W261 Final Project - Predicting Flight Delays

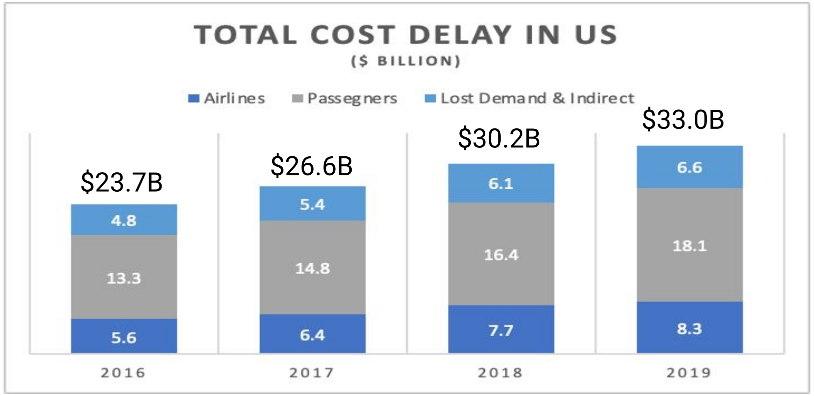
Team04:
Brittany Dougall, Jesse Miller, Chris Skokowski,
Fengjiao Sun

Introduction

- Business Case:
  - maximize the **accuracy, f1, recall** in predicting flight delays for passengers

- Datasets
  - Airlines data (primary resources for Time Series features)
  - Weather data (unused by model)

Passenger-centric Approach



Evaluation metrics

- **Accuracy = (TN + TP) / (TN + TP + FN + FP)**
- **F1 = TP / (TP+0.5(FP+FN))**
- **Recall = TP/(TP+FN)**

Define question

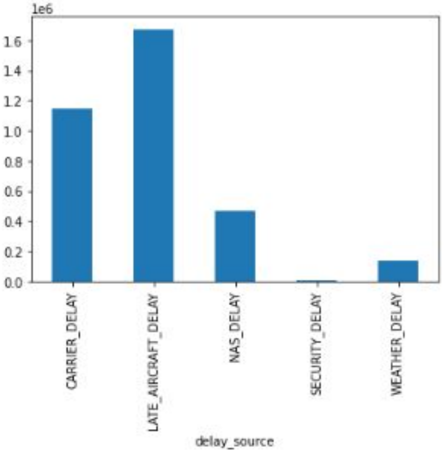
$$Y = \begin{cases} 0, \text{if flight is on-time} \\ 1, \text{if Dep\_Del15, Cancelled, or Diverted} \end{cases}$$

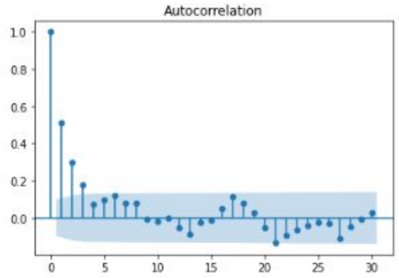
EDA

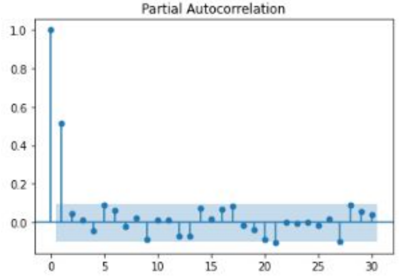
Key Findings:

- Time-based features key
- Significant autocorrelation for response var
- Late aircraft arrival is the most frequently listed departure delay cause, i.e. departure delays can often be predicated by delays further upstream

