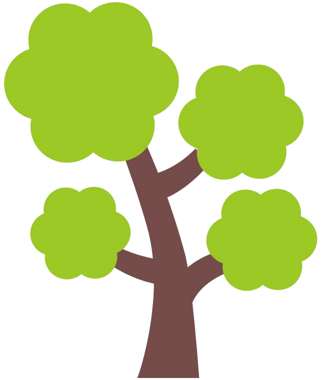
A Tree-based Prediction of Air Traffic Delays

By: Alice Hua - Harvi Singh - Omar Kapur - Preethi Raju



W261 Final Project - Spring 2020
Team 15 - Section 4
Final Presentation

AGENDA

1. Problem Statement & Business Case
2. Datasets
3. EDA & Data Transformation & Data Joins
4. Feature Engineering
5. Algorithms - Comparison & Choice & Toy Example
6. ML Pipeline & Cross Validation
7. Evaluation Metrics
8. Final Results
9. Limitations & Challenges
10. Q&A

Problem Statement & Business Case

<u>Problem</u>: Given weather and scheduled flight data, **will a flight be at least 15 minutes delayed**?

<u>Requirement:</u> Predictions need to be made **2 hours before** scheduled flight departure using only information available up to that point in time

<u>Business case:</u> our target customer is the **airlines**. They are interested in a model that captures more total delays in order to **better prepare** for delays and **improve service** to their customers while **saving on costs** and **improve decision making**.

Model Implications: Err on the side of FP when a trade off is required

DATASETS

Airline Dataset

2015 - 2019 Passenger flights TranStats DOT

	top 15 columns	null percent	bottom 15 columns	null percent
0	DIV5_TAIL_NUM	100.0 %	FLIGHTS	0.0 %
1	DIV4_TAIL_NUM	100.0 %	CRS_ELAPSED_TIME	0.0 %
2	DIV3_WHEELS_OFF	100.0 %	DIVERTED	0.0 %
3	DIV3_TAIL_NUM	100.0 %	CANCELLED	0.0 %
4	DIV4_AIRPORT	100.0 %	ARR_TIME_BLK	0.0 %
5	DIV4_AIRPORT_ID	100.0 %	CRS_ARR_TIME	0.0 %
6	DIV4_AIRPORT_SEQ_ID	100.0 %	DEP_TIME_BLK	0.0 %
7	DIV4_TOTAL_GTIME	100.0 %	CRS_DEP_TIME	0.0 %
8	DIV4_LONGEST_GTIME	100.0 %	DEST_WAC	0.0 %
9	DIV4_WHEELS_OFF	100.0 %	DEST_STATE_NM	0.0 %
10	DIV4_WHEELS_ON	100.0 %	DEST_STATE_FIPS	0.0 %
11	DIV5_AIRPORT_ID	100.0 %	DEST_STATE_ABR	0.0 %
12	DIV5_AIRPORT_SEQ_ID	100.0 %	DEST_CITY_NAME	0.0 %
13	DIV5_WHEELS_ON	100.0 %	DEST	0.0 %
14	DIV5_TOTAL_GTIME	100.0 %	DISTANCE	0.0 %

Weather Dataset

2015 - 2019 NOAA Integrated Surface Database

	top 15 columns	null percent	bottom 15 columns	null percent
0	GL1	100.0 %	SOURCE	13.3 %
1	GD5	100.0 %	NAME	0.8 %
2	MW6	100.0 %	STATION	0.6 %
3	GG4	100.0 %	DEW	0.0 %
4	MV2	100.0 %	TMP	0.0 %
5	AL3	100.0 %	VIS	0.0 %
6	AW7	100.0 %	CIG	0.0 %
7	GG3	100.0 %	WND	0.0 %
8	AW6	100.0 %	QUALITY_CONTROL	0.0 %
9	GG2	100.0 %	SLP	0.0 %
10	AX6	100.0 %	CALL_SIGN	0.0 %
11	AU5	100.0 %	REPORT_TYPE	0.0 %
12	HL1	100.0 %	ELEVATION	0.0 %
13	UG2	100.0 %	LONGITUDE	0.0 %
14	MW5	100.0 %	LATITUDE	0.0 %

Data Transformation

Airline Dataset ~ 1.2GB

Get Lat, Long

- Mapped Lat, Long to each ORIGIN & DEST per flight using a helper table

UTC Time Conversion

- Map Airport ORIGIN & DEST local times to UTC using a helper table

Dropped flights

- Dropped Diverted & Cancelled flights

Weather Dataset ~ 25GB

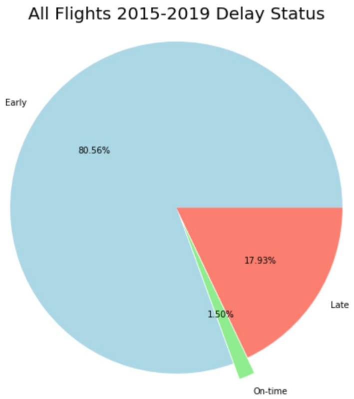
Parsing weather columns

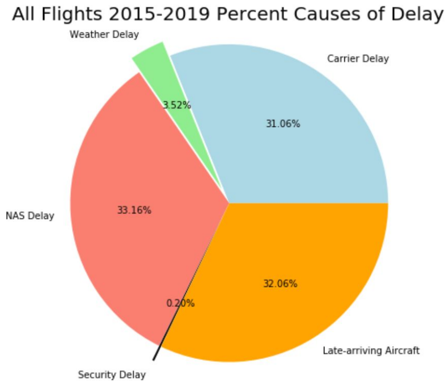
- Wind, SLP, Air Temp, Dew Point, Visibility, Ceiling all needed to be parsed into multiple variables
- Use quality codes to filter out suspect or erroneous records (denoted by 7 or 3)
- Filled 9999* values as None for missing data

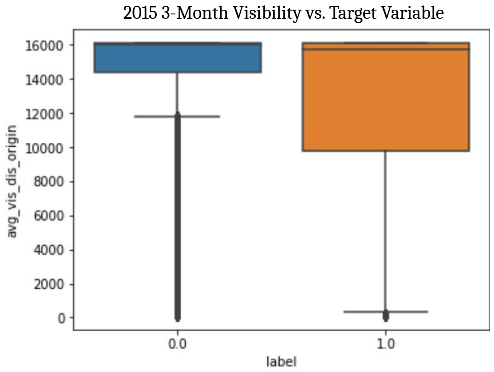
Aggregated weather data/hour

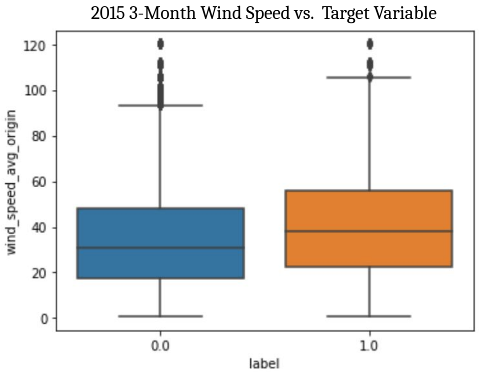
- Threw out duplicated report types like SOD (Summary of Day), SOM (Summary of Month & CRN05 (Report 5-min interval)
- GroupBy date, hour, lat, long -> take avg
- Windowed averaged over 7 days for the hour that each variable that still have nulls.

