Predicting Flight Delays & Cancellations in the United States

Section 1: Group 4
Team: House Spark

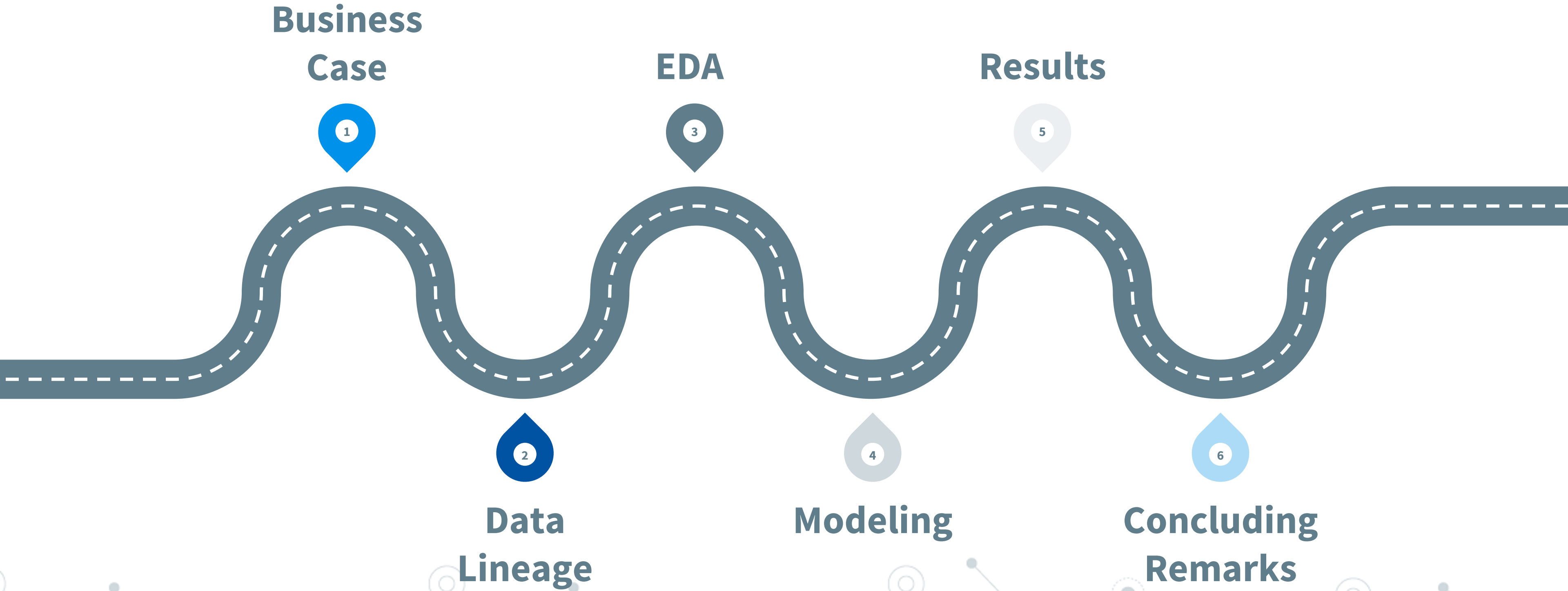
Oleg Ananyev

Briana Hart

Anuradha (Annie) Passan

Neil Prabhu

Agenda


Business Case
1
Data Lineage
2
EDA
3
Modeling
4
Results
5
Concluding Remarks
6

1.

Business Case

Business Case

Context

Flights can be delayed due to several reasons, whether mechanical, weather related, onboarding complication, etc. When they occur - they are a significant inconvenience to passengers and staff, and results in significant costs to airlines, airports, and all other involved parties.

Objective

Predict if a domestic flight in the United States will be delayed (15 + minutes) ; and if so by how many minutes.

2.