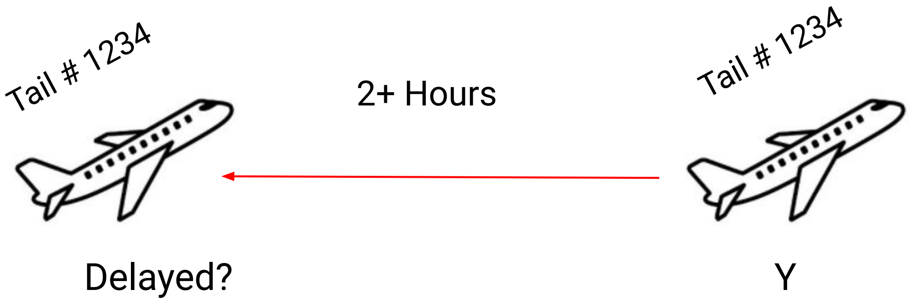
TAIL_NUM	utc_scheduled_departure	DEP_DELAY	CANCELLED	OUTCOME
N832AS	2015-01-22T00:12:00.000+0000	-7	0	0
N832AS	2015-01-22T02:57:00.000+0000	7	0	0
N832AS	2015-01-22T10:55:00.000+0000	-6	0	0
N832AS	2015-01-22T13:16:00.000+0000	-6	0	0
N832AS	2015-01-22T15:15:00.000+0000	-3	0	0
N832AS	2015-01-24T13:37:00.000+0000	123	0	1
N832AS	2015-01-24T15:11:00.000+0000	104	0	1
N832AS	2015-01-25T13:37:00.000+0000	98	0	1
N832AS	2015-01-25T15:11:00.000+0000	77	0	1
N832AS	2015-01-26T17:31:00.000+0000	75	0	1
N832AS	2015-01-26T19:32:00.000+0000	114	1	1
N832AS	2015-01-27T13:16:00.000+0000	null	1	1
N832AS	2015-01-27T15:15:00.000+0000	null	1	1
N832AS	2015-01-27T17:32:00.000+0000	null	1	1
N832AS	2015-01-27T19:27:00.000+0000	null	1	1
N832AS	2015-01-28T02:50:00.000+0000	-8	0	0
N832AS	2015-01-28T11:01:00.000+0000	-6	0	0
N832AS	2015-01-28T13:32:00.000+0000	-12	0	0
N832AS	2015-01-28T19:00:00.000+0000	-1	0	0
N832AS	2015-01-28T22:19:00.000+0000	-2	0	0







Feature Engineering




```
df = df.withColumn('last_flight_delayed',
                   when(((col('last_flight_scheduled_dep') + expr('INTERVAL + 15 MINUTES')) > col('utc_2hrs_before')), -1)\
                   .when(((col('last_flight_scheduled_dep') + expr('INTERVAL + 15 MINUTES')) <= col('utc_2hrs_before')), col('last_flight_outcome'))\
                   .otherwise(-1))
```

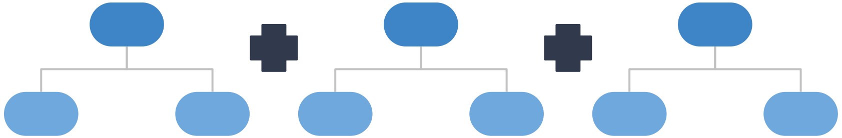
```
df = df.withColumn('arrival_time_2_next_flight',
                   when(col('last_flight_scheduled_arrival').isNotNull(),
                        col('utc_scheduled_departure').cast("long") - col('last_flight_scheduled_arrival').cast("long"))\
                   .otherwise(-9999))
```

Algorithms Tried

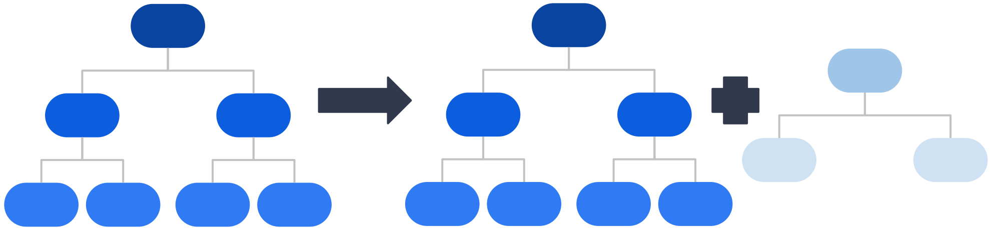
Validation Scores:

- **Logistic Regression**
  - Accuracy: 0.759
  - Recall: 0.639
  - F1: 0.510
- **Decision Tree**
  - Accuracy: 0.808
  - Recall: 0.664
  - F1: 0.577
- **Random Forest**
  - Accuracy: 0.809
  - Recall: 0.597
  - F1: 0.551
- **Gradient Boosted Decision Tree**
  - Accuracy: 0.812
  - Recall: 0.695
  - F1: 0.592

