PRC Data Challenge

Outline

- EDA
- Data Preprocessing
- Initial Modeling
- ML Pipeline

Outline

- EDA
- Data Preprocessing
- Initial Modeling
- ML Pipeline

EDA

(Only Meaningful)

Dataset

Trajectory

- Specific Flights
  - Altitude
  - Ground Speed
  - Vertical Rate
  - Component of wind
  - Temp/Humid
- Use to find Thrust - Drag
- 150+ GB

Challenge (Train)

- Flight lists
  - Aircraft Type
  - WTC
  - Flight Duration
  - Taxiout Time
  - Flown Distance
  - TOW
- 369,013 flights

Submission (Test)

- Same as Train but no “TOW”
- 105,959 flights

Missing Value

Trajectory

- 19,432 missing for
  - Ground Speed
  - Track
  - Vertical Rate
- 0.33 %

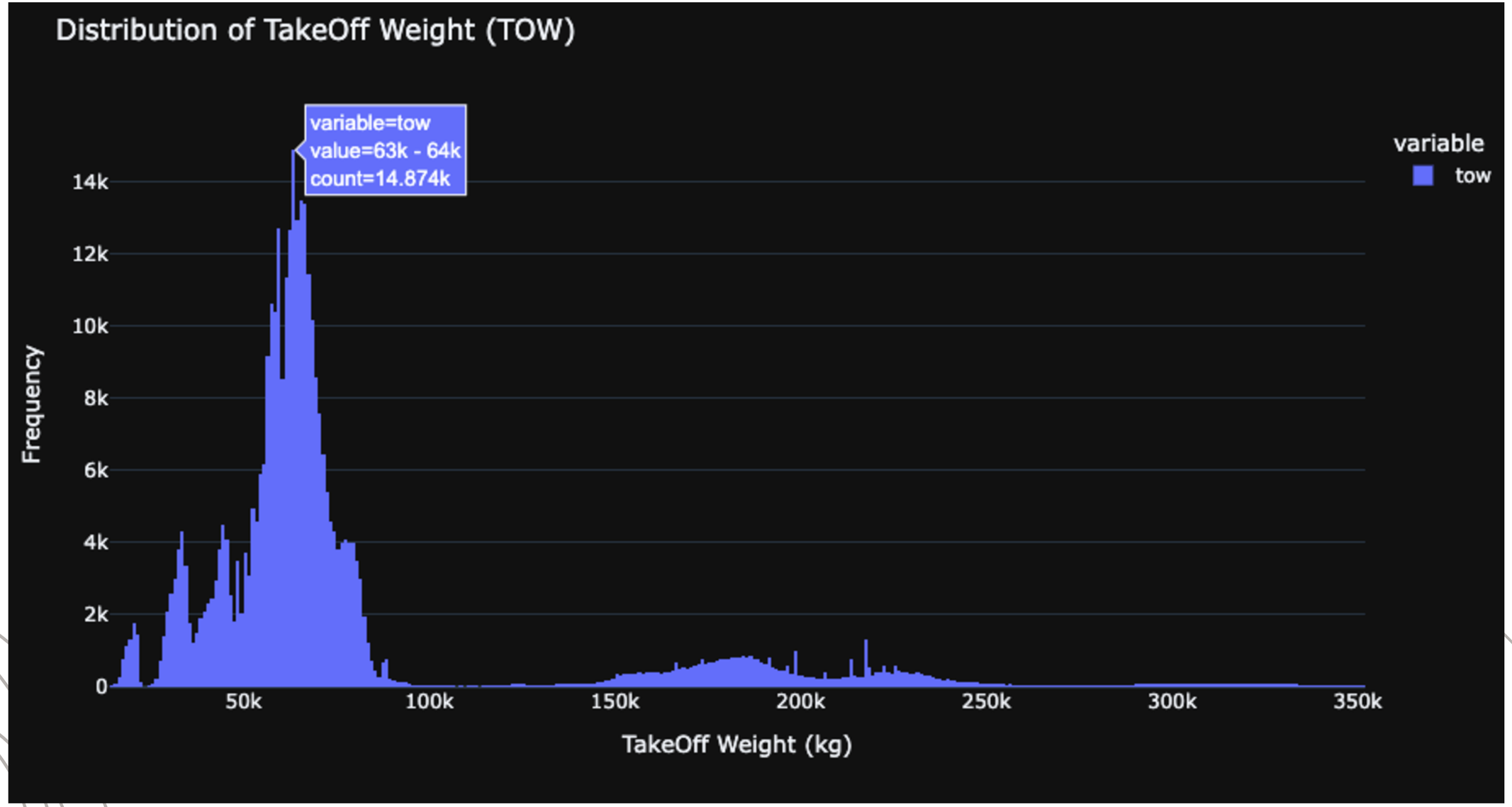
Challenge (Train)

- No

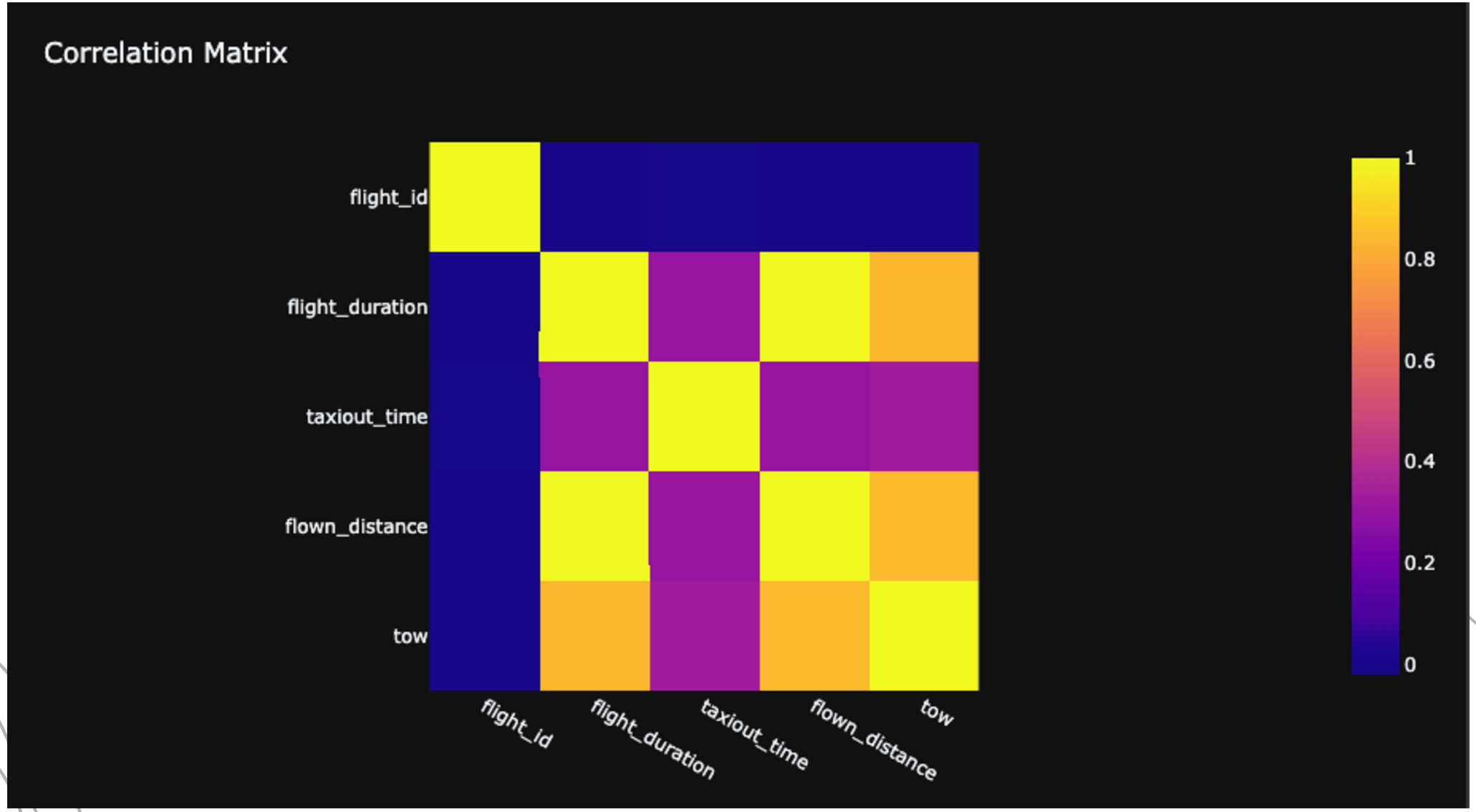
Submission (Test)