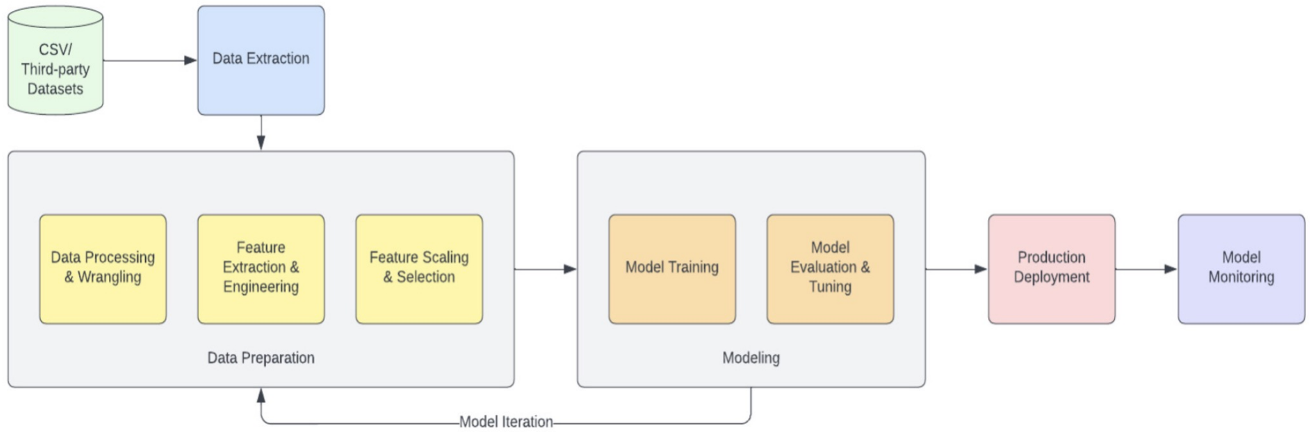
Data Lineage

Data Lineage (I): Data, Pipeline

Data

Dataset	Size	Source	Columns	Rows
Airline	2.8GB	US Department of Transportation	108	74M
Weather	33GB	National Oceanic and Atmospheric Observatory Repository	123	899M
Station	53MB	Provided by W261 Instructors	12	5M
Airport	5.6MB	Third-Party, Open Source Dataset	12	73K

Pipeline



Data Lineage (II) : Join

Join Steps

- **Airlines to Airports**, key: *IATA Code*
  - *Convert Timestamps to UTC*
  - *Create ICAO code*
- Subset Weather data on closest weather stations to airports based on 'distance_to_neighbor'
- **Airline_Airport to Station**, key: ICAO Code
- **Airline_Airport_Station to Weather (origin and destination)**, key: Station, UTC Timestamp (to the hour)
  - *Drop unnecessary columns based on EDA*

95M
Columns

42.4M
Rows

1.5 GB
Size

12 min
Write to Parquet

Data Lineage (III): Post Join

Key Cleaning

- Drop certain columns based on EDA
- Convert columns to proper data types
- Trim trailing/leading spaces
- Adjust variables to our definition of delay 15+ minutes

Engineered Features

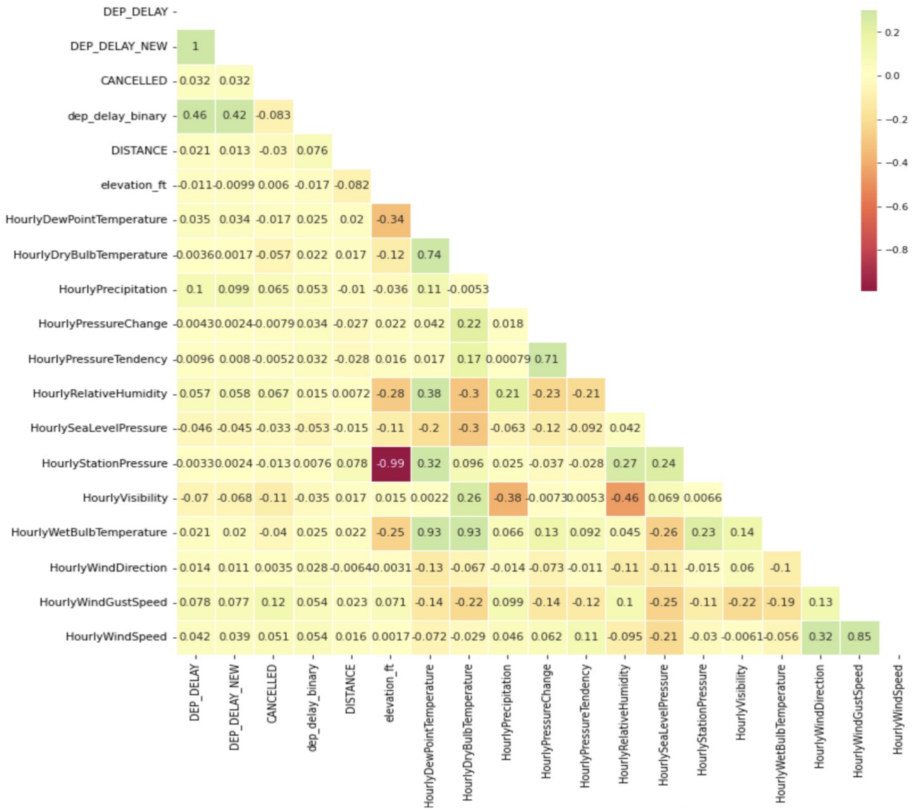
- Per tail number in past 24 hours (Model Pipeline)
  - **# Flights**
  - **# Flight delays**
  - **# Flight cancellations**
  - **% Flight delays**
  - **% Flight cancellations**
- **Year**
- **Holiday** – binary: 1 if holiday
- **Holiday_in2DayRange** – binary: if holiday +/- 2-days
- **C19** - Ordinal value on the different stages of Covid as it relates to air travel