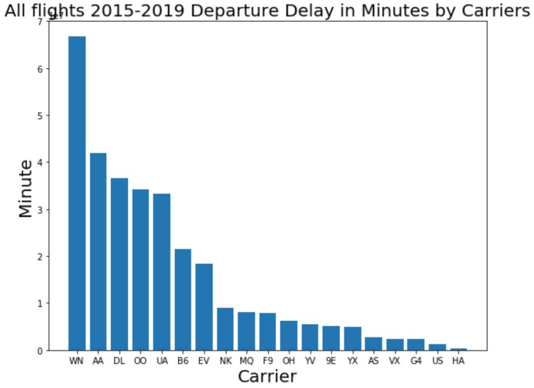
EDA

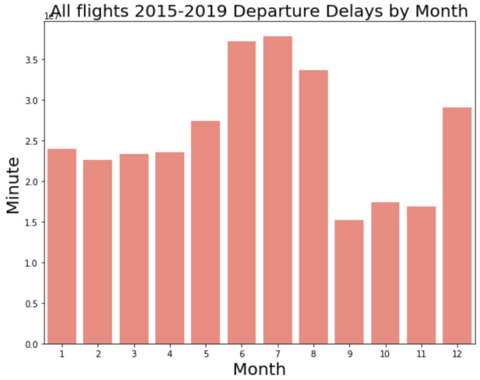

All Flights 2015-2019 Delay Status
Early
80.56%
17.93%
Late
1.50%
On-time

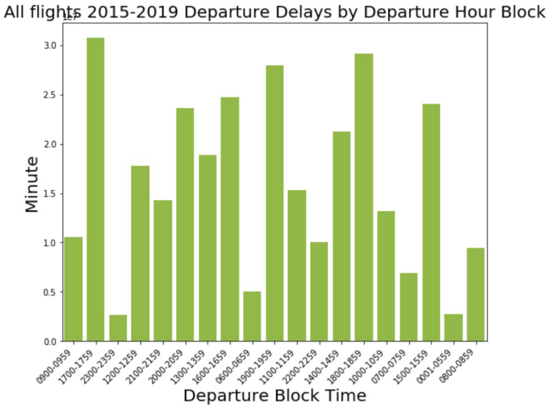

All Flights 2015-2019 Percent Causes of Delay
Weather Delay
Carrier Delay
3.52%
31.06%
NAS Delay
33.16%
32.06%
0.20%
Late-arriving Aircraft
Security Delay


2015 3-Month Visibility vs. Target Variable
avg_vis_dis_origin
16000
14000
12000
10000
8000
6000
4000
2000
0
0.0
1.0
label

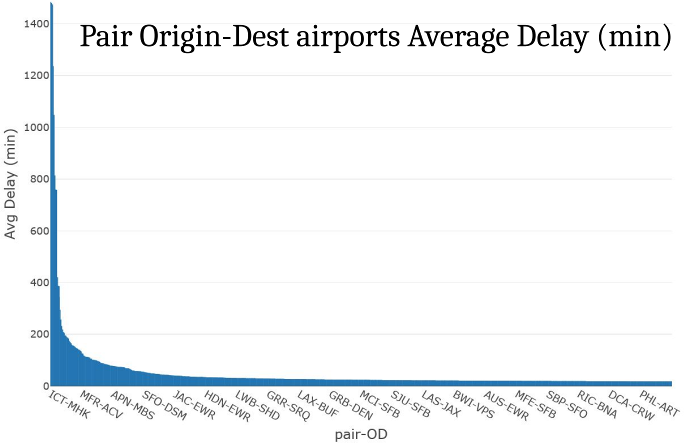

2015 3-Month Wind Speed vs. Target Variable
wind_speed_avg_origin
120
100
80
60
40
20
0
0.0
1.0
label


All flights 2015-2019 Departure Delay in Minutes by Carriers
Minute
Carrier
WN AA DL OO UA B6 EV NK MQ F9 OH YV 9E YX AS VX G4 US HA

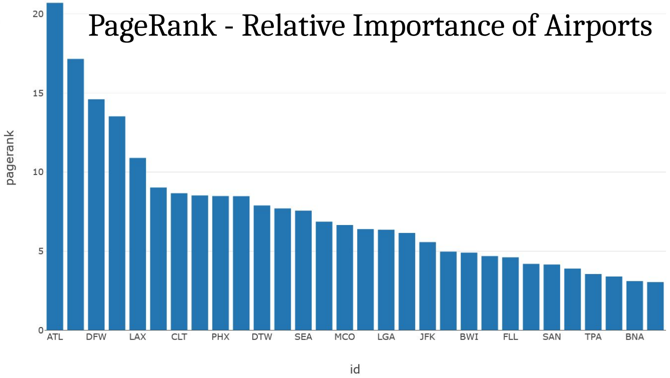

All flights 2015-2019 Departure Delays by Month
Minute
Month
1 2 3 4 5 6 7 8 9 10 11 12


All flights 2015-2019 Departure Delays by Departure Hour Block
Minute
Departure Block Time

