Flight Delay Prediction

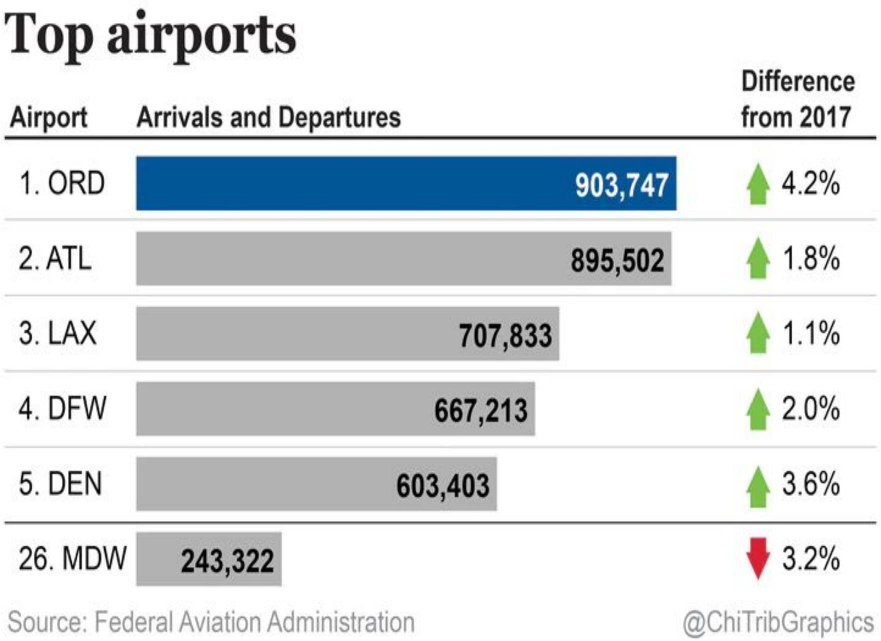
W261 - Fall 2020 Project Team 26

Hanyu Dai
Edward Tong
Shishir Agarwal
Praveen Kasireddy

Agenda

- Introduction of Business Case
- Problem Statement and Research Question
- Overall Approach
- Data Sets Overview
- Evaluation Metrics
- EDA
- Features Engineering
- Features Selection
- Algorithms Exploration and Implementation
- Performance and Scalability
- Limitations and Challenge
- Conclusion

Business Case



https://www.chicagotribune.com/news/breaking/ct-biz-ohare-flight-numbers-20190204-story.html

Total Cost of Delay in the U.S. (dollars, billion)

	2016	2017	2018	2019
Airlines	5.6	6.4	7.7	8.3
Passengers	13.3	14.8	16.4	18.1
Lost Demand	1.8	2.0	2.2	2.4
Indirect	3.0	3.4	3.9	4.2
Total	**23.7**	**26.6**	**30.2**	**33.0**

https://www.faa.gov/data_research/aviation_data_statistics/media/cost_delay_estimates.pdf

Total Cost of Delay

Problem Statement & Research Question

***S**: What are the Problems caused by Flight Delay?*

https://www.transportation.gov/briefing-room/dot8717

Q : Predicting ***"Departure"*** Flight

- *Delay* or *NO Delay*

https://www.airlines.org/dataset/per-minute-cost-of-delays-to-u-s-airlines/#

Definitions:

- Delay
  - >= 15 minute w.r.t to the planned time of departure (CRS).
- Prediction
  - "Two hours" ahead of CRS departure time

Overall Approach - Machine Learning Phases

EDA and Preprocessing

Analysis and Imputation of Missing Values

Shortlist of most important features

Understand and Visualize Data
- Histogram, Heatmaps, Correlation Matrix, Scatter Plots

Feature Engineering

Transforming and Scaling Features
- Numerical
- Categorical

Derived Variables

Algorithms Implementation

Logistics Regression

Decision Tree

Ensemble Random Forest Gradient Boosting

Hyperparameter Tuning

Adjusting Hyperparameters

Ensemble

Selecting best model based on prediction metrics

“Precision”
&
False Positive

Data Sets - Overview

1. **Raw** (before any processing)

	Period	No. of Features	No. of Records
Flights	2015-2019	109	31,746,841
Weather	2015-2019	177	630,904,436
Stations	NA	11	29,771

Other Dataset used:

- Airports data from openflights.org for time zone transformation

2. Modeling (after processing)

	Period	No. Of Features	No. Of Records
Training	2015-2018	53	16,528,705
Test	2019	53	5,082,228

Model Evaluation Metrics

Classifications :

- "Delayed"
  - **positive** class
- "No Delayed "
  - **negative** class

Year	No Delayed	Delayed
2015	4,675,372	1,057,554
2016	4,600,659	953,543
2017	4,580,433	1,013,845
2018	5,789,777	1,306,435

Accuracy : (TP +TN) / (TP + TN + FP + FN)

Precision: (TP) / (TP + FP)

Recall: (TP) / (TP + FN)

?