Feature Family	Count
Time Related	8
Flight Info.	5
Aircraft History	5
Origin Info.	6
Destination Information	4
Holiday	2
Covid-19	1
Origin Weather	22
Destination Weather	22
Other	12
10 Families	**89**

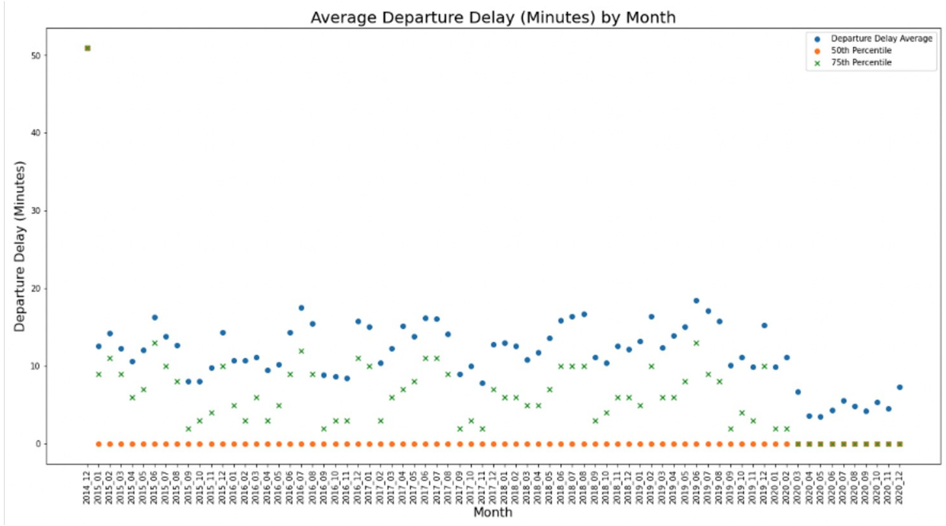
3.