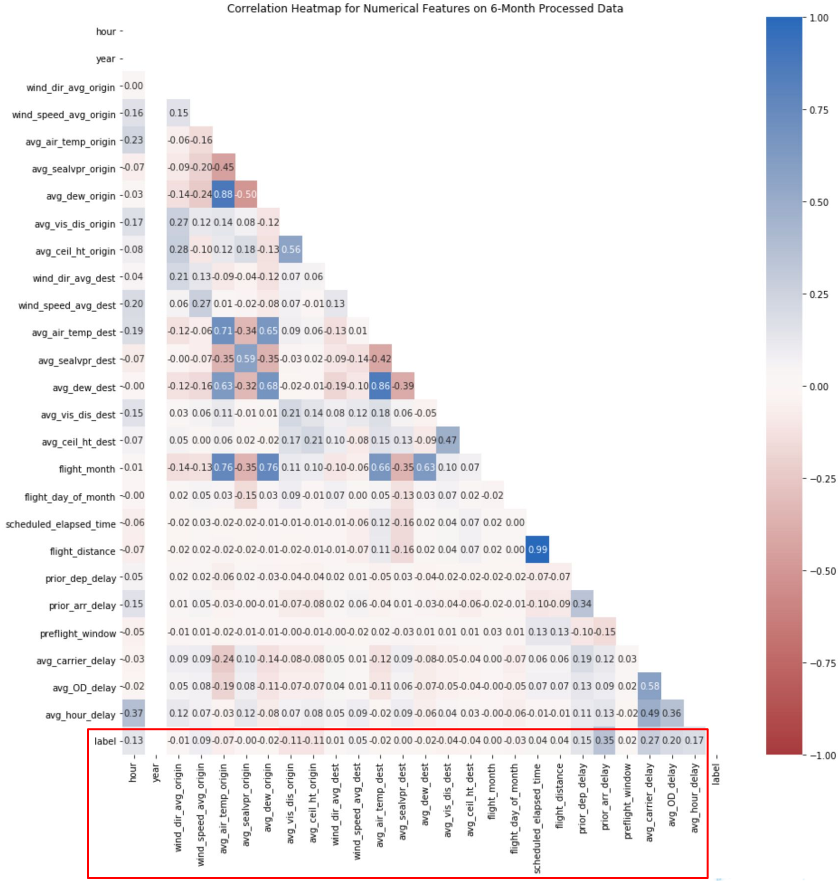
EDA cont…



Pair Origin-Dest airports Average Delay (min)
Avg Delay (min)
pair-OD
ICT-MHK
MFR-ACV
APN-MBS
SFO-DSM
JAC-EWR
HDN-EWR
LWB-SHD
GRR-SRQ
LAX-BUF
GRB-DEN
MCI-SFB
SJU-SFB
LAS-JAX
BWI-VPS
AUS-EWR
MFE-SFB
SBP-SFO
RIC-BNA
DCA-CRW
PHL-ART



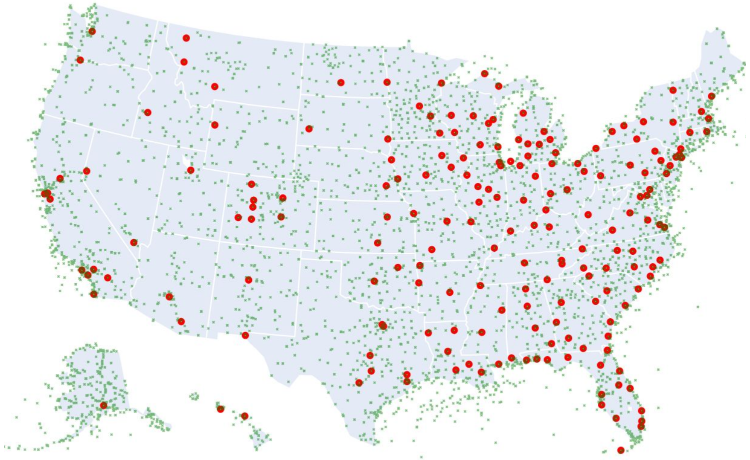
PageRank - Relative Importance of Airports
pagerank
id
ATL
DFW
LAX
CLT
PHX
DTW
SEA
MCO
LGA
JFK
BWI
FLL
SAN
TPA
BNA



Correlation Heatmap for Numerical Features on 6-Month Processed Data
hour
year
wind_dir_avg_origin
wind_speed_avg_origin
avg_air_temp_origin
avg_sealvpr_origin
avg_dew_origin
avg_vis_dis_origin
avg_ceil_ht_origin
wind_dir_avg_dest
wind_speed_avg_dest
avg_air_temp_dest
avg_sealvpr_dest
avg_dew_dest
avg_vis_dis_dest
avg_ceil_ht_dest
flight_month
flight_day_of_month
scheduled_elapsed_time
flight_distance
prior_dep_delay
prior_arr_delay
preflight_window
avg_carrier_delay
avg_OD_delay
avg_hour_delay
label

Optimizing the Airline & Weather Join

Airports and Weather stations mapped



Theta join ~2.5 hours on 3-month data

```
select [variables]
from airline_data a
left join weather_data w_origin on
    abs(w_origin.latitude - a.latitude_origin) < tolerance and
    abs(w_origin.longitude - a.longitude_origin) < tolerance and
    difference(a.flight_time - w_origin.weather_time) is between 2 and 3 hours (or some window)

(repeat join again on weather data for destination airports)
```

VS

Bucketed Join ~5-10 minutes on 3-month data

```
select [variables]
from prepred_airline_data a
left join prepared_weather_data w_origin on
  w_origin.latitude_bucket = a.origin_latitude_bucket and
  w_origin.longitude_bucket = a.origin_longitude_bucket and
  w_origin.report_hour = a.flight_hour_minus_2hrs and
  w_origin.report_date = a.flight_date_minus_2hrs

(repeat join again on weather data for destination airports)
```

(0.8% of flights without weather data)

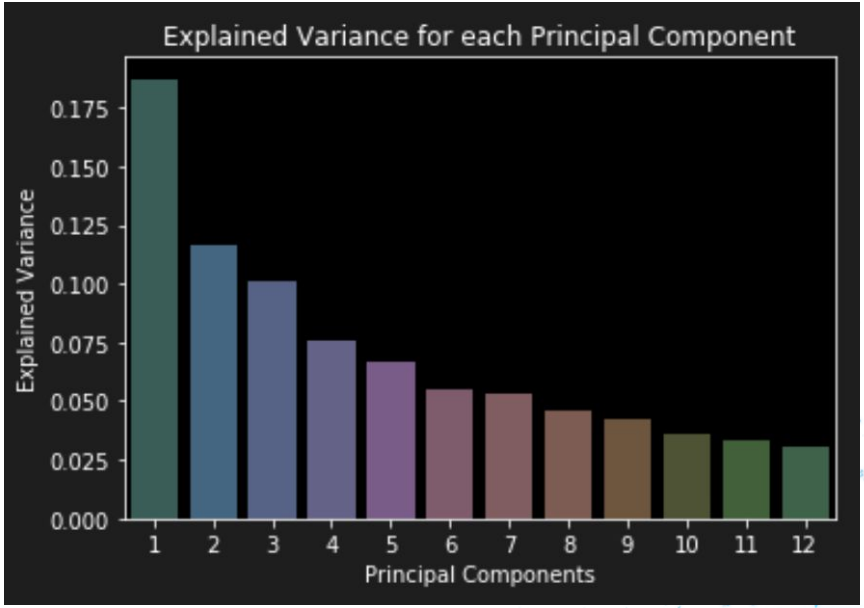
Exploring Dimensionality Reduction

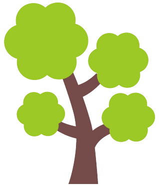
Using 24 continuous variables (including MTE for categorical)