Model Evaluation Metrics - Confusion Matrix

Predictions

Actual

Flights	Delay	No Delay
Delay	True Positive , TP Reality: DELAY Prediction:DELAY Outcome: EVERYONE is Fine Eg. TP = 1	False Negative , FN Reality: DELAY Prediction: NO DELAY Outcome: EVERYONE are Annoyed E.g FN =8
No Delay	False Positive, FP Reality: NO DELAY Prediction: DELAY Outcome: EVERYONE is UNHAPPY Eg. FP = 1	True Negative, TN Reality: NO DELAY Prediction: NO DELAY Outcome: EVERYONE is FINE E.g TN = 90

Weather EDA - Forward Filling Null Data

Station	Date	Timestamp	WND_DIRECTION
72219013874	2015-01-01	2015-01-01T00:52:00.000+0000	330
72219013874	2015-01-01	2015-01-01T01:52:00.000+0000	310
72219013874	2015-01-01	2015-01-01T02:52:00.000+0000	null

Before

Code:

```
window = Window.partitionBy('STATION','date').orderBy('Timestamp').rowsBetween(-250, 0)
filled_column = last(weather_data['WND_Direction'], ignorenulls=True).over(window)
```

Station	Date	Timestamp	WND_DIRECTION
72219013874	2015-01-01	2015-01-01T00:52:00.000+0000	330
72219013874	2015-01-01	2015-01-01T01:52:00.000+0000	310
72219013874	2015-01-01	2015-01-01T02:52:00.000+0000	310

After

Weather EDA - Resolving Multiple Readings

Station	Date	Timestamp	WND_SPEED
72219013874	2015-01-02	2015-01-02T19:22:00.000+0000	36
72219013874	2015-01-02	2015-01-02T19:47:00.000+0000	21
72219013874	2015-01-02	2015-01-02T19:52:00.000+0000	31

Before

Code:

```
weather_data = weather_data.withColumn("DATE", f.date_trunc("hour", "DATE"))
weather_data = weather_data.groupBy("STATION", "DATE").agg(f.max("WND_SPEED"))
```

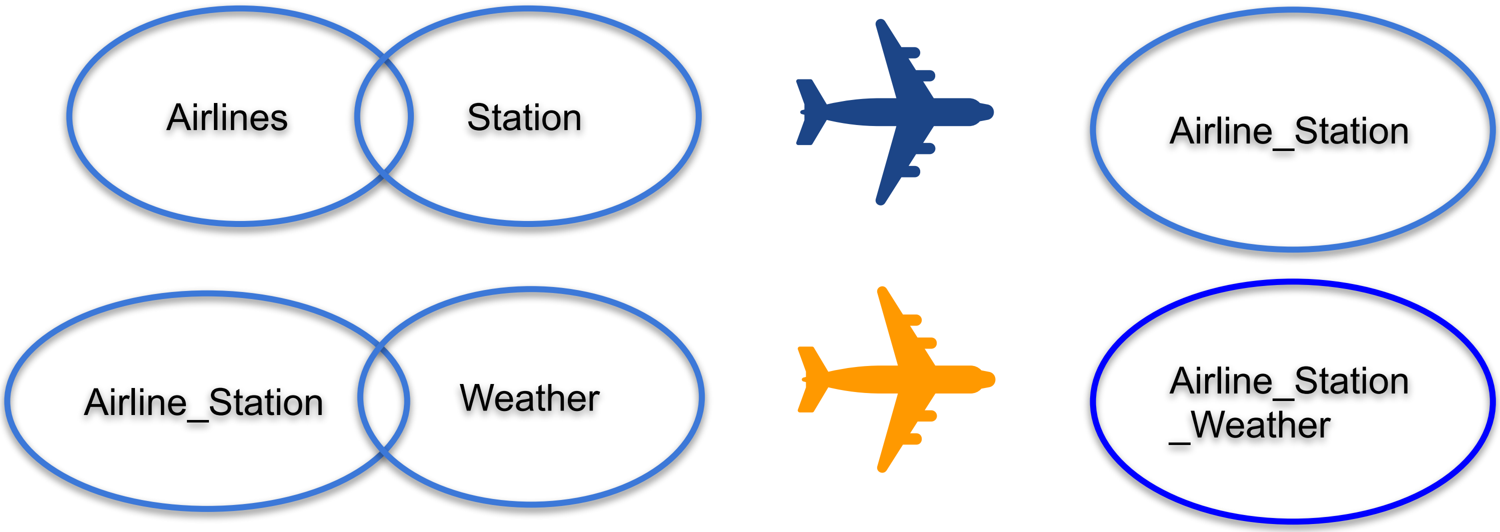
Station	Date	Timestamp	WND_SPEED
72219013874	2015-01-02	2015-01-02T19:22:00.000+0000	36
72219013874	2015-01-02	2015-01-02T19:47:00.000+0000	36
72219013874	2015-01-02	2015-01-02T19:52:00.000+0000	36