- No

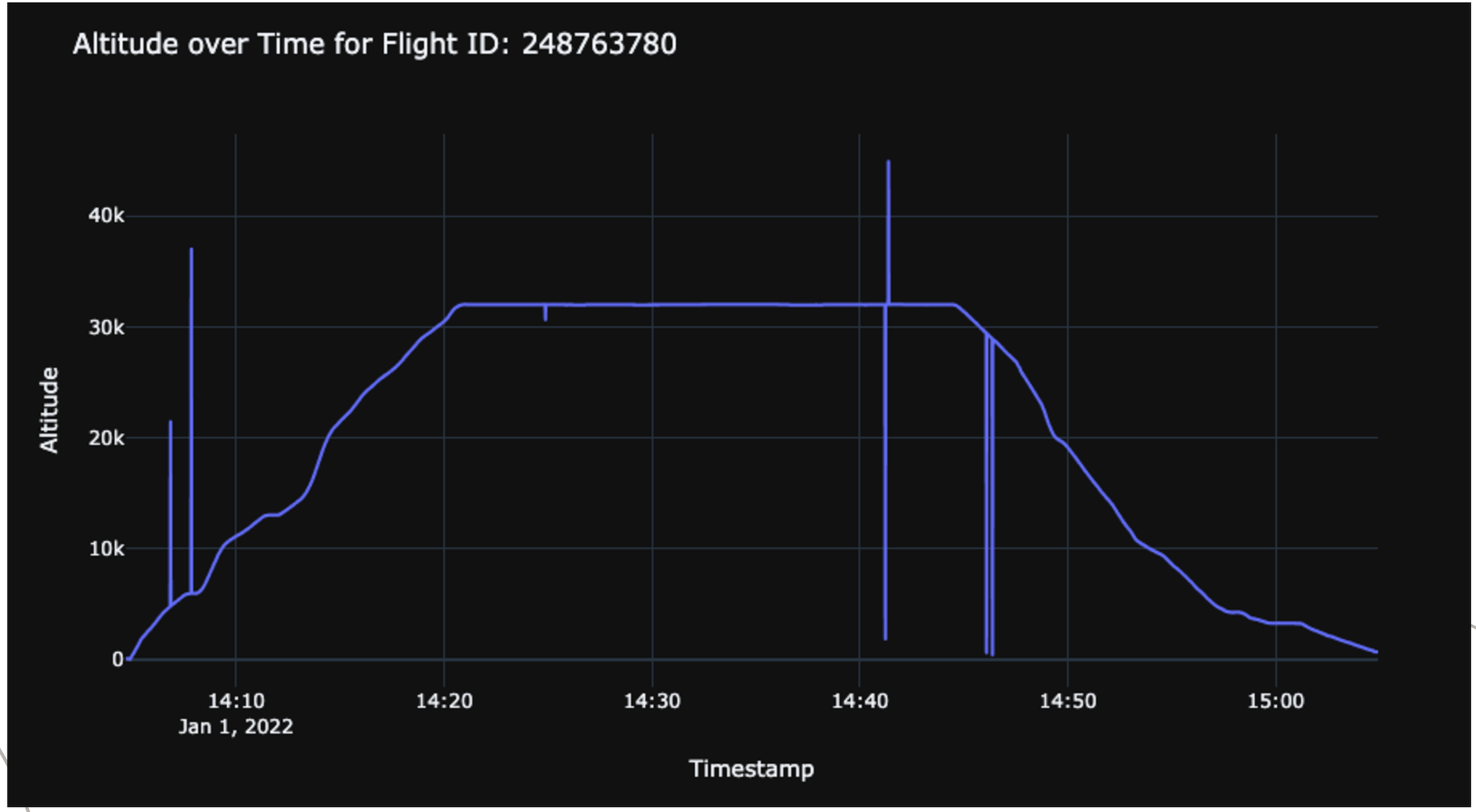
TOW Distribution (Train)



Correlation Matrix (Train)



Sample Altitude for one figh (Trajectory)



Outline

- EDA
- Data Preprocessing
- Initial Modeling
- ML Pipeline

Data Preprocessing

(Only Meaningful)

Can we get initial weight for each aircraft_type?

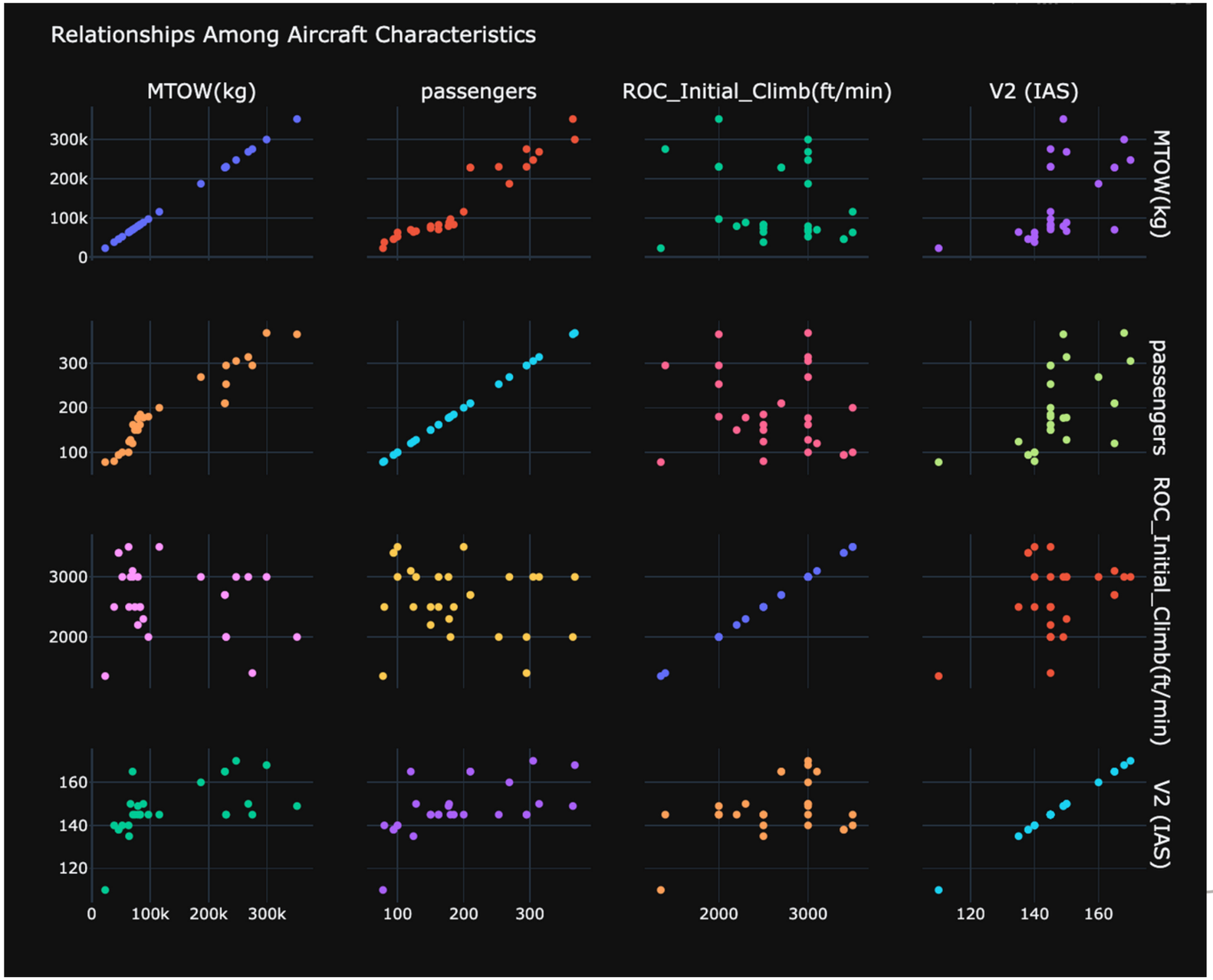
Yes

Aircraft Performance Database

Can we get initial weight for each aircraft_type?