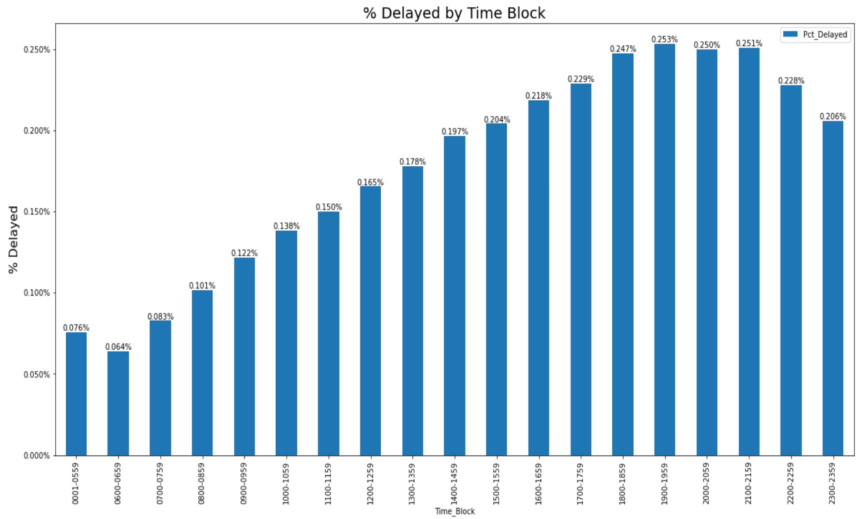
EDA

EDA (I): Correlation Matrix

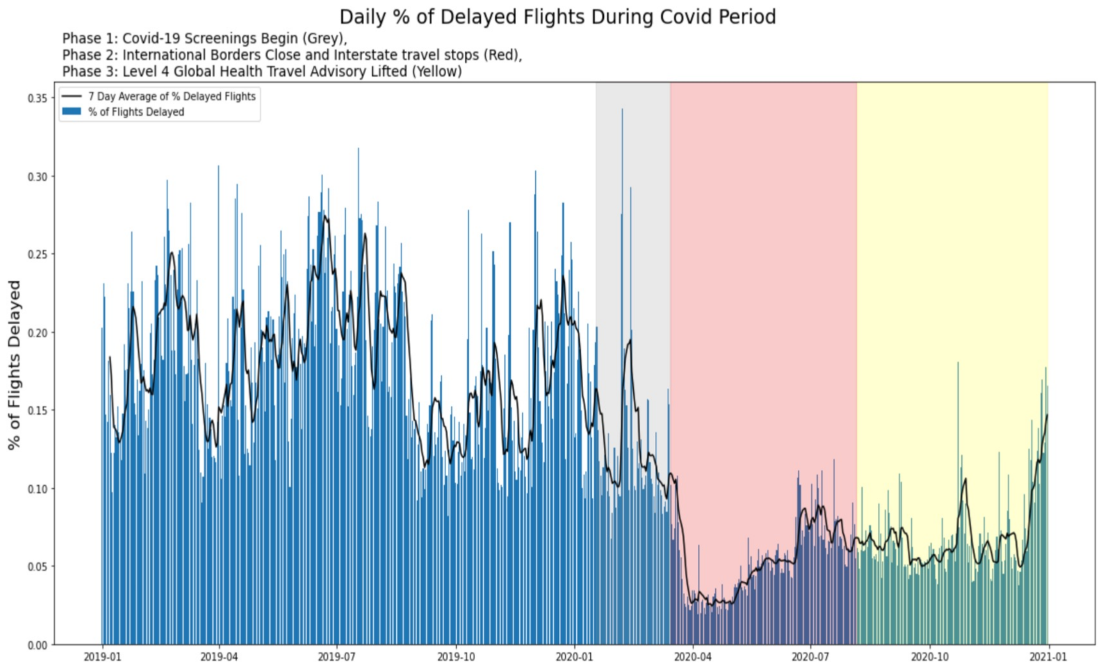


EDA (II): Flight Delays & Month, Hour





EDA (III): Flight Delays & COVID-19



4.

Modeling

Modeling (I): Algorithms and Metrics

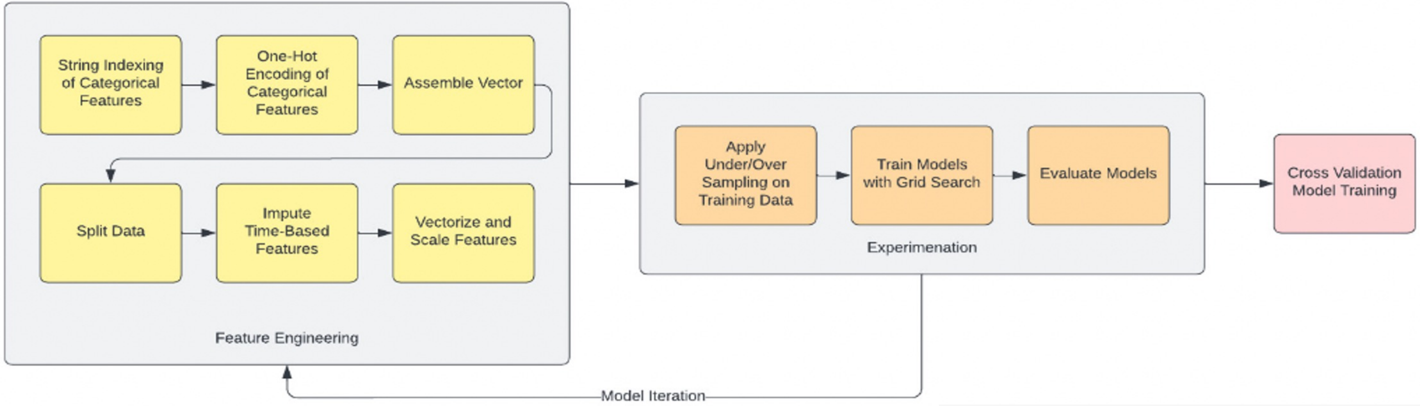
Algorithms Investigated

- Classification ~ Predict Delay Event
  - Logistic Regression
  - Decision Tree
  - Random Forest
  - Multiple Layer Perceptron
- Regression ~ Predict Delay by Minutes
  - Linear Regression
  - Decision Tree
  - Random Forest
  - Gradient Boosted Tree

Metrics Used

- Classification ~ Predict Delay Event
  - F1 Score
    - Precision
    - Recall
- Regression ~ Predict Delay by Minutes
  - Mean Absolute Value (MAE)

Modeling (II): Pipeline & Data Leakage




```
### FINAL FEATURE ADDITIONS
## GET NUMBER & PERCENTAGE OF TIMES A PLANE (BY TAIL NUMBER) HAS BEEN DELAYED OR CANCELLED IN THE PAST 3 MONTHS (2 COLUMNS)
# Make window function
df = df.withColumn('roundedMonth', f.date_trunc('month', df.scheduled_departure_UTC))
window_3m = Window().partitionBy('TAIL_NUM').orderBy(f.col('roundedMonth').cast('long')).rangeBetween(-(86400), 0) # changed to 1 day instead of 3 months

# Add in Columns
# Number of flights delayed/cancelled
df = df.withColumn('no_delays_last1d', when(df.TAIL_NUM.isNotNull(), f.sum('dep_delay_15').over(window_3m)).otherwise(-1)) \
       .withColumn('no_cancellation_last1d', when(df.TAIL_NUM.isNotNull(), f.sum('CANCELLED').over(window_3m)).otherwise(-1))
# Percentage of flights delayed/cancelled
df = df.withColumn('count_flights_last1d', when(df.TAIL_NUM.isNotNull(), f.count('TAIL_NUM').over(window_3m)).otherwise(-1))
df = df.withColumn('perc_delays_last1d', when(df.count_flights_last1d != -1, (df.no_delays_last1d/ df.count_flights_last1d)).otherwise(-1.0)) \
       .withColumn('perc_cancellation_last1d', when(df.count_flights_last1d != -1, (df.no_cancellation_last1d/ df.count_flights_last1d)).otherwise(-1.0))
```