PCA results:
12 principal components to explain over 80% of the variance

Determination: not worth using PCA-derived features

1. We lose explainability
2. The amount of continuous feature reduction is not significant





Feature Engineering

1. Preflight Window
Delta = Scheduled Departure Time (CRS)- Previous Arrival Time (CRS)
12hrs < Delta > 2hrs

2. Prior Arrival Delay

1: ARR_DELAY > 0 (late)
0: ARR_DELAY < 0 (early)
-1: Preflight window < 0 (don't know)

3. Prior Departure Delay

1: DEP_DEL15 == 1 (late)
0: DEL_DEL15 == 0 (early)
-1: Preflight window < 0 (don't know)

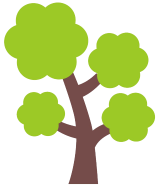
```
  windowSpec_dep = Window.partitionBy('TAIL_NUM').orderBy('scheduled_dep_time_UTC')

  airlines_tz_both = airlines_tz_both.withColumn('previous_arr_time_UTC', f.lag('scheduled_arr_time_UTC',1, None).over(windowSpec_arr))\
                      .withColumn('preflight_window', (f.unix_timestamp('scheduled_dep_time_UTC') -
f.unix_timestamp('previous_arr_time_UTC'))/60)\
                      .withColumn('can_arr_delay', f.when((f.col('preflight_window') < 43200/60 ) & (f.col('preflight_window') > 7200/60),
'yes').otherwise('no'))\
                      .withColumn('prior_arr_delay', f.when((f.col('can_arr_delay') == 'yes') & (f.col('ARR_DELAY') > 0), 1).otherwise(0))\
                      .withColumn('prior_arr_delay', f.when((f.col('preflight_window') < 0), -1).otherwise(f.col('prior_arr_delay')))\
                      .withColumn('previous_flight_delay_status', f.lag('DEP_DEL15', 1, None).over(windowSpec_dep))\
                      .withColumn('prior_dep_delayed', f.when((f.col('can_arr_delay') == 'yes') & (f.col('previous_flight_delay_status') == 1),
1).otherwise(0))\
                      .withColumn('prior_dep_delayed',  f.when((f.col('preflight_window') < 0), -1).otherwise(f.col('prior_dep_delayed')))\
```

More Feature Engineering

Additional 6 Features:

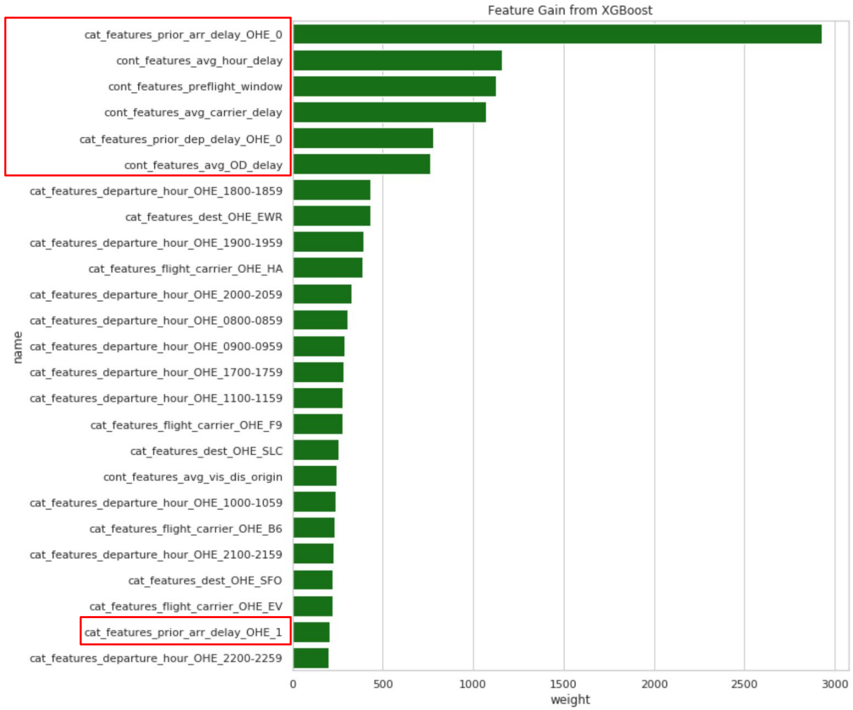
1. Number of Flights at Origin
2. Number of Flights at Destination
3. Origin-Destination Pair
4. Prior Day Hourly Average Delay
5. Prior Day Carrier Average Delay
6. Prior Day Origin-Destination Average Delay

```
windowSpec_car = Window.partitionBy('OP_CARRIER')\
                              .orderBy(f.unix_timestamp('scheduled_dep_time_UTC'))\
                              .rangeBetween(-93600, -7200)
airlines_carrier = airlines_fl_day.withColumn('avg_carrier_delay', f.round(f.avg('DEP_DEL15').over(windowSpec_car),2))

airlines_carrier = airlines_carrier.withColumn('OD_Pair', f.concat(f.col('ORIGIN'), f.lit('-'), f.col('DEST')))
windowSpec_od = Window.partitionBy('OD_Pair')\
                              .orderBy(f.unix_timestamp('scheduled_dep_time_UTC'))\
                              .rangeBetween(-93600, -7200)
airlines_OD = airlines_carrier.withColumn('avg_OD_delay', f.round(f.avg('DEP_DEL15').over(windowSpec_od),2))

windowSpec_hr = Window.partitionBy('hour')\
                              .orderBy(f.unix_timestamp('scheduled_dep_time_UTC'))\
                              .rangeBetween(-93600, -7200)
airlines_hr = airlines_OD.withColumn('avg_hour_delay', f.round(f.avg('DEP_DEL15').over(windowSpec_hr),2))
```