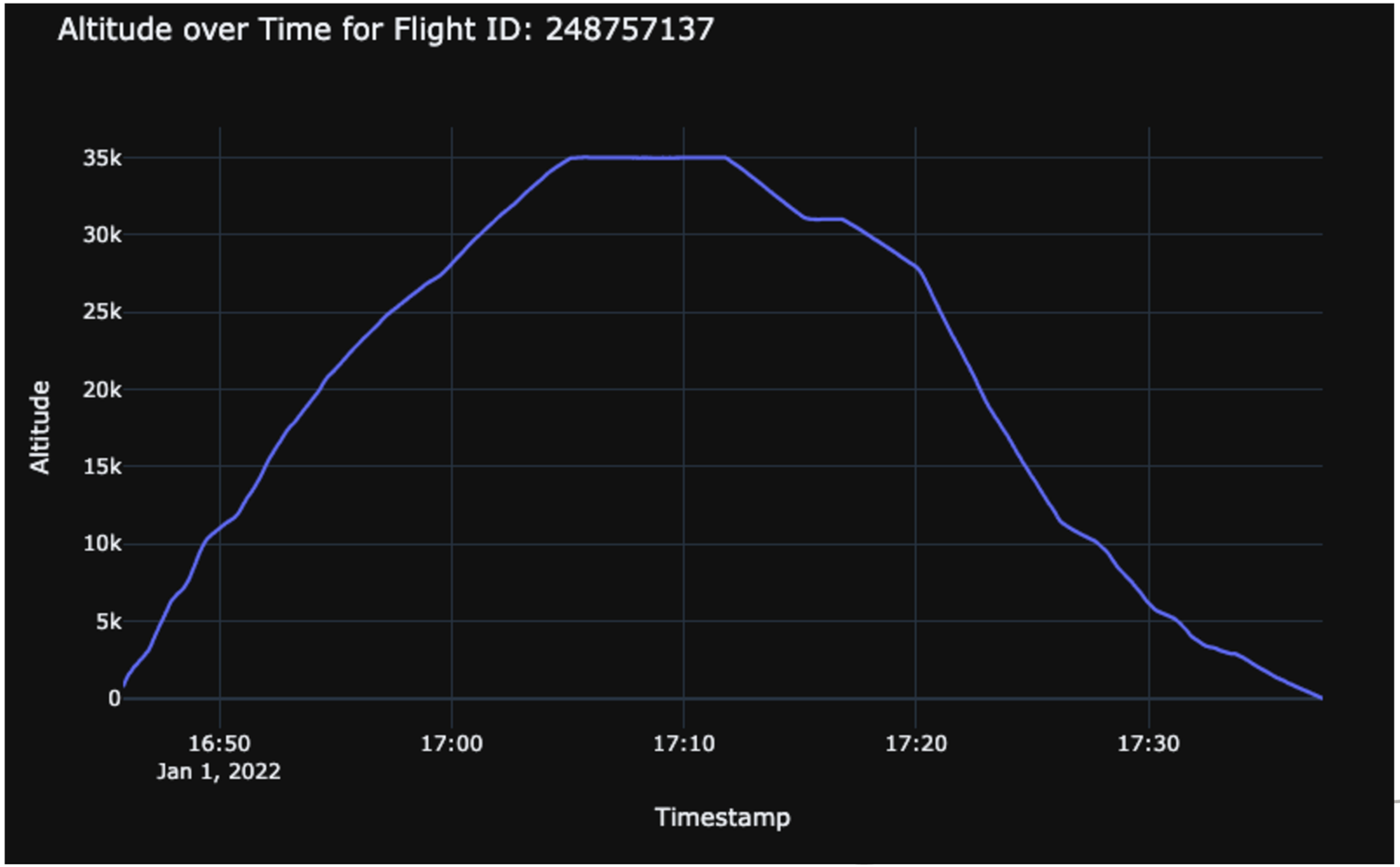
```
external_information = {
  "B738": {
      "MTOW(kg)": 70530,
      "passengers": 162,
      "ROC_Initial_Climb(ft/min)": 3000,
      "V2 (IAS)": 145,
  },
```

Can we get initial weight for each aircraft_type?



Remove Outliers (Errors)

- Group the data in the same flight to remove outliers.
- Traditional method (IQR, Z-Score) is prolonged since we have a lot of data.
- Use isolation forest algorithm instead. (Efficient, scales well with large datasets)



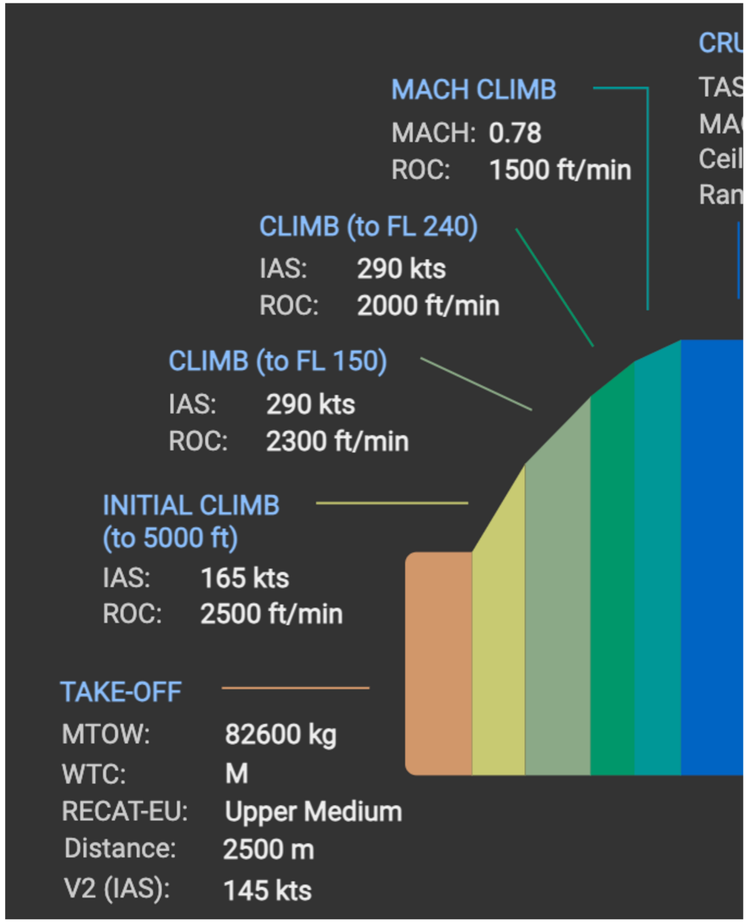
Find Thrust – Drag

1. Segment the trajectory into flight phases (Separate by the altitude)
    a. takeoff
    b. climb
    c. cruise
    d. descent
    e. landing
2. Focus on the `takeoff and initial climb phases`
3. Apply the formula from P’Ta
4. Merge with train data.