Data Leakage Safeguards

- All cleaning prior to data split record based
- Time-based features created after data split
- Null imputations done post data split

Modeling (III): Baseline Models

Classification

- Logistic Regression
  - 10 iterations maximum
  - No regularization
- Train: 2015-2019

Train Accuracy	0.757361891
Train Precision	0.826164895
Train Recall	0.988667877
Train F1 Score	0.757361891

- Validate: 2020

Val Accuracy	0.874172948
Val Precision	0.914797405
Val Recall	0.983889928
Val F1 Score	0.874172948

Regression

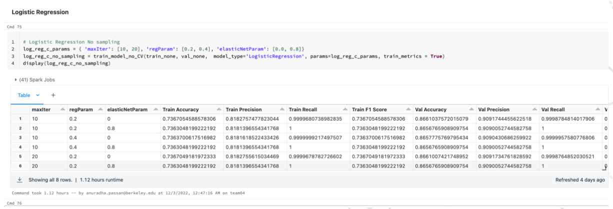
- Linear Regression
  - 10 iterations maximum
  - No regularization
- Train: 2015-2019

Train MAE	17.9826124
Train RMSE	41.4261159

- Validate: 2020

Val MAE	15.4381885
Val RMSE	35.4042738

Modeling (IV): Experimentation



Logistic Regression

Cmd 75

```
# Logistic Regression No sampling
log_reg_c_params = { 'maxIter': [10, 20], 'regParam': [0.2, 0.4], 'elasticNetParam': [0.0, 0.8]}
log_reg_c_no_sampling = train_model_no_CV(train_none, val_none,  model_type='LogisticRegression', params=log_reg_c_params, train_metrics = True)
display(log_reg_c_no_sampling)
```

(41) Spark Jobs

Table +

	maxIter	regParam	elasticNetParam	Train Accuracy	Train Precision	Train Recall	Train F1 Score	Val Accuracy	Val Precision	Val Recall	V
1	10	0.2	0	0.7367054588578306	0.8182757477823044	0.9999680738982835	0.7367054588578306	0.8661037572015079	0.9091744455622518	0.9998784814017906	0
2	10	0.2	0.8	0.7363048199222192	0.8181396554341768	1	0.7363048199222192	0.8656765908909754	0.9090052744582758	1	0
3	10	0.4	0	0.7363700617516982	0.8181618522433426	0.9999999217497507	0.7363700617516982	0.8657775769795434	0.9090430686259922	0.9999957580776806	0
4	10	0.4	0.8	0.7363048199222192	0.8181396554341768	1	0.7363048199222192	0.8656765908909754	0.9090052744582758	1	0
5	20	0.2	0	0.7367049181972333	0.8182755615034469	0.9999678782726602	0.7367049181972333	0.8661007421748952	0.9091734761828592	0.9998764852030521	0
6	20	0.2	0.8	0.7363048199222192	0.8181396554341768	1	0.7363048199222192	0.8656765908909754	0.9090052744582758	1	0

Showing all 8 rows. | 1.12 hours runtime

Refreshed 4 days ago

Command took 1.12 hours -- by anuradha.passan@berkeley.edu at 12/3/2022, 12:47:16 AM on team04

Cmd 76

Experimentation Process

0.1% → 10% → Full → CV → Selection

No. Experiments

Cross Validation	0.1%	10%	Full
No	570	168	43
Yes: K=5	-	5	-
Total: 786 Experiments			

Features

Feature Families (9)	No. Features
Time, Flight info., Aircraft history, Origin info., Dest. Info., Holiday, C19, Origin + Dest. Weather	**62**

Hyperparameters Explored

Model Type	Hyperparameters
Classification (no, over, under sampling)	Logistic: maxIter, reg_param, elasticNet
	DT: max_depth, impurity, maxBins, minInfoGain
	RF: numTrees, max_depth, impurity, maxBins, minInfoGain
	MLP: maxIter, stepSize, blockSize, tol
Regression (no sampling)	Linear: maxIter, reg_param, elasticNet
	DT: max_depth, minInfoGain
	RF: numTrees, max_depth, minInfoGain
	GBT: max_depth, maxIter, stepSize, minInfoGain