After

Joining Data

Code :

```
airlines.join(stations, (stations.call==airlines.ORIGIN) &
(stations.state==airlines.ORIGIN_STATE_ABR), 'inner')
```

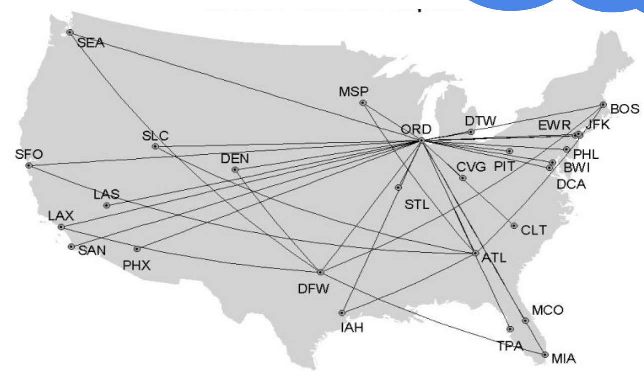


Code:

```
sqlContext.sql("SELECT * FROM airlines INNER JOIN weather ON airlines.station=
weather.station AND airlines.FL_TIMESTAMP_EARLY = weather.DATE")
```

Feature Engineering via Pagerank

Airport	Pagerank
ORD	14.276905975254800
ATL	13.212343130314500
MSP	8.033267082044770
IAH	7.423410445540550
DTW	7.199458684764240
CLT	7.167862270451910
SLC	6.195332319456120
SFO	6.166775954573200
EWR	5.836649337579180
PHX	5.440932366785200
LAX	5.368049462519130




```
# create Edge
E =
DF_train.groupBy(["ORIGIN","DEST","DEP_DELAY"]).count().withColumnRenamed("ORIGIN","src").withColumnRenamed("DEST","dst").withColumnRenamed("count","weight")

# create Vertics
V = DF_train.select(["DEST"]).distinct().withColumnRenamed("DEST", "id")

# create graph with E and G
airlinesGraph = GraphFrame(V, E)

ranks = airlinesGraph.pageRank(resetProbability=0.15, maxIter=5)
```

Feature Engineering

LST_DELAY | Departure delay of the last last flight

Over Clause in Spark SQL for iteration

Long turnaround time nullifies dep. delay

Use dep. delay of last last flight just in case

```
airlines_data = airlines_data.filter(airlines_data.TAIL_NUM.isNotNull())
airlines_data.registerTempTable("airlines_data")
airlines_data = sqlContext.sql("SELECT *, LAG(CRS_DEP) OVER (PARTITION BY TAIL_NUM ORDER BY CRS_DEP DESC) AS NXT_CRS_DEP, RANK() OVER (PARTITION BY TAIL_NUM ORDER BY CRS_DEP) AS rank FROM airlines_data")
airlines_data = airlines_data.withColumn('DEL_T',when((f.col('NXT_CRS_DEP').cast('long') - f.col('CRS_ARR').cast('long'))/60 <= 90, airlines_data.DEP_DELAY).otherwise(0))
w = Window().partitionBy().orderBy(*['TAIL_NUM', 'CRS_DEP'])
airlines_data = airlines_data.select("*", lag("DEL_T", offset=2).over(w).alias("LST_DELAY"))
airlines_data = airlines_data.withColumn('LST_DELAY',when(f.col('rank') == 1, 0).otherwise(f.col('LST_DELAY')))
airlines_data = airlines_data.withColumn('LST_DELAY',when(f.col('rank') == 2, 0).otherwise(f.col('LST_DELAY')))
```