Find Thrust - Drag

```python
def identify_flight_phases(group, flight_phases_refinement=False):
  group = group.sort_values('timestamp').reset_index(drop=True)
  group['altitude_diff'] = group['altitude'].diff()

  # Applies a Savitzky-Golay filter to smooth the altitude profile, reducing noise.
  altitude_smooth = signal.savgol_filter(group['altitude'],
                                         window_length=min(21, len(group) // 2 * 2 + 1),
                                         polyorder=3)

  # Calculate the rate of climb (ROC) (gradient[i] = (f[i+1] - f[i-1]) / (x[i+1] - x[i-1]))
  group['ROC'] = np.gradient(altitude_smooth, group['timestamp'].astype(int) / 10**9)
  max_altitude = group['altitude'].max()
  takeoff_end = group[group['altitude'] > group['altitude'].quantile(0.1)].index[0]
  top_of_climb = group[group['altitude'] > max_altitude * 0.95].index[0]

  takeoff_phase = group.loc[:takeoff_end]
  initial_climb_phase = group.loc[takeoff_end:top_of_climb]
  cruise_phase = group.loc[top_of_climb:]

  if flight_phases_refinement:
    # Refine takeoff phase (focus on the most significant part of the takeoff)
    # The assumption is that the aircraft's weight (ATOW) will have the most significant impact during this high-ROC portion of takeoff.
    takeoff_phase = takeoff_phase[takeoff_phase['ROC'] > takeoff_phase['ROC'].quantile(0.5)]

    # Refine initial climb phase
    initial_climb_phase = initial_climb_phase[
        (initial_climb_phase['ROC'] > initial_climb_phase['ROC'].quantile(0.25)) &
        (initial_climb_phase['altitude'] < max_altitude * 0.8)
    ]

  return takeoff_phase, initial_climb_phase, cruise_phase
```

Find Thrust – Drag

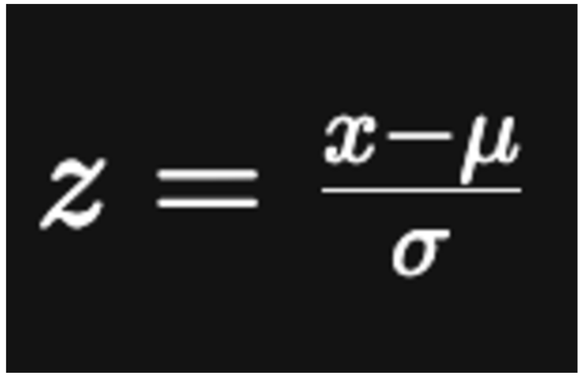


Aggregate Features

- Aggregate feature in each group based on the flight ID
- We use mean, max, and std for each of the numerical variables.
- Merged into train data.

Normalization

- Normalize all the numerial variables using StandardScaler (Z-score normalization)



Categorical Encoding

- Convert text variables into numbers.
- From the unique values, we only convert the wtc and ignore the others.

```
callsign 10860
adep 457
name_adep 457
country_code_adep 102
ades 362
name_ades 362
country_code_ades 76
aircraft_type 28
wtc 2
airline 29
```

Outline

- EDA
- Data Preprocessing
- Initial Modeling
- ML Pipeline

Initial Modeling

**(Just to find the baseline,
Similar parameters, No tuning)**

Initial Modeling

Case 1: Use 10% of trajectory data (Only one file)

CatBoost

RMSE:
16984.43

XGB

RMSE:
17802.60

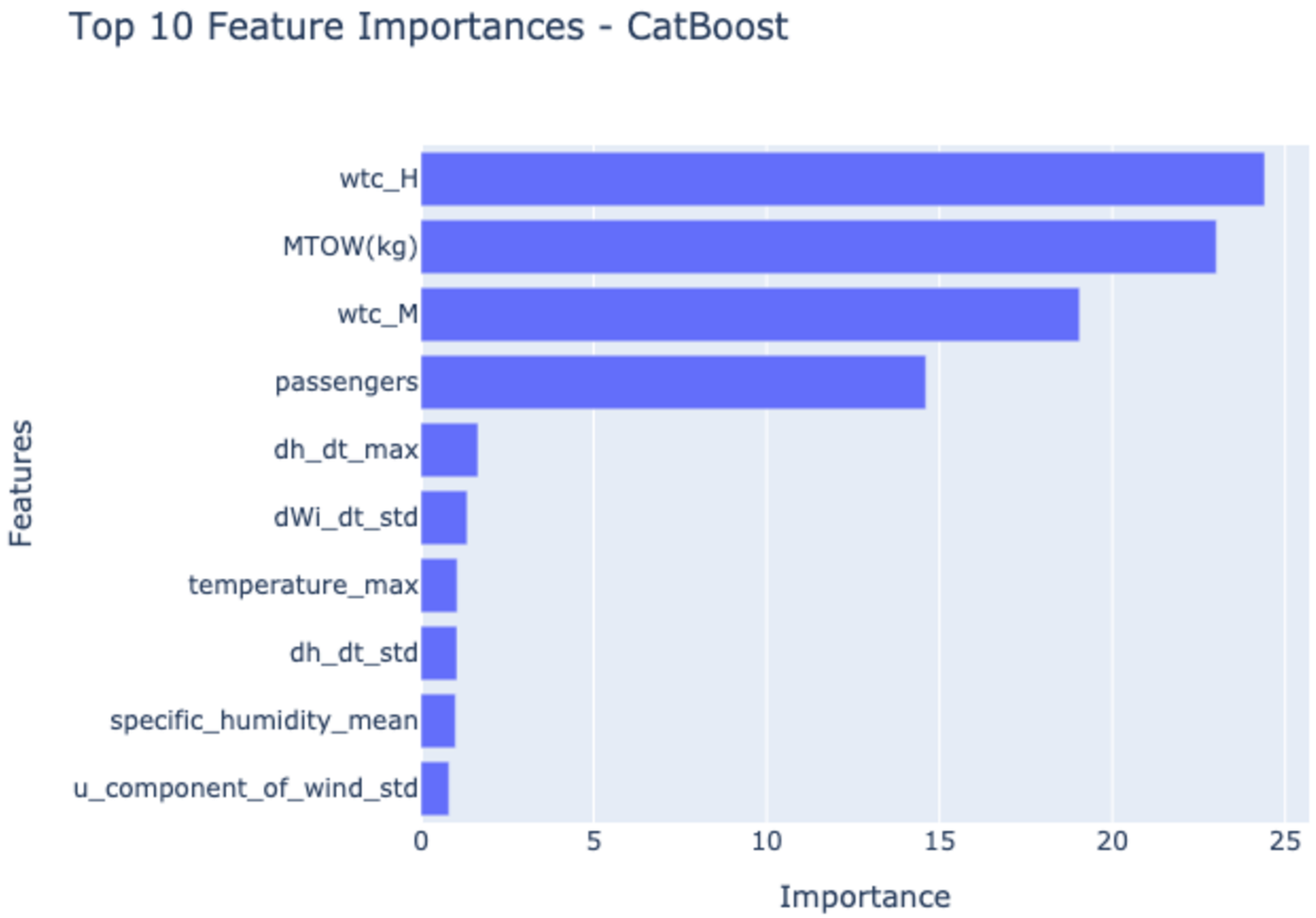
RandomForest

RMSE:
12465.82

Initial Modeling

Case 1: Use 10% of trajectory data (Only one file)