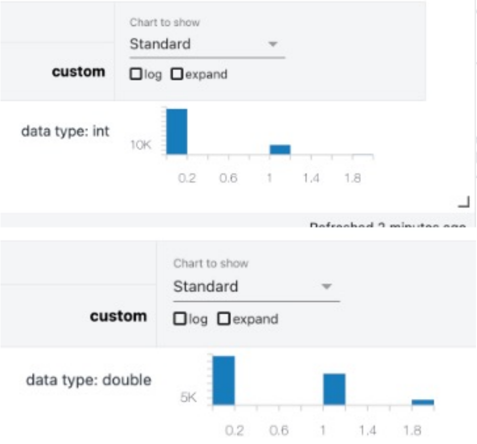
Modeling (V): Challenges

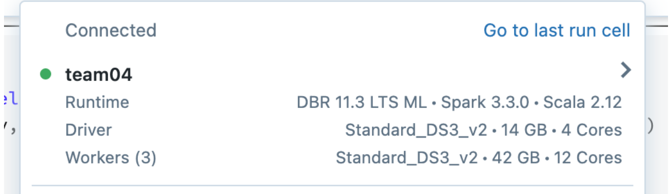
Challenge

- Stale time-based features
- Limited Computation Resources
- Over prediction of cancellations

Action Taken

- Modified: past 3 month to 24 hours
- Adjust experiment-ation with CV
- Switch to binary classification





5.

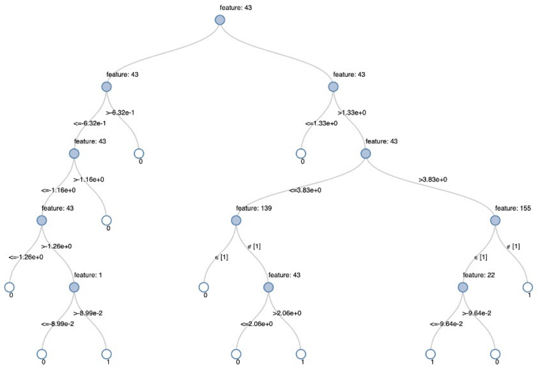
Results

Results (I): Top Performing Models

Model	Hyperparameters	Metrics
Decision Tree	Max Depth: 5 Impurity: Gini Max Bins: 32 Min Information Gain: 0.0	Accuracy: 0.7746 Precision: 0.8384 Recall: 0.9739 F1 Score: 0.7746 AUC: 0.5956
Linear Regression	Max Iterations: 20 Regularization Param: 0.2 Elastic Net Param: 0.8	MAE: 18.0776 RMSE: 45.7000

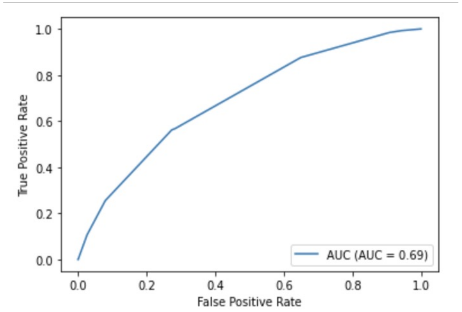
Results (II): Classification Model

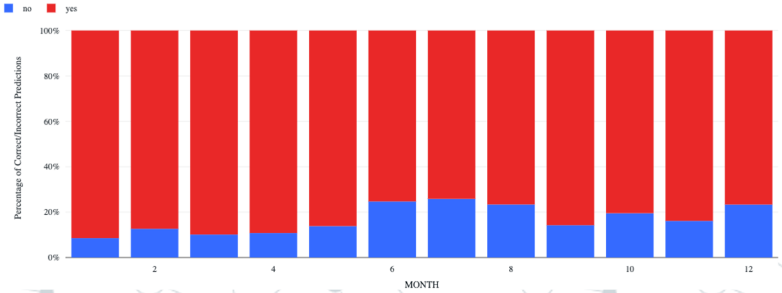
Features



	index	feature_importance	name
1	43	0.977885465939109	perc_delays_last1d
2	139	0.02137717944620855	features_cat_DEP_TIME_BLK_index_encoded_0600-0659
3	155	0.00047963389466508966	features_cat_DEP_TIME_BLK_index_encoded_0001-0559
4	22	0.00019334238801444202	dest_HourlyAltimeterSetting
5	1	0.00006437833200290147	origin_HourlyAltimeterSetting
6	0	0	features_cont_DISTANCE