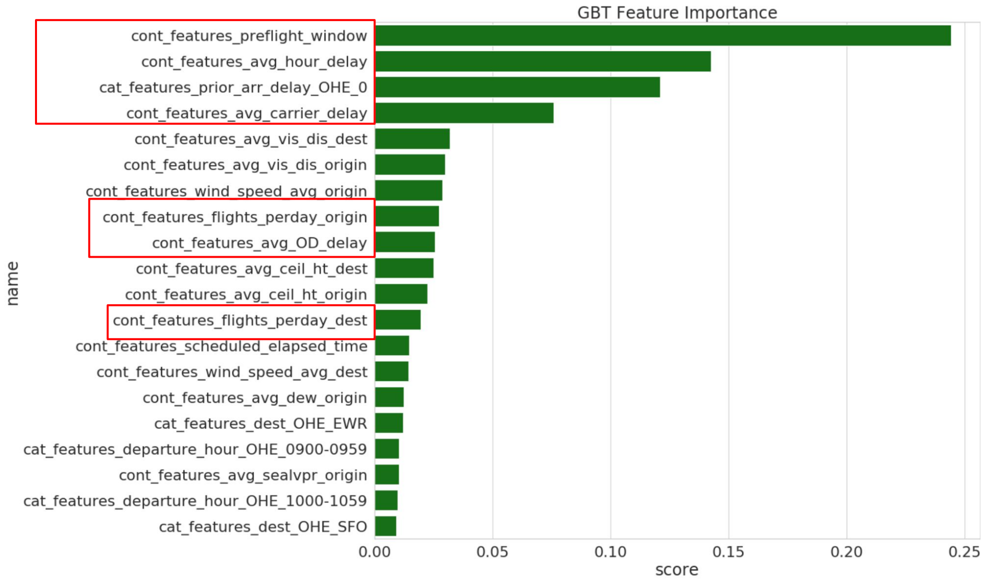


Feature Selection

Number of Features
339
Out[112]:



Out[50]:



Model Comparison - GBT vs XGBoost

"Boosting" Tree Classifiers

- Weak Learner → Strong Learner
- Each tree is fit using previous residuals
- Sequential processing - can be slow without early stopping or tuning
- Hyperparameters: max depth, iterations, learning rate

GBT

- Tree splits evaluated on entropy/gini
- Minimum impurity threshold
- Built-in, flexible threshold
- Warm Start

XGBoost

- Second Order Taylor Approximation
- Unique tree type (similarity score, gain)
- Regularization and tree pruning
- It's fast!

Toy Example

Create tree → Regularize and prune → Update probability → Minimize Loss

prior_arr_delay	label
0	0
0	0
1	1
0	0
0	0

Gamma: 1
Lambda: 0.1
Base Score: 0.5
Learning Rate: 0.3

Gain: 3

Prior arrival delay > 0

Average hr delay > 1

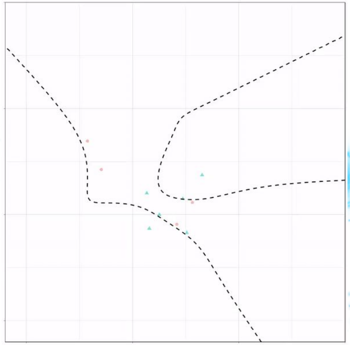
Wind Speed > 20

Gain: 2

No further split after subtracting gamma

Output: 2

...



$$log(odds)_{new} = log(odds)_{old} + learning\_rate * prediction\_residual$$

Overview of ML Pipeline Strategy

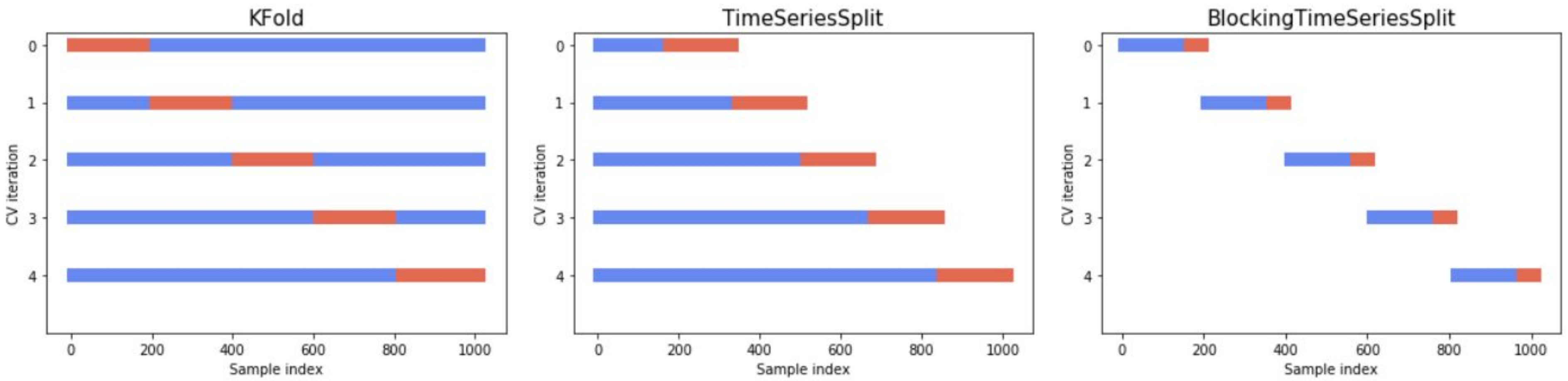
01	**Deal with Class Imbalance** (>80% on-time flights)	• Class Weights • **Down Sampling** (4x faster than CWs) • Up Sampling
02	**Pipeline to feed features to model and fit model** (Bucketizer/Indexer/OHE/ChiSqSel/Standard Scaler/Vector Assembler)	• Naive Bayes<LogReg<SVM, **GBT, XGBoost,** voting ensembles • **OHE** vs MTE for Categorical Features • Feature Selection using Chi Squared
03	**Tune Hyper-parameters** using Grid-Search while respecting the chronological time-series data	• Train-Validation-Test split vs **k-fold** • Test **Time Series Split** vs Blocking Time Series Split