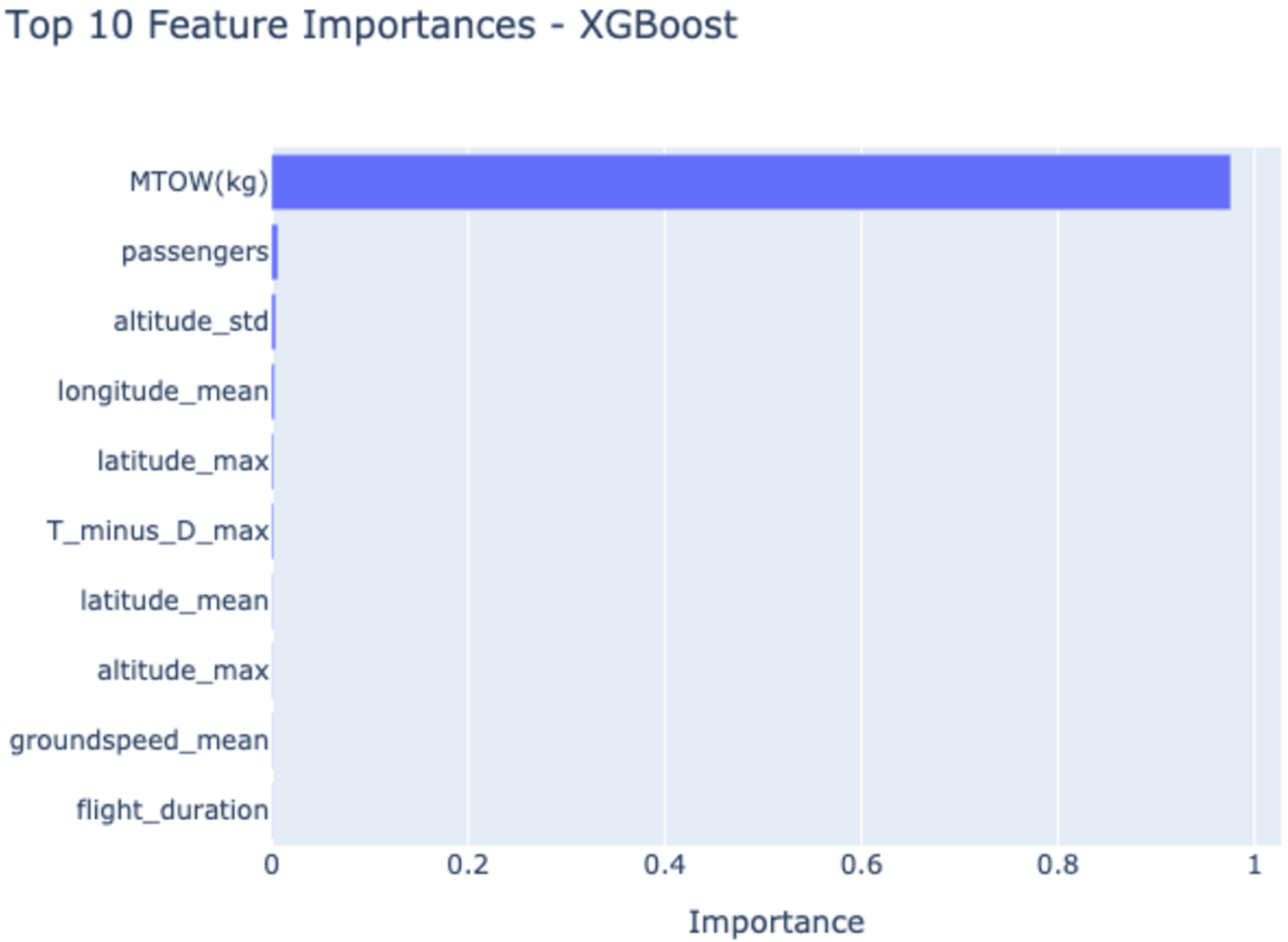
CatBoost



Initial Modeling

Case 1: Use 10% of trajectory data (Only one file)

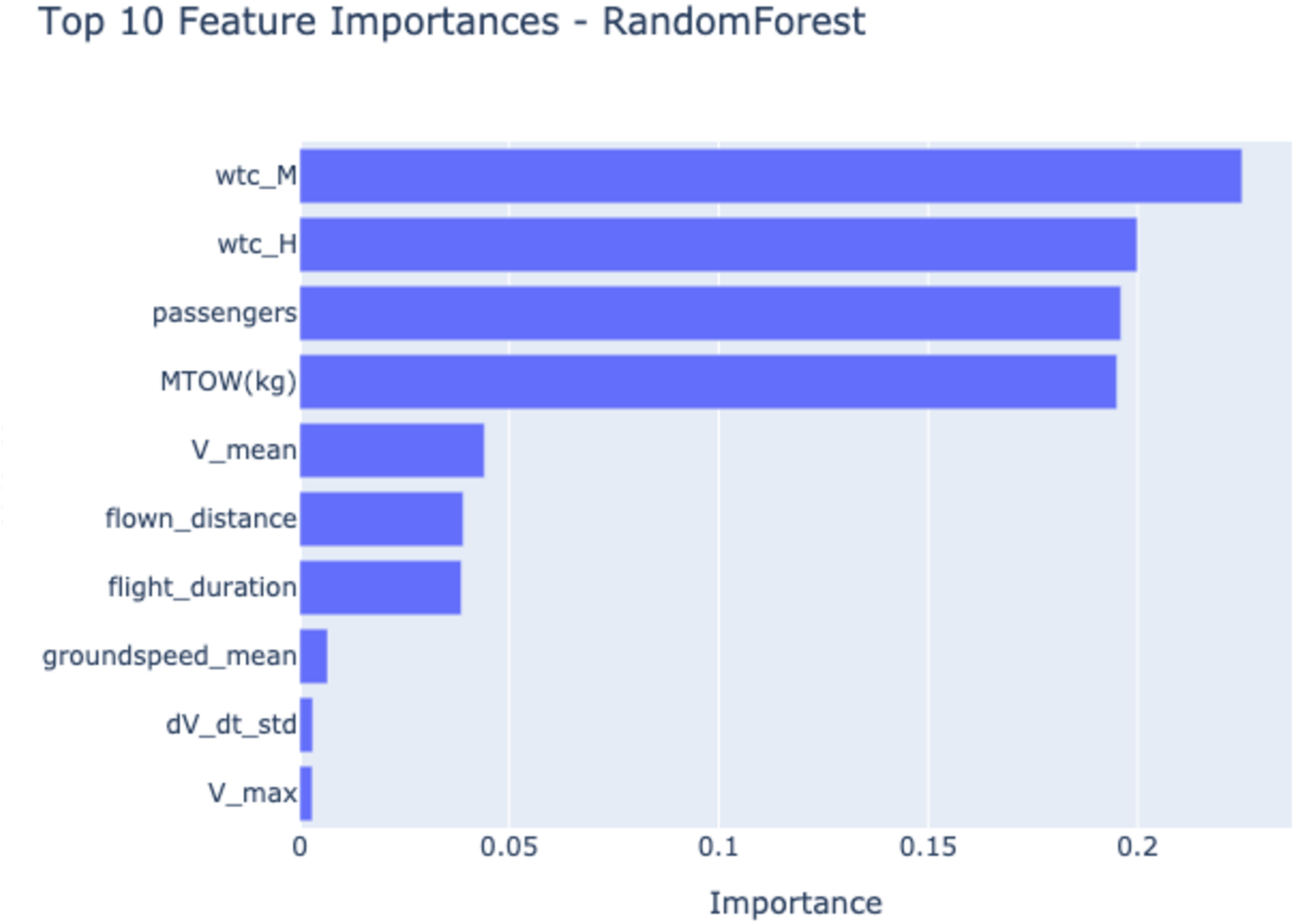
XGB



Initial Modeling

Case 1: Use 10% of trajectory data (Only one file)

Random Forest



Initial Modeling

Case 2: Use 100% of trajectory data (Only one file)