Performance





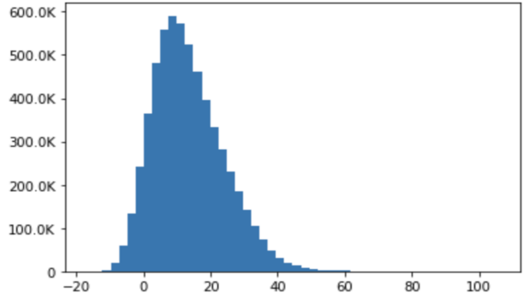
Results (III): Regression Model

Features

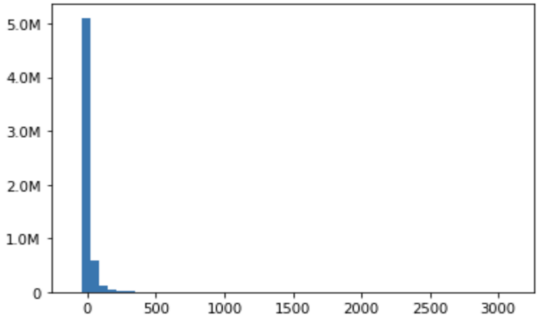
	index	0	name
1	139	-7.9145546570355325	features_cat_DEP_TIME_BLK_index_encoded_0600-0659
2	155	-6.954199201886445	features_cat_DEP_TIME_BLK_index_encoded_0001-0559
3	140	-6.623772945339058	features_cat_DEP_TIME_BLK_index_encoded_0700-0759
4	141	-5.085634888372415	features_cat_DEP_TIME_BLK_index_encoded_0800-0859
5	147	-4.129184067377121	features_cat_DEP_TIME_BLK_index_encoded_0900-0959
6	115	-2.598523667466155	features_cat_OP_UNIQUE_CARRIER_index_encoded_WN

	index	0	name
1	415	24.951917065544396	features_cat_ORIGIN_AIRPORT_ID_index_encoded_10154
2	850	21.42911476894827	features_cat_DEST_AIRPORT_ID_index_encoded_11447
3	43	9.4609760686959	perc_delays_last1d
4	562	8.860605743175402	features_cat_DEST_AIRPORT_ID_index_encoded_11618
5	483	8.122721697185955	features_cat_ORIGIN_AIRPORT_ID_index_encoded_12012
6	786	7.8253821048898935	features_cat_DEST_AIRPORT_ID_index_encoded_13459

Performance



Distribution of Model Predictions



Distribution of Labels

6.

Concluding Remarks

Final Remarks

- **Goal:** Develop classification model to predict if a flight will be delayed to feed into a regression model to predict by how many minutes a flight will be delayed. Models that incorporate information on basic flight information, aircraft history, weather conditions, holidays, and COVID-19 can predict this.
- **Top Models:**
  - Classification: Decision Tree (F1 Score: 0.7746)
  - Regression: Linear Regression (MAE: 18.0776)
- **Key Lessons learned:**
  - The need to strategize cleaning and manipulation in different parts of the data pipeline to prevent leakage
  - Should have looked more at confusion matrices during grid search
- **Moving Forward**
  - Look into implementing more sophisticated techniques for weather null imputations
  - Investigate further on how to better differentiate between delays and cancellations
  - Implement pipeline where the classification model feeds into the regression model (working with 10% data)
  - Further investigate more sophisticated sampling techniques (follow up on SMOTE)

Thank You

Appendix

Modeling: Experiment Results (no CV)

Model	Hyperparameters	Metrics
Logistic Regression	Max Iterations: 10 Regularization Param: 0.2 Elastic Net Param: 0.0	F1 Score: 0.8661 Precision: 0.9091 Recall: 0.9998 Accuracy: 0.8661
Decision Tree Classification	Max Depth: 5 Impurity: Gini Max Bins: 32 Min Info. Gain: 0.0	F1 Score: 0.8656 Precision: 0.9090 Recall: 1 Accuracy: 0.8656
Random Forest Classification	Number of Trees: 3 Max Depth: 15 Impurity: Gini Max Bins: 32 Min Info. Gain: 0.01	F1 Score: 0.8656 Precision: 0.9090 Recall: 1 Accuracy: 0.8656
Multiple Layer Perceptron (5 - Sigmoid - 4 Sigmoid - 2 Softmax)	Max Iterations: 20 Step Size: 0.03 Block Size: 300 Tolerance: 0.00001 Solver: Gradient Descent	F1 Score: 0.882 Precision: 0.920 Recall: 0.9768 Accuracy: 0.882

Model	Hyperparameters	Metrics
Linear Regression	Max Iterations: 20 Regularization Param: 0.2 Elastic Net Param: 0.8	MAE: 15.3747 RMSE: 35.3285
Decision Tree Regression	Max Depth: 5 Min Info. Gain: 0.0	MAE: 17.4483 RMSE: 35.9332
Random Forest Regression	Number of Trees: 50 Max Depth: 5 Min Info. Gain: 0.0	MAE: 17.6723 RMSE: 35.8424
Gradient Boosted Trees Regression	Max Depth: 3 Max Iterations: 15 Step Size: 0.0	MAE: 17.7356 RMSE: 35.9672