Random Forest



XGBoost

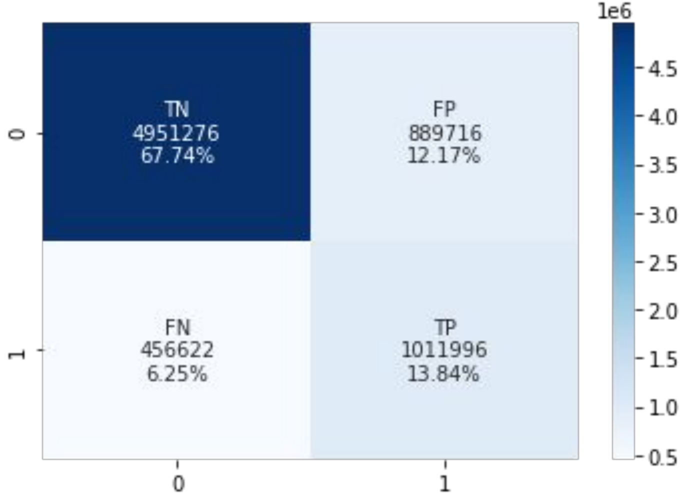


Outcome / Evaluation Metrics

- ~18% of observations misclassified
    - Highest rates of misclassified flights in June - Aug
    - FN for long intervals between arrival & next scheduled departure

- ~69 % accuracy in predicting delays 15+min/diversions/cancellations

Metric	Train	Test
F1-score	0.753	0.600
Recall	0.699	**0.689**
Precision	0.817	0.532
Accuracy	77.5%	81.6%



Gap Analysis

- **3rd** Highest F1: 0.600
- **3rd** Highest Recall: 0.689
- **3rd** Highest Accuracy: 0.819

Metric	Train	Test
F1-score	0.753	0.600
Recall	0.699	0.689
Precision	0.817	0.532
Accuracy	77.5%	81.6%

- Despite our focus on the recall metric, our model performed well across all three of the main metrics

- No model outperformed ours on every metric, indicating each had different priorities

Challenges

- Class imbalance
  - SMOTE
- Crashing XGBoost
- Additional feature engineering
  - Weather
  - Time-based
  - Holiday
  - Passenger booking

SMOTE

```python
# Select 50 records from the training and validation set and checkpoint them for testing with SMOTE
trial_50 = train_and_val.limit(50).cache()

# Display the class counts before and after
row_nbrs = smote_rdd(trial_50).cache()
display(trial_50.groupBy('label').count())
display(row_nbrs.groupBy('label').count())
```

(9) Spark Jobs

trial_50: pyspark.sql.dataframe.DataFrame = [YEAR: integer, MONTH: integer ... 38 more fields]

row_nbrs: pyspark.sql.dataframe.DataFrame = [features: udt, label: double]

Table Data Profile

	label	count
1	0	41
2	1	9

Showing all 2 rows.

Table Data Profile

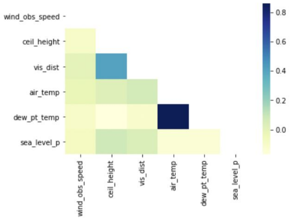
	label	count
1	1	45
2	0	41

Showing all 2 rows.

Additional Features

Weather

```python
sns.heatmap(df, fmt="g",cmap="YlGnBu",mask=np.triu(corrmatrix))
plt.show()
```



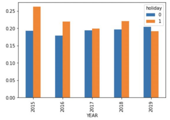
Holiday

```python
def check_holiday(month, day_of_month):
    x = 0
    if 11<= month <=12 and 17<= day_of_month <= 31:
        x=1
    elif month == 1 and day_of_month <=5:
        x=1
    return x
check_holiday = udf(check_holiday)
df = df_airlines.withColumn('holiday', check_holiday('MONTH','DAY_OF_MONTH'))\
                .select('YEAR','FL_DATE','OUTCOME','holiday')

df_pd = df.groupBy('YEAR','holiday').agg((sum('OUTCOME')/count('OUTCOME')).alias('perc_delay')).toPandas()
df_pd_new = df_pd.pivot(index="YEAR", columns="holiday", values="perc_delay")

df_pd_new.plot(kind='bar')
```

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [YEAR: integer, FL_DATE: string ... 2 more fields]

Out[65]:

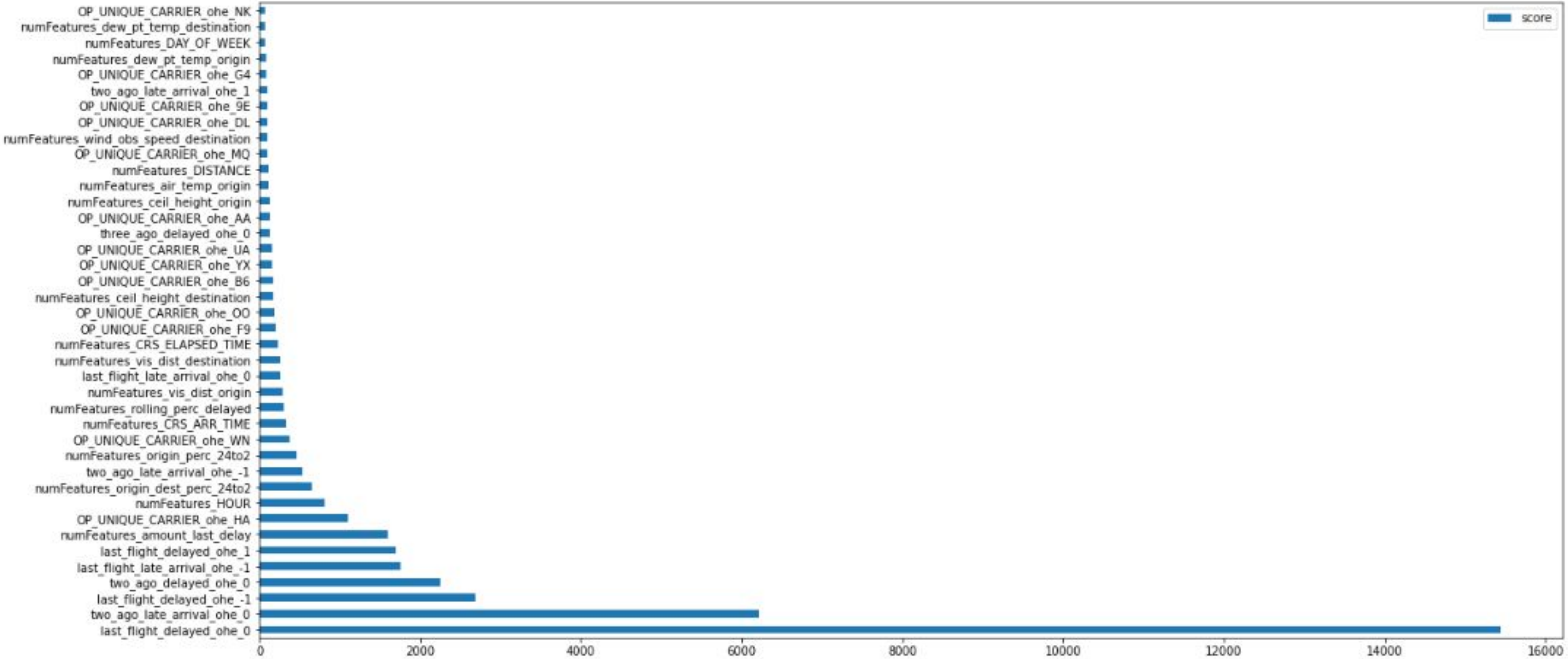


Next Steps

- New algorithms for class imbalance
- Reduce size of dataset for training
- **<u>More feature engineering</u>**
    - Weather
    - Time-based
    - Holidays
    - Passenger bookings

TAIL_NUM	ORIGIN	DEST	utc_scheduled_departure	last_flight_delayed	amount_last_delay	arrival_time_2_next_flight	prediction	label
218NV	SFB	XNA	2019-01-01T13:20:00.000+000	1	29	46920	1	1
218NV	XNA	SFB	2019-01-01T16:33:00.000+000	1	46	2700	1	1
218NV	SFB	FWA	2019-01-01T20:47:00.000+000	1	35	7020	1	1
218NV	FWA	SFB	2019-01-01T23:49:00.000+000	1	27	2700	1	1
218NV	SFB	PBG	2019-01-02T11:30:00.000+000	1	27	33900	0	1
218NV	PBG	SFB	2019-01-02T15:11:00.000+000	1	8	2700	1	1
218NV	SFB	FNT	2019-01-03T11:45:00.000+000	1	23	62880	0	1
218NV	FNT	SFB	2019-01-03T15:09:00.000+000	1	0	3000	1	0
218NV	SFB	SBN	2019-01-04T12:20:00.000+000	0	13	67140	0	0
218NV	SBN	SFB	2019-01-04T15:38:00.000+000	0	-5	3000	0	0