CatBoost

RMSE: 7107.75

XGB

RMSE: 7101.09

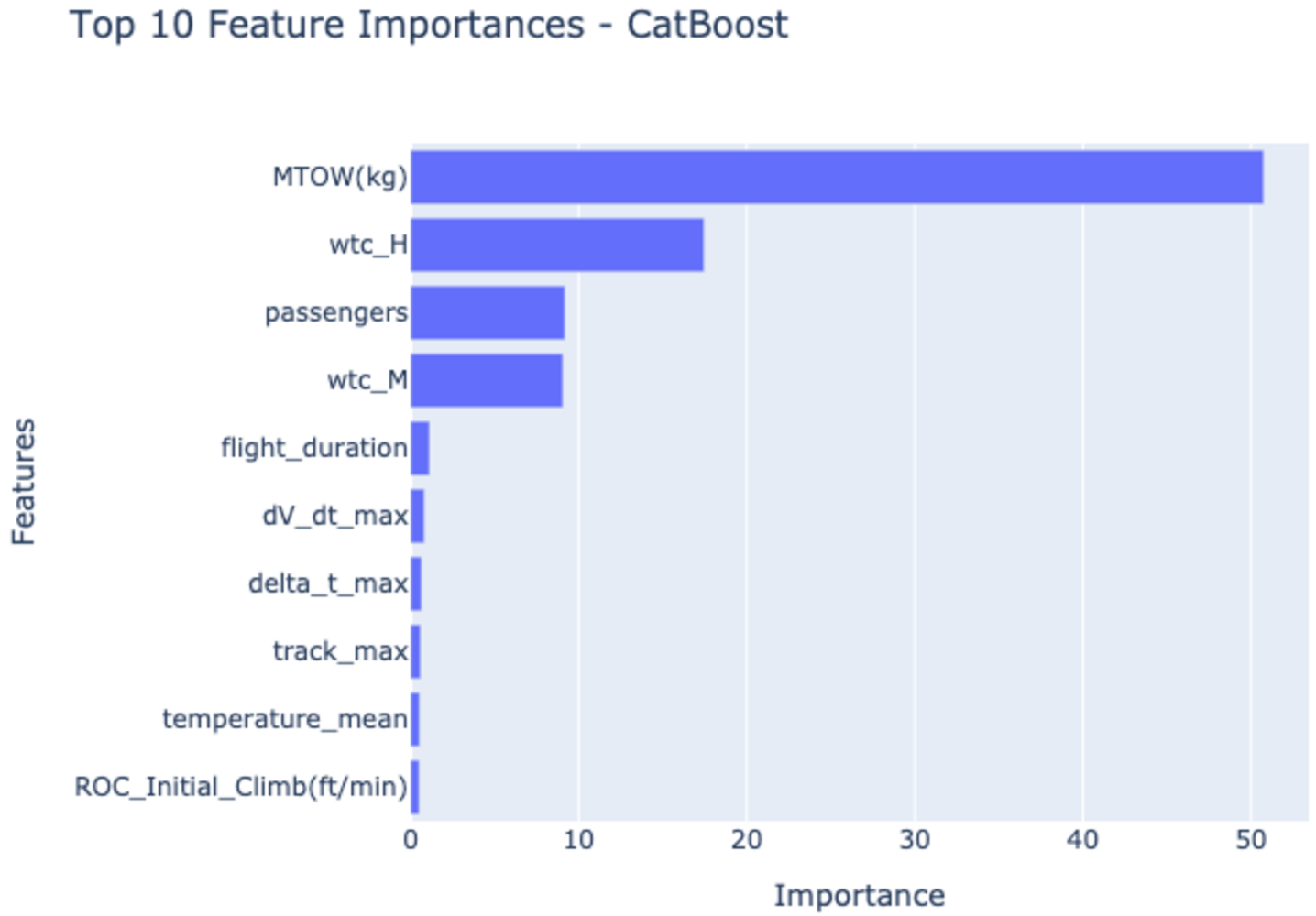
RandomForest

RMSE: 7740.56

Initial Modeling

Case 2: Use 100% of trajectory data (Only one file)

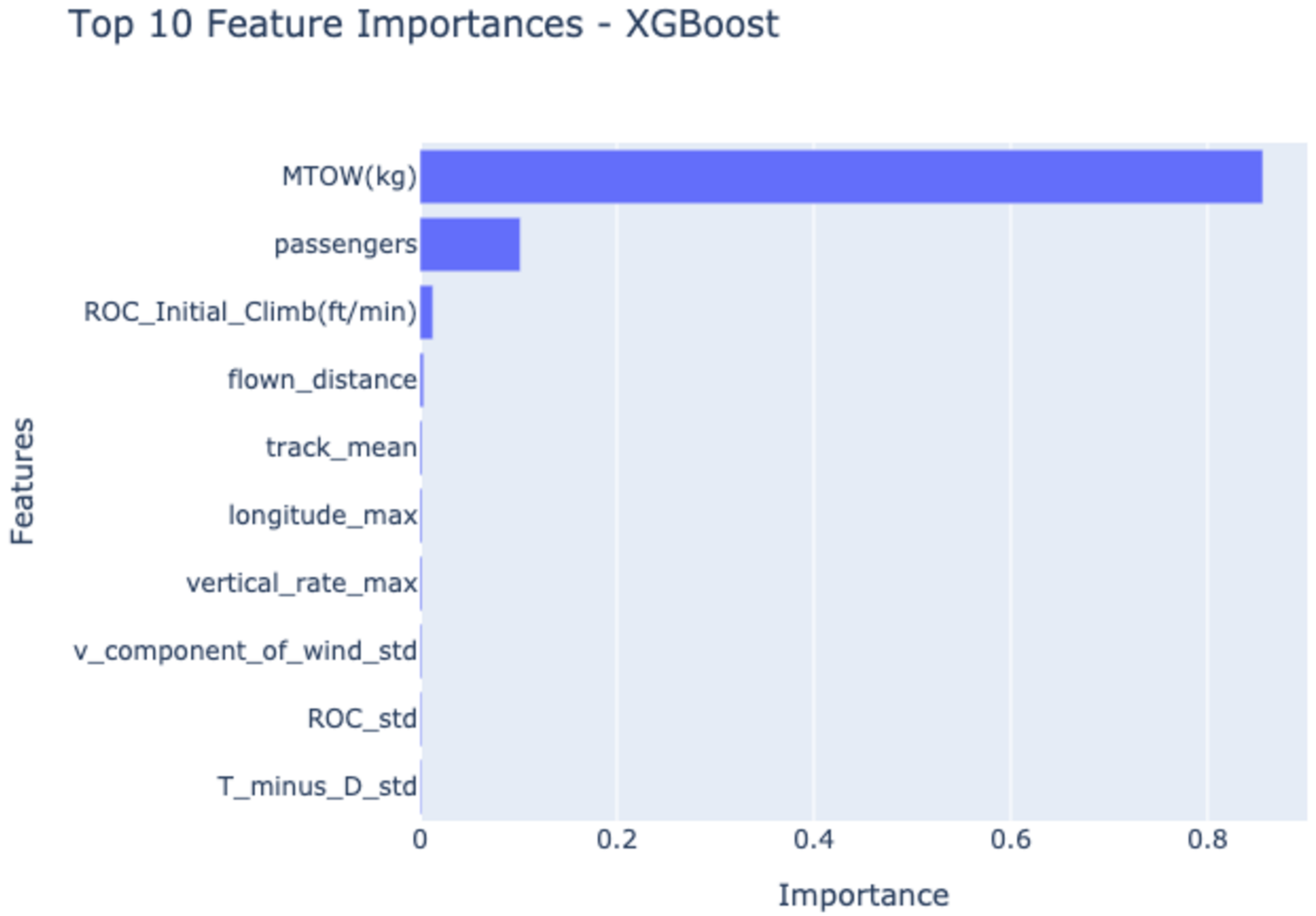
CatBoost



Initial Modeling

Case 2: Use 100% of trajectory data (Only one file)