ML Pipeline

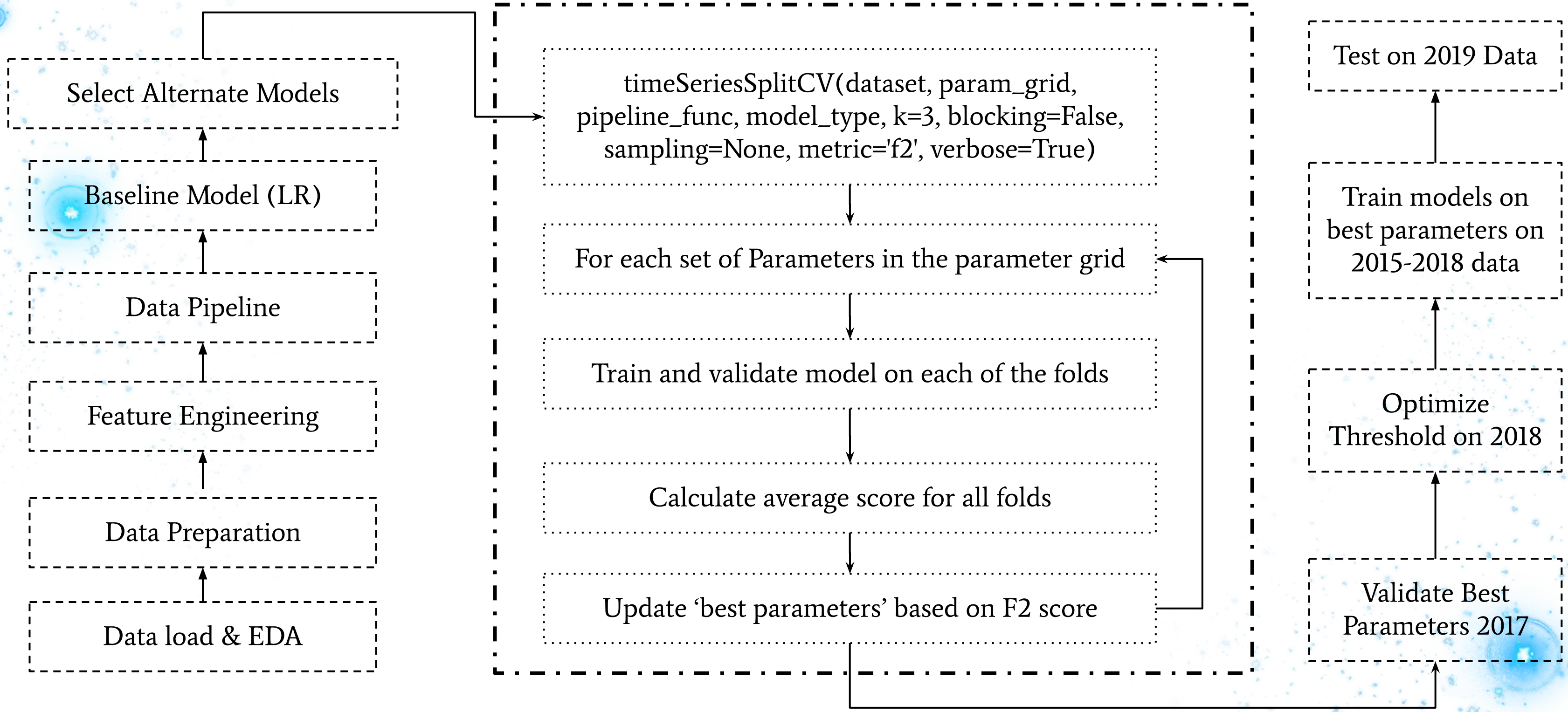
Bucketizer	Bucketized the wind direction
Indexer + OHE	Index 8 categorical variables and one hot encoded
Chi Squared	Categorical features from over 1000 to under 170
Standard Scaler	For continuous variables
Vector Assembler	Assembled features into a vector

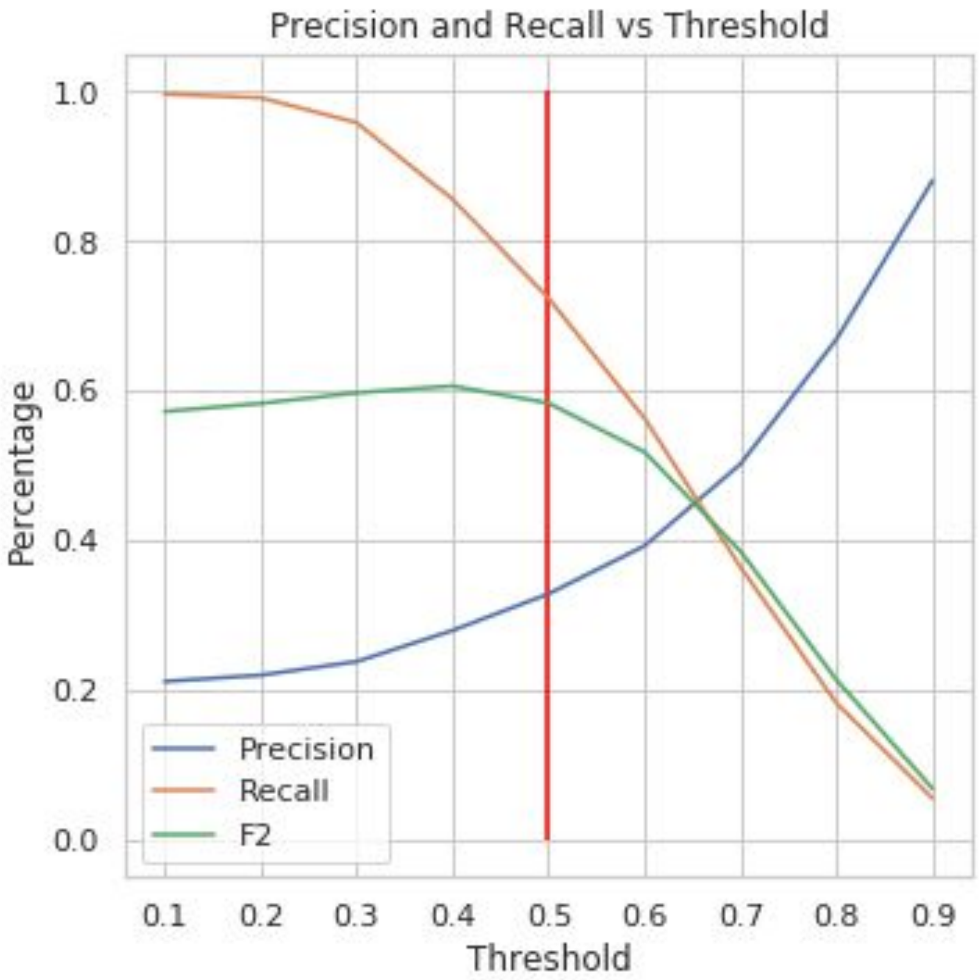
Cross Validation and Grid Search

- Why not Random sampling ?
- Split based on time, maintaining chronological order to avoid the look-ahead bias.
- Hyper-parameter tuning - 2018 data for training
- Final validation done on 2017 data-set



Grid Search & Overview of Training Process





Why Threshold Matters?

Validation Metrics

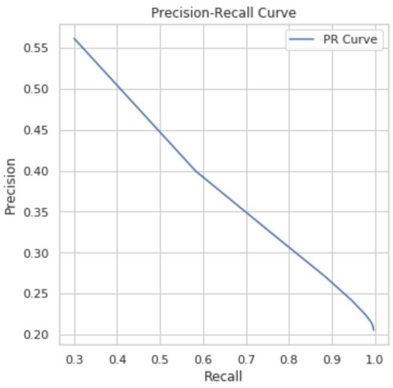
We chose...