Appendix


score
OP_UNIQUE_CARRIER_ohe_NK
numFeatures_dew_pt_temp_destination
numFeatures_DAY_OF_WEEK
numFeatures_dew_pt_temp_origin
OP_UNIQUE_CARRIER_ohe_G4
two_ago_late_arrival_ohe_1
OP_UNIQUE_CARRIER_ohe_9E
OP_UNIQUE_CARRIER_ohe_DL
numFeatures_wind_obs_speed_destination
OP_UNIQUE_CARRIER_ohe_MQ
numFeatures_DISTANCE
numFeatures_air_temp_origin
numFeatures_ceil_height_origin
OP_UNIQUE_CARRIER_ohe_AA
three_ago_delayed_ohe_0
OP_UNIQUE_CARRIER_ohe_UA
OP_UNIQUE_CARRIER_ohe_YX
OP_UNIQUE_CARRIER_ohe_B6
numFeatures_ceil_height_destination
OP_UNIQUE_CARRIER_ohe_OO
OP_UNIQUE_CARRIER_ohe_F9
numFeatures_CRS_ELAPSED_TIME
numFeatures_vis_dist_destination
last_flight_late_arrival_ohe_0
numFeatures_vis_dist_origin
numFeatures_rolling_perc_delayed
numFeatures_CRS_ARR_TIME
OP_UNIQUE_CARRIER_ohe_WN
numFeatures_origin_perc_24to2
two_ago_late_arrival_ohe_-1
numFeatures_origin_dest_perc_24to2
numFeatures_HOUR
OP_UNIQUE_CARRIER_ohe_HA
numFeatures_amount_last_delay
last_flight_delayed_ohe_1
last_flight_late_arrival_ohe_-1
two_ago_delayed_ohe_0
last_flight_delayed_ohe_-1
two_ago_late_arrival_ohe_0
last_flight_delayed_ohe_0
0
2000
4000
6000
8000
10000
12000
14000
16000

Datasets

- Airlines Data
  (Core + Response)
- Weather Data
  (Additional Features)
- ICAO to IATA Table
  (Used For Join)
- Stations Data
  (Used For Join)

Weather Data

STATION	DATE
72530094846	2015-02-01T00:00:00.000+0000

Airlines Data

FL_DATE	ORIGIN
2015-02-01	ORD

station_id	distance_to_neighbor	neighbor_call
72530094846	0	KORD

Stations Data

ICAO	IATA
KORD	ORD

ICAO to IATA

Coverage of Weather Stations

Weather Stations
2649

Airports
372

Airports w/ Weather Station
359

Performance & Scalability

Performance

- 3 month data: 1.09 hours for crossfold validation
- Full data: xx hours for crossfold validation

Algorithm	Accuracy	Precision	Recall
Logistic Regression	0.9302	0.9472	0.7103

Datasets

- Airlines Data
  (Core + Response)
- Weather Data
  (Additional Features)
- ICAO to IATA Table
  (Used For Join)
- Stations Data
  (Used For Join)

Stations Data

station_id	neighbor_id	neighbor_call	distance_to_neighbor
72530094846	72530094846	KORD	0

station_id ICAO = neighbor_call when distance_to_neighbor = 0

```
df_stations.agg(countDistinct('station_id')).show()
```

(3) Spark Jobs

```
+-----------------+
|count(station_id)|
+-----------------+
|             2237|
+-----------------+
```

```
station_names = df_stations.select('station_id','neighbor_call','distance_to_neighbor').filter(df_stations.distance_to_neighbor==0)
station_names.agg(countDistinct('station_id')).show()
```

(3) Spark Jobs

station_names: pyspark.sql.dataframe.DataFrame = [station_id: string, neighbor_call: string ... 1 more field]

```
+-----------------+
|count(station_id)|
+-----------------+
|             2237|
+-----------------+
```