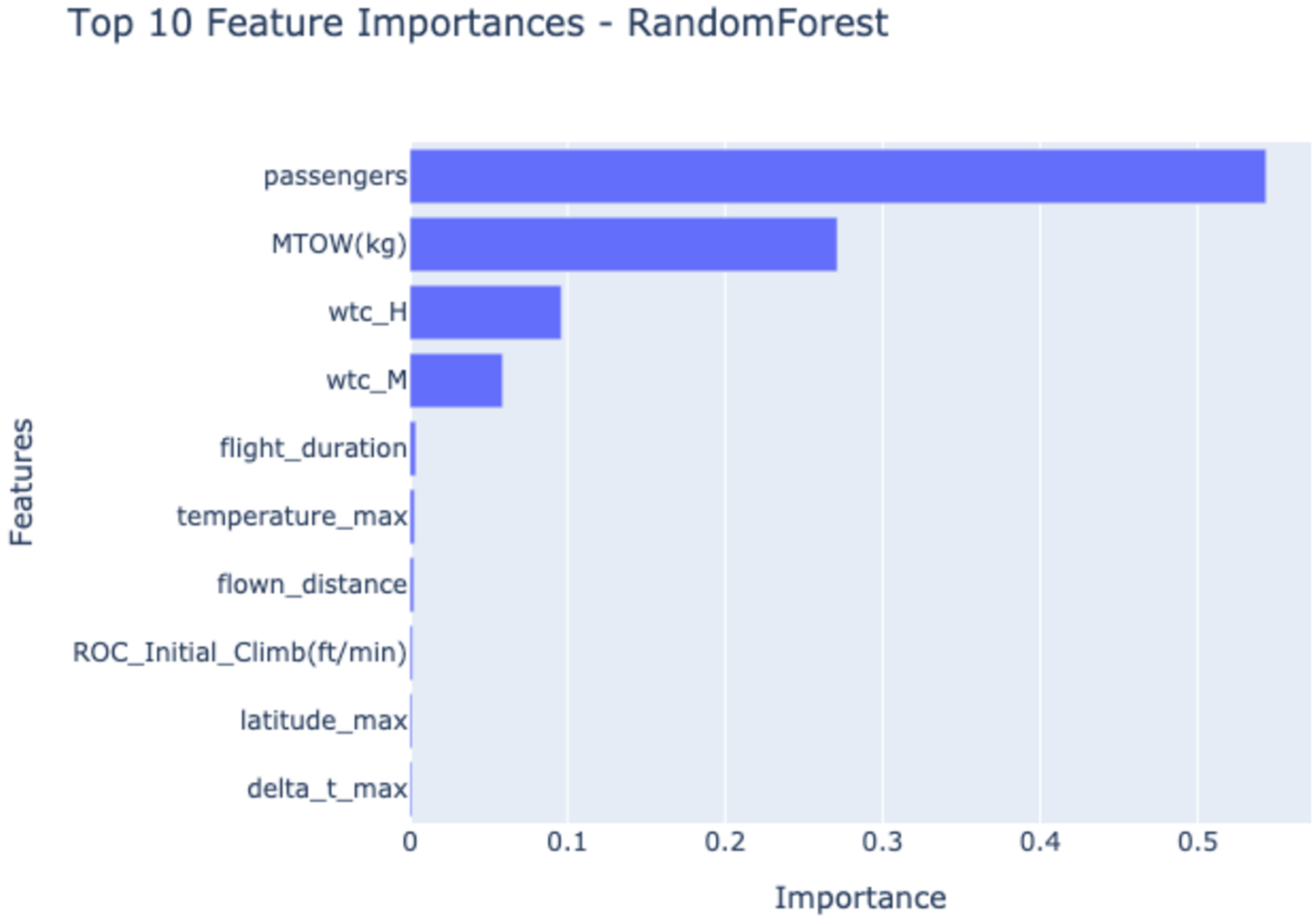
XGB



Initial Modeling

Case 2: Use 100% of trajectory data (Only one file)

Random Forest



Initial Modeling

Case 3: Use 0% of trajectory data (No Thrust - Drag)

CatBoost

RMSE:
5393.07

XGB

RMSE:
5368.75

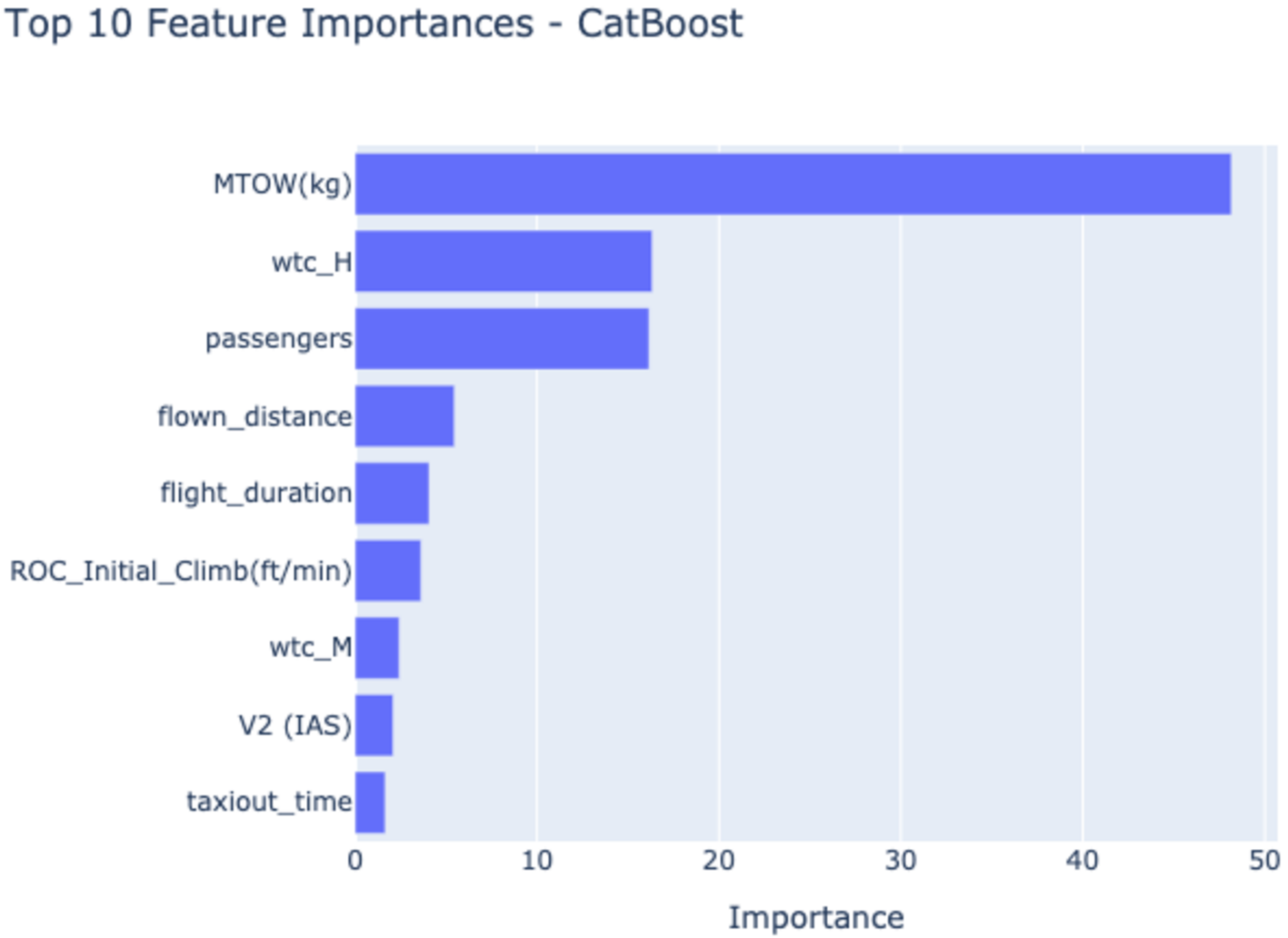
RandomForest

RMSE:
6642.23

Initial Modeling

Case 3: Use 0% of trajectory data (No Thrust – Drag)