All with no sampling

Modeling: Experiment Results (with CV)

Model	Hyperparameters	Metrics
Logistic Regression	Max Iterations: 10 Regularization Param: 0.2 Elastic Net Param: 0.0	Accuracy: 0.7483 Precision: 0.8265 Recall: 0.9999 F1 Score: 0.7483 AUC: 0.7294
Decision Tree Classification	Max Depth: 5 Impurity: Gini Max Bins: 32 Min Info. Gain: 0.0	Accuracy: 0.7746 Precision: 0.8384 Recall: 0.9739 F1 Score: 0.7746 AUC: 0.5956
Random Forest Classification	Number of Trees: 50 Max Depth: 5 Impurity: Entropy Max Bins: 32 Min Info. Gain: 0.0	Accuracy: 0.7478 Precision: 0.8264 Recall: 1.0 F1 Score: 0.7478 AUC: 0.6744
Multiple Layer Perceptron	Max Iterations: 5 Step Size: 0.09 Block Size: 300 Tolerance: 0.00001 Seed: 1234 Solver: Gradient Descent	Accuracy: 0.7478 Precision: 0.8264 Recall: 1.0 F1 Score: 0.7478 AUC: 0.6734

Model	Hyperparameters	Metrics
Linear Regression	Max Iterations: 20 Regularization Param: 0.2 Elastic Net Parameter: 0.8	MAE: 18.0776 RMSE: 45.7000

*All with no sampling

Gap Analysis

- Join time: ~12 minutes
  - In line with groups with close to 42M rows in their dataset, like ours
- Training times a bit difficult to gauge given computation issues experienced by many groups
- Classification model:
  - One of the higher precision, recall, and F1 Scores
  - But given our knowledge on when our model works and doesn’t, we know we have room for improvement
- Regression model:
  - Not many teams using MAE
  - Our RMSE scores (~45.7) was similar to another group; but lower than another group that manage to get ~18.5

Credit Assignment

Task	Contributor(s)	Date Started	Date Completed	Hours Spent
Notebook Write Up	Everyone	11/30/2022	12/04/2022	20
Additional EDA	Oleg	11/30/2022	12/03/2022	10
Model Pipeline Development	Bri	12/01/2022	12/04/2022	15
Linear Model Experiments/GS	Neil, Annie	11/30/2022	12/03/2022	10
Tree-Based Model Experiments/GS	Annie	11/30/2022	12/03/2022	15
Neural Network Experiments/GS	Neil	12/01/2022	12/03/2022	15
Cross Validation Experiments/GS	Bri	11/30/2022	12/03/2022	10
Final Model Testing	Bri	12/01/2022	12/03/2022	5
Experimentation Write Up	Annie	12/02/2022	12/03/2022	3
Results Write Up	Bri	11/30/2022	12/04/2022	4
Gap Analysis	Oleg	12/04/2022	12/04/2022	2
Conclusion	Bri, Annie	12/03/2022	12/04/2022	2
Powerpoint Presentation (Short)	Neil	11/30/2022	12/04/2022	4
Powerpoint Presentation (Long)	Neil, Annie	11/30/2022	12/04/2022	4
2 Minute Video Update	Bri	12/04/2022	12/04/2022	20 min
Update Project Leaderboard	Oleg	12/04/2022	12/04/2022	5 min
Submission	Oleg	12/04/2022	12/04/2022	15 min

Cross Validation Function

```
# ---------- Split Data ---------- #
df_ranked = df.withColumn("rank", row_number().over(Window.partitionBy().orderBy("scheduled_departure_UTC")))
df_ranked = df_ranked.filter(col('Year') <= 2020).cache()
fold_size = df_ranked.count() / k

for i in range(k):

  # Split the original dataframe into folds
  fold_df = df_ranked.where(f"{i * fold_size} < rank").where(f" rank <= {(i+1) * fold_size}")
  # Split the fold into train and validation sets
  train = fold_df.where(f"rank <= {(fold_size * i) + (fold_size * 0.7)}").drop("rank")
  train = get_sampling(train, sampling)
  val   = fold_df.where(f"rank  > {(fold_size * i) + (fold_size * 0.7)}").drop("rank")

  # ---------- Impute and Scale Features ---------- #
  train_df_full = impute_and_scale_features(train)
  val_df_full   = impute_and_scale_features(val)

  # ---------- Train Model ---------- #
  if model_type == 'MultilayerPerceptronClassifier':
    model, ml_type = get_model(model_type, params, train_df_full.schema["features_all"].metadata["ml_attr"]["num_attrs"])
  else:
    model, ml_type = get_model(model_type, params)
  trained_model  = model.fit(train_df_full)
  predictions    = trained_model.transform(val_df_full)
```