Weather Data

STATION	DATE
72530094846	2015-02-01T00:00:00.000+0000

Airlines Data

FL_DATE	ORIGIN
2015-02-01	ORD

station_id	neighbor_call
72530094846	KORD

Stations Data

ICAO	IATA
KORD	ORD

ICAO to IATA

Crossfold Validation & Evaluation Metric

Crossfold Validation

- Datasets:
  - Train: 2015-2017
  - Validation: 2018
  - Test: 2019
- ParamGrid
  - Logistic Regression: `regParam`
  - Decision Tree: `MaxDepth`

Evaluation Metrics:

Algorithm	Accuracy	Precision	Recall
Decision Tree			
Logistic Regression			

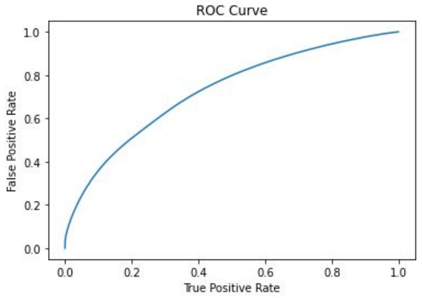
Algorithm 1 - Logistic Regression

1. Evaluation Metrics
   Recall = TP / (TP + FN)

2. Crossfold Validation
   - Current status: CrossValidator
   - Next step: Blocking Time Series Split

Evaluation Metrics

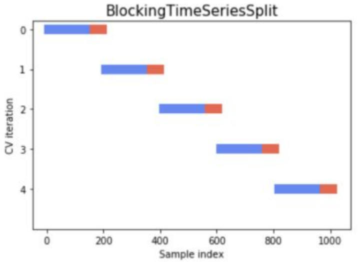
Accuracy	0.7997
Precision	0.794
Recall	0.017




```
Training set areaUnderROC: 0.7262749599356745
```

Crossfold Validation

- kFolds, k=3
- regParam, [2.0, 1.0, 0.1, 0.01]
- Next step: Blocking Time Series Cross Validation



Business Problem

- Predict delays for flights
- We’ll take a passenger-centric approach to establishing the problem space:
  - Reroutes and cancellations are equivalent to delays
  - Focus on identifying true positives

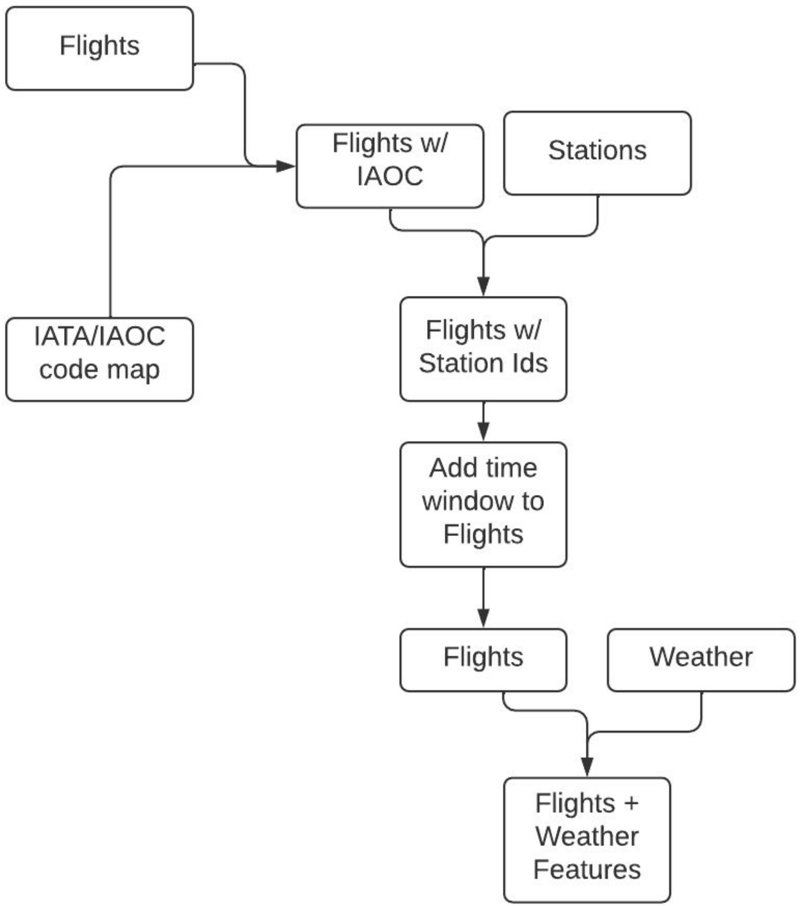
Target Variable:

$$X = \begin{cases} 0, & \text{if flight is on-time} \\ 1, & \text{if flight is delayed, rerouted, or cancelled} \end{cases}$$