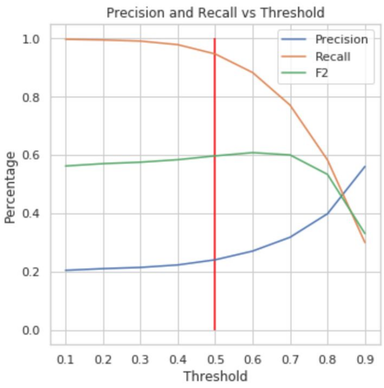
$$F_\beta = (1+\beta^2)\frac{precision * recall}{\beta^2 precision + recall}$$

$$F2 = (1+2^2)\frac{precision * recall}{2^2 precision + recall}$$

where...

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$





- ➔ **F2:** A metric that combines Precision + Recall to measure the model's performance

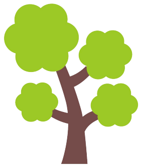
  Gives less weight to Precision & more weight to Recall

  Minimizing FN > minimizing FP
- ➔ **Precision**: Out of those predicted Positives, how many are actually Positives
- ➔ **Recall**: How many of True Positives our model captures, through labeling it as Positives
- ➔ We want Precision >= 30% while maximizing F2 because it's more important to classify correctly as many Positives (Delays) as possible
- ➔ We used **AUC PR** to gauge for **Decision Threshold**

Cross Validation - Grid Search Summary

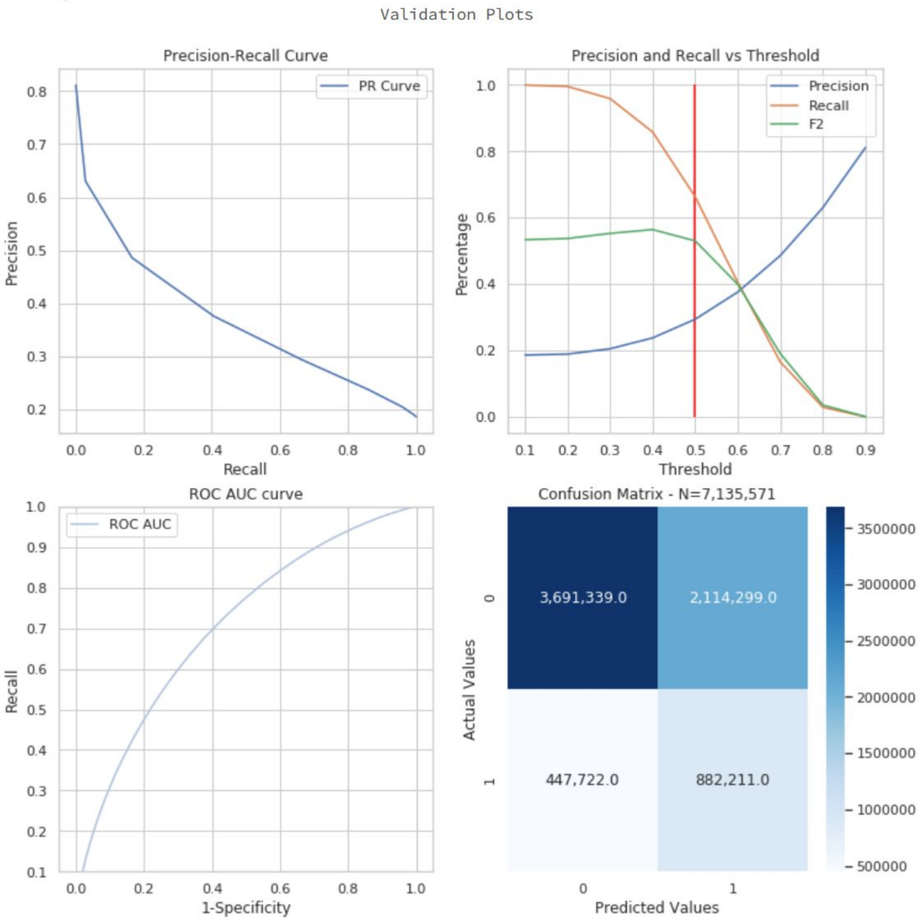
We killed some trees along the way.....

Model	CV Params (Tuned on 2018) 3 rounds where k=2 each round	CV Scores (Validate on 2017) 3 rounds	CV Runtime (minutes)
GBT	maxDepth: [2, 4, 5, 6, 8, **9**] maxBins:[**280**, 325, 350, 400, 430] maxIter: [4, **6**, 7] stepSize: [0.1, 0.13, **0.2**]	PR AUC: 0.41 - 0.46 ROC AUC: 0.75 - 0.77 **F2: 0.56 - 0.57** **Recall: 0.69 - 0.70** **Precision: 0.31 - 0.33** Accuracy: 0.67 - 0.69	4 + 2 + 7 = **13 hrs**
XGBoost	max_depth: [3, 5, **6**, 8] n_estimators: [100, 120, **150**] reg_lambda: [0, **1**] reg_alpha: [**0**, 1] objective: [**binary:logistic**] learning_rate': [0.15, **0.2**, 0.3, 0.6] gamma: [**0.05**, 0.8] scale_pos_weight: [**2**, 3] min_child_weight: [1, **1.5**]	PR AUC: 0.48 - 0.49 ROC AUC: 0.79 **F2: 0.60 - 0.61** **Recall: 0.90 - 0.95** **Precision: 0.24 - 0.27** Accuracy: 0.45 - 0.54	6 + .50 + 1 = **7.5 hrs**



Global Logistic Baseline

Train on 2015 - 2018 / Test on 2019



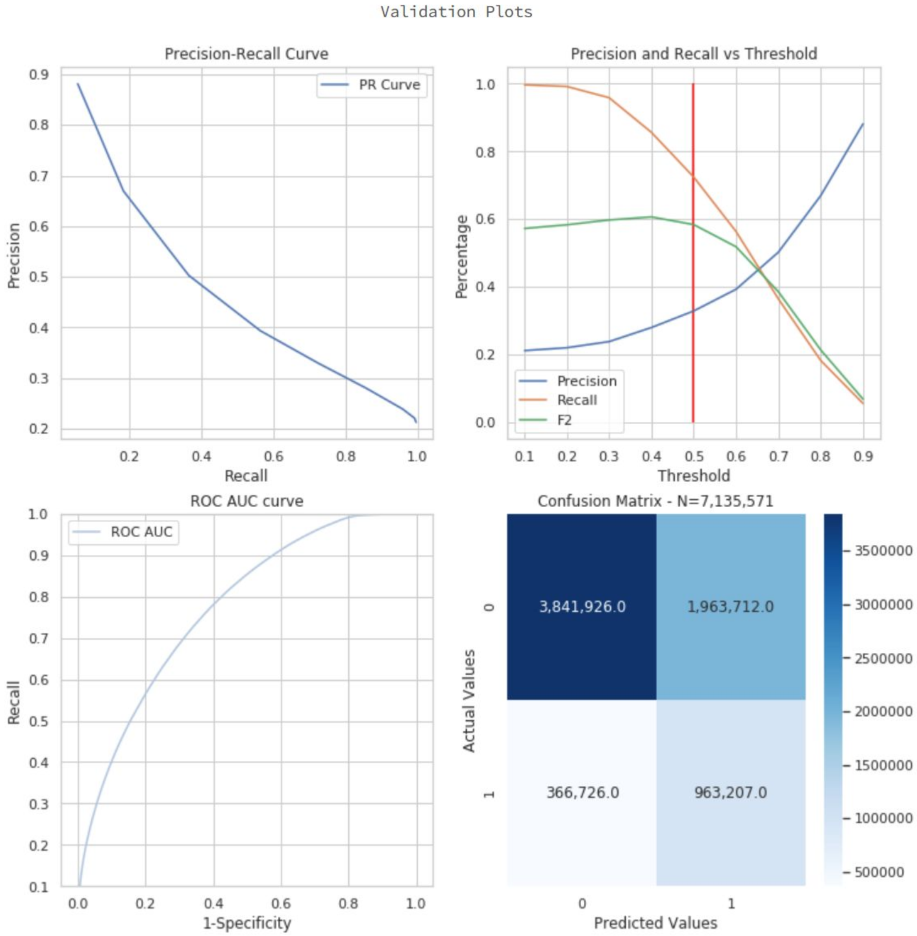

```
Evaluating model on test data using threshold of 0.5
Global Baseline Logistic Regression
```