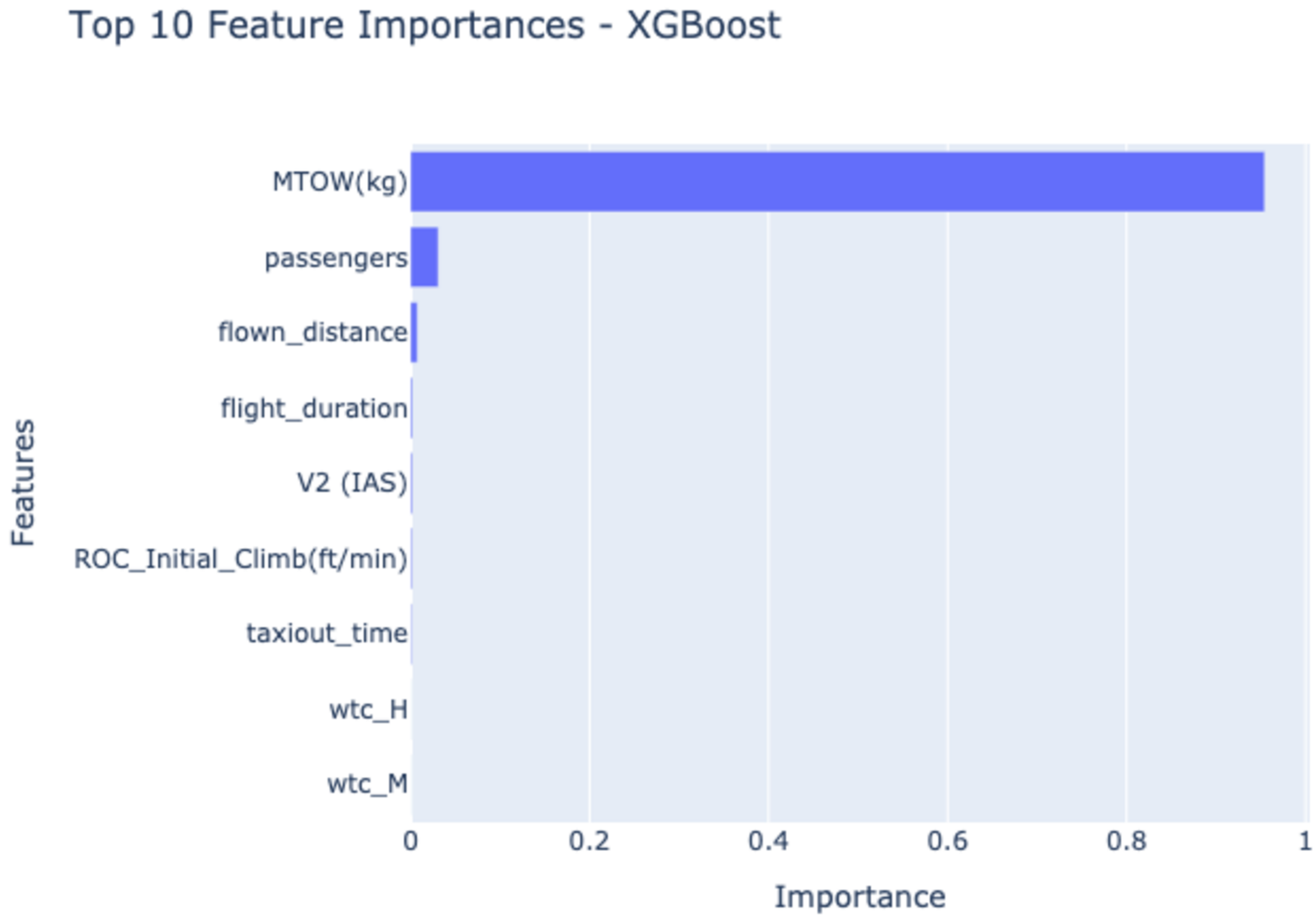
CatBoost



Initial Modeling

Case 3: Use 0% of trajectory data (No Thrust - Drag)

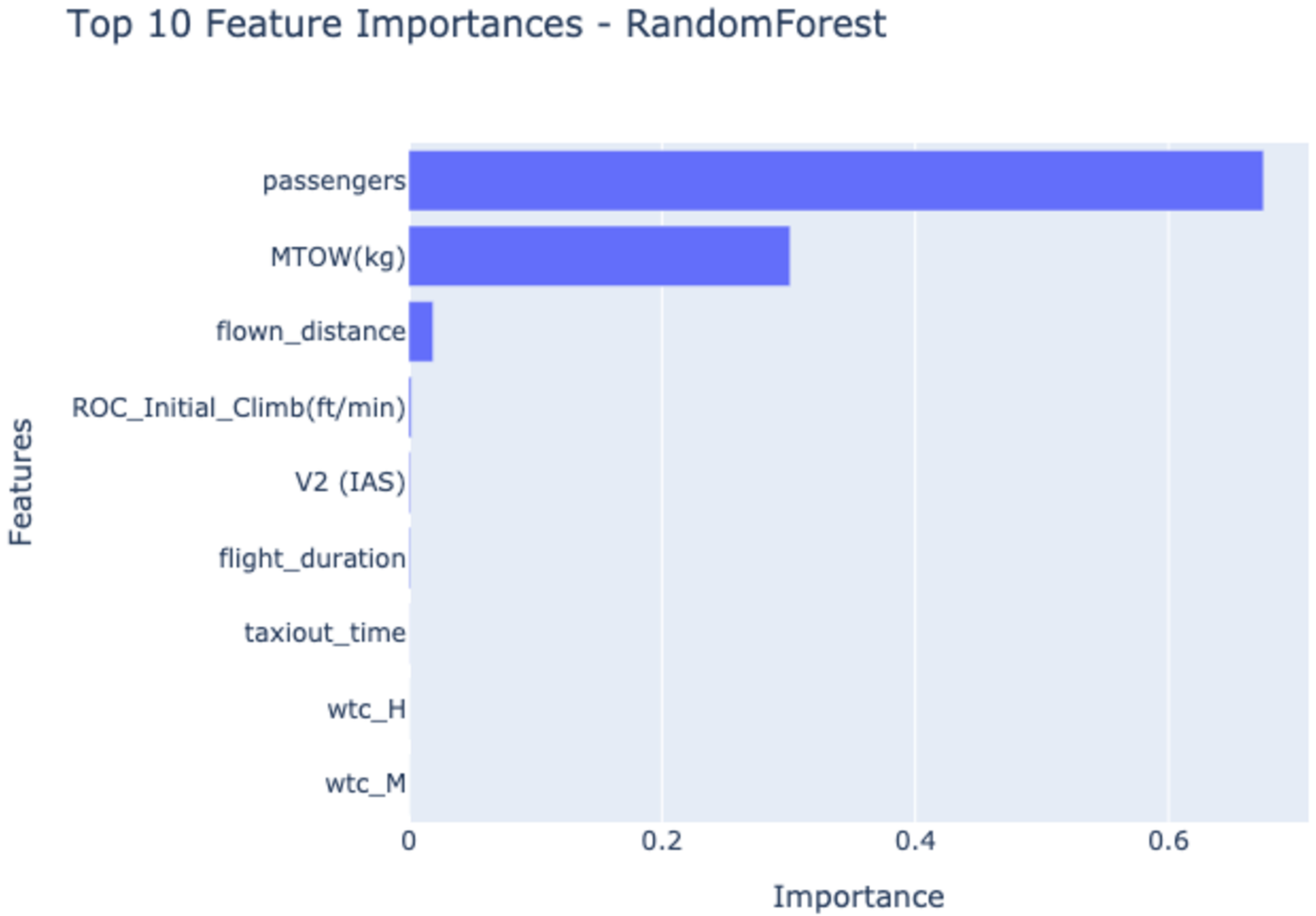
XGB



Initial Modeling

Case 3: Use 0% of trajectory data (No Thrust - Drag)

Random Forest



Outline

- EDA
- Data Preprocessing
- Initial Modeling
- ML Pipeline

ML Pipeline

(Demo)