airport	0.043414
0	flight_id	0.040492
8	actual_arrival	0.034601
1	flight_no	0.028396
4	departure_airport	0.023876
6	aircraft_code	0.018329
9	seat_no	0.017972

70% delays from scheduling



Reflection

- The limitations of synthetic vs. real-world datasets.
- The difficulty of generalizing models to imbalanced, messy real data.
- Building, debugging, and refining the engineering pipeline mattered as much as accuracy itself.

Demo and Project links

- **GitHub Repo**: https://github.com/Gordonandric/Flight-Delay-Prediction-Modeling
- **Kaggle:**
  https://www.kaggle.com/writeups/gordonandric/flight-delay-prediction-modeling
  https://www.kaggle.com/code/gordonandric/airline-synthetic
  https://www.kaggle.com/datasets/gordonandric/synthetic-airline-data
  https://www.kaggle.com/code/gordonandric/airline-real
  https://www.kaggle.com/datasets/gordonandric/real-airline-data/

- **Demo:**
  https://github.com/Gordonandric/Flight-Delay-Prediction-Modeling/blob/main/Airline_Synthetic.webm
  https://github.com/Gordonandric/Flight-Delay-Prediction-Modeling/blob/main/Airline_Real.webm

Thank You