	Train	Test
PR AUC	0.3727	0.3649
ROC AUC	0.7155	0.7051
F0.5 Score	0.3347	0.3313
F2 Score	0.5281	0.5304
Recall	0.6541	0.6633
Precision	0.2983	0.2944
Accuracy	0.6605	0.6410

Params used: {'regParam' : 0.1
'sampling: 'weights',
'threshold': 0.5}

Model - GBT

Train on 2015 - 2018 / Test on 2019



Evaluating model on test data using threshold of 0.5

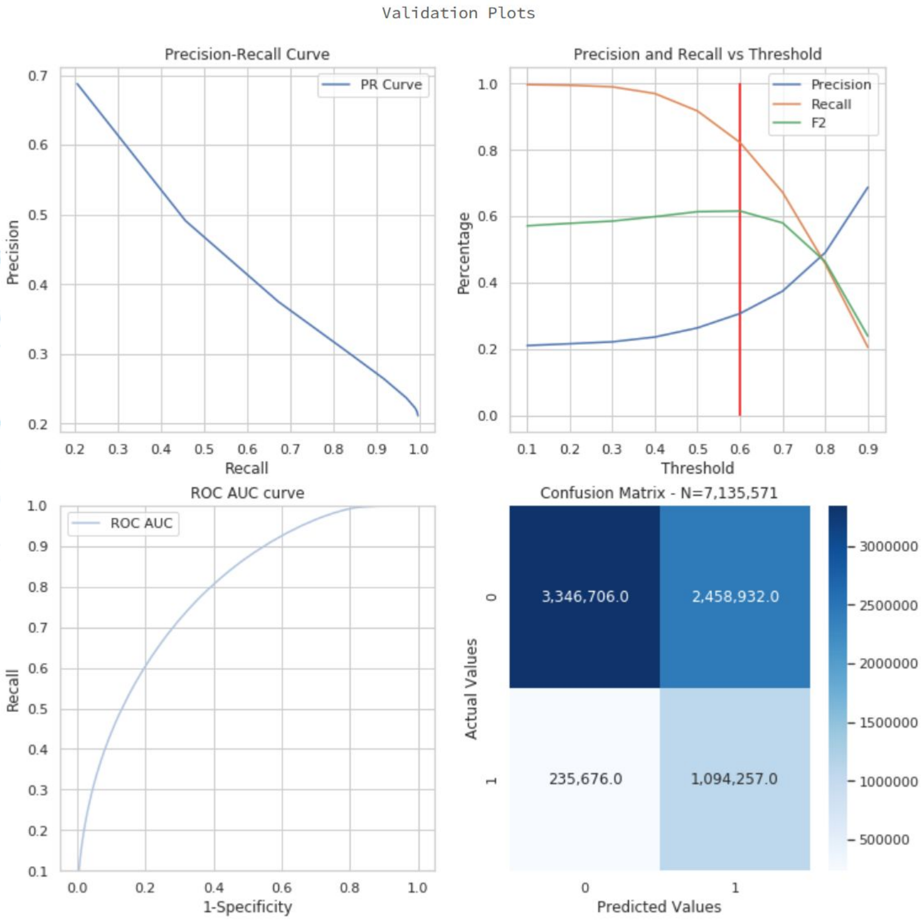
Final Gradient Boosted Tree Model

	Train	Test
PR AUC	0.4666	0.4756
ROC AUC	0.7760	0.7729
F0.5 Score	0.3705	0.3694
F2 Score	0.5768	0.5840
Recall	0.7082	0.7243
Precision	0.3310	0.3291
Accuracy	0.6896	0.6734

Params used: {'maxDepth': 9,
'maxBins': 280,
'maxIter': 6,
'stepSize': 0.2,
'sampling': 'down',
'threshold': 0.5}

Model -xgboost

Train on 2015 - 2018 / Test on 2019




```
Evaluating model on test data using threshold of 0.6
Final XGBoost Model
                Train    Test
PR AUC          0.5023   0.5061
ROC AUC         0.7966   0.7915
F0.5 Score      0.3521   0.3520
F2 Score        0.6141   0.6166
Recall          0.8166   0.8228
Precision       0.3083   0.3080
Accuracy        0.6369   0.6224
```

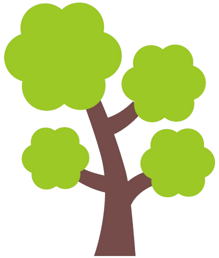
Params used: {'max_depth': 6,
'n_estimators': 150,
'reg_lambda': 1,
'reg_alpha': 0,
'objective':
'binary:logistic',
'base_score': 0.5,
'learning_rate': 0.2,
'gamma': 0.05,
'scale_pos_weight': 2,
'min_child_weight': 1.5
'sampling': 'down',
'threshold': 0.6}

Limitations

- Time for more Hyperparameter tuning over more data/more folds - (XGBoost has a lot of knobs)
- No Neural Net - (used in a State of the Art MIT paper)
- Need more creative features
- PySpark does not have SMOTE or WOE, or MTE, takes much longer to implement by ourselves + troubleshooting pipeline issues
- Leakage challenges, 2 hour requirement
- Imbalanced target classes

Takeaways

- Feature engineering is critical
- Optimizing joins is vital for big data
- For custom functions, scala would be good to learn
- Evaluation metrics should be connected to the business case



Q & A