Evaluation Metric:

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

Airlines + Weather Join

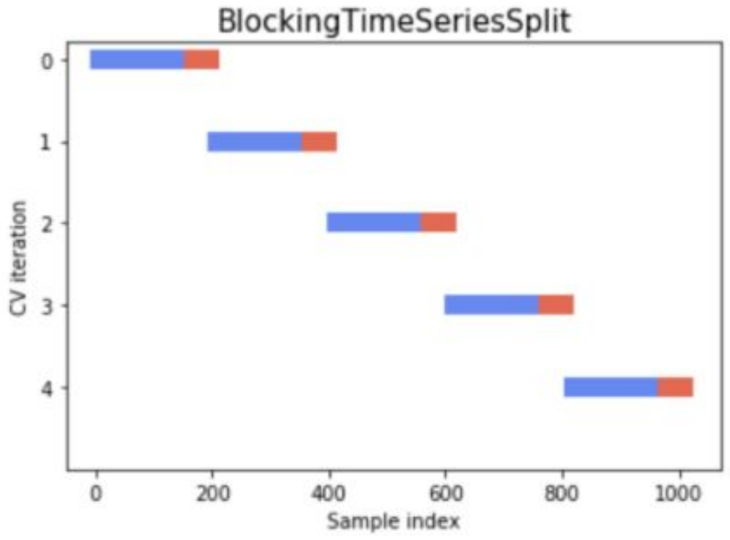


Train/Validation/Test Split

Key Principle:

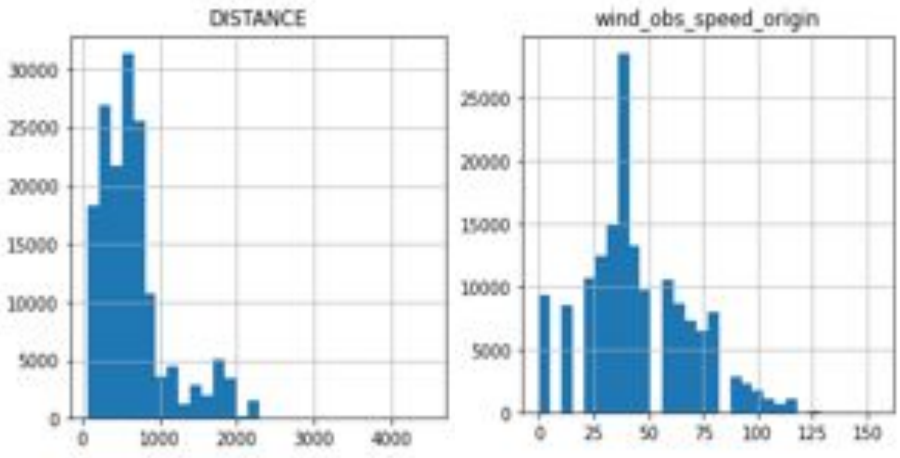
When working with time-series data, don't train on the future and test on the past

Proposed Approach:



Generating the Dataset + EDA on Joined Table

- Join
  - Flights to stations
  - Weather filtered by flights
  - Flights to weather
- 99% of airline observations retained
- Issues for modeling:
  - High correlation between some numeric features
  - Differences in scale



Data Cleaning & Feature Engineering

- Missing Data: drop rows with null, 9, 9999, 99999, 999999 and +9999
- Non-numerical features
    - One-hot encoding for logistic regression model
    - Use categorical features in original form for decision tree approach
- Features used:

- `DAY OF WEEK`
- `OP_UNIQUE_CARRIER` : one-hot encoding
- `ORIGIN` and `DEST` : one-hot encoding
- `CRS_DEP_TIME` : extract hours as a new column
- `WND` : it is parsed and derived into `wind_obs_type` and `wind_obs_speed`
- `CIG` : the first element `ceil_height` was selected
- `VIS` : the first element `vis_dist` was selected
- `TMP` : the first element `air_temp` was selected
- `DEW` : the first element `dew_pt_temp` was selected