Feature Engineering

Aircraft arriving late
Air carrier delay
NAS delay
Security delay

Flight #/hr | Number of incoming/outgoing flights per hour

GroupBy actual flight time
and origin/destination

```
airlines_data = airlines_data.withColumn("ACT_DEP", (f.unix_timestamp(airlines_data['CRS_DEP'])
                                         + 60*airlines_data['DEP_DELAY']).cast('timestamp'))
airlines_data = airlines_data.withColumn("CRS_ARR", (f.unix_timestamp(airlines_data['ACT_DEP'])
                                         + 60*airlines_data['CRS_ELAPSED_TIME']).cast('timestamp'))
airlines_data = airlines_data.withColumn("ACT_ARR", (f.unix_timestamp(airlines_data['ACT_DEP'])
                                         + 60*airlines_data['ACTUAL_ELAPSED_TIME']).cast('timestamp'))

airlines_data = airlines_data.join(airlines_data.groupBy('ORIGIN','ACT_DEP_DATE','ACT_DEP_HOUR').count()
                      .withColumnRenamed("count", "outgoing_hour"),
                      on=['ORIGIN','ACT_DEP_DATE','ACT_DEP_HOUR'], how='inner')

airlines_data = airlines_data.join(airlines_data.groupBy('DEST','ACT_ARR_DATE','ACT_ARR_HOUR').count()
                      .withColumnRenamed("DEST", "ORIGIN").withColumnRenamed("count", "incoming_hour"),
                      on=['ORIGIN','ACT_ARR_DATE','ACT_ARR_HOUR'], how='inner')
```

Feature Engineering

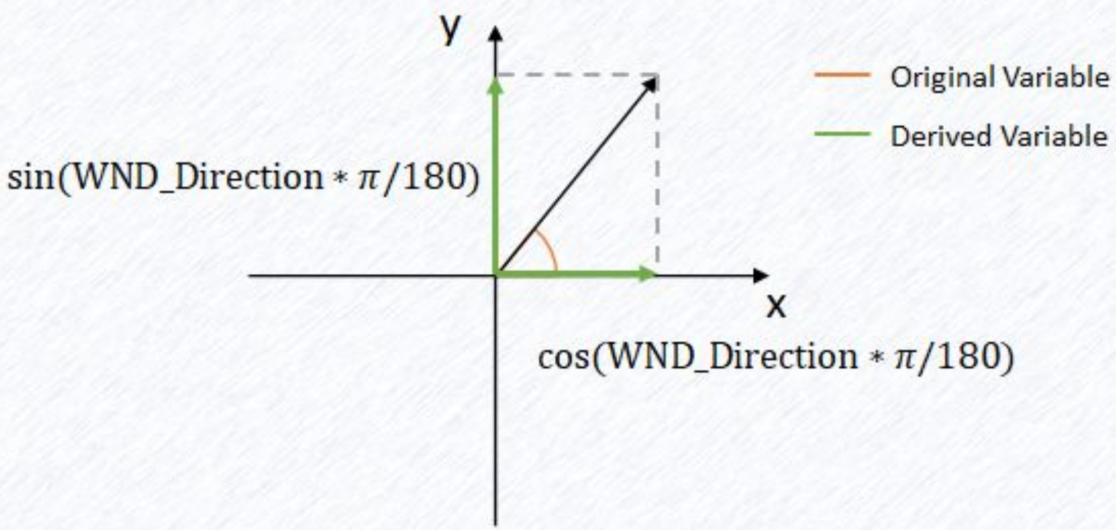
DELAY% | Percentage of delayed flights per hour

GroupBy actual flight time, origin, (DEP_DEL15).

```
DF_delay_count.registerTempTable("airlines_data_delay_count")
current_delay_stats= sqlContext.sql("SELECT cur_fl_early_utc, ORIGIN,
(sum(DEP_DEL15)*count(distinct(earlier_fl_tail_num))/count(*)) as DEP_DEL15_COUNT,
(sum(DEP_DEL15)/count(*)) as DEP_DEL15_RATE,
(sum(DEP_DELAY_NEW)*count(distinct(earlier_fl_tail_num))/count(*)) as DEP_DELAY_NEW_SUM,
(sum(DEP_DELAY_NEW)/count(*)) as DEP_DELAY_NEW_MEAN, count(distinct(earlier_fl_tail_num)) as
flcount from airlines_data_delay_count group by cur_fl_early_utc, ORIGIN sort by cur_fl_early_utc,
Origin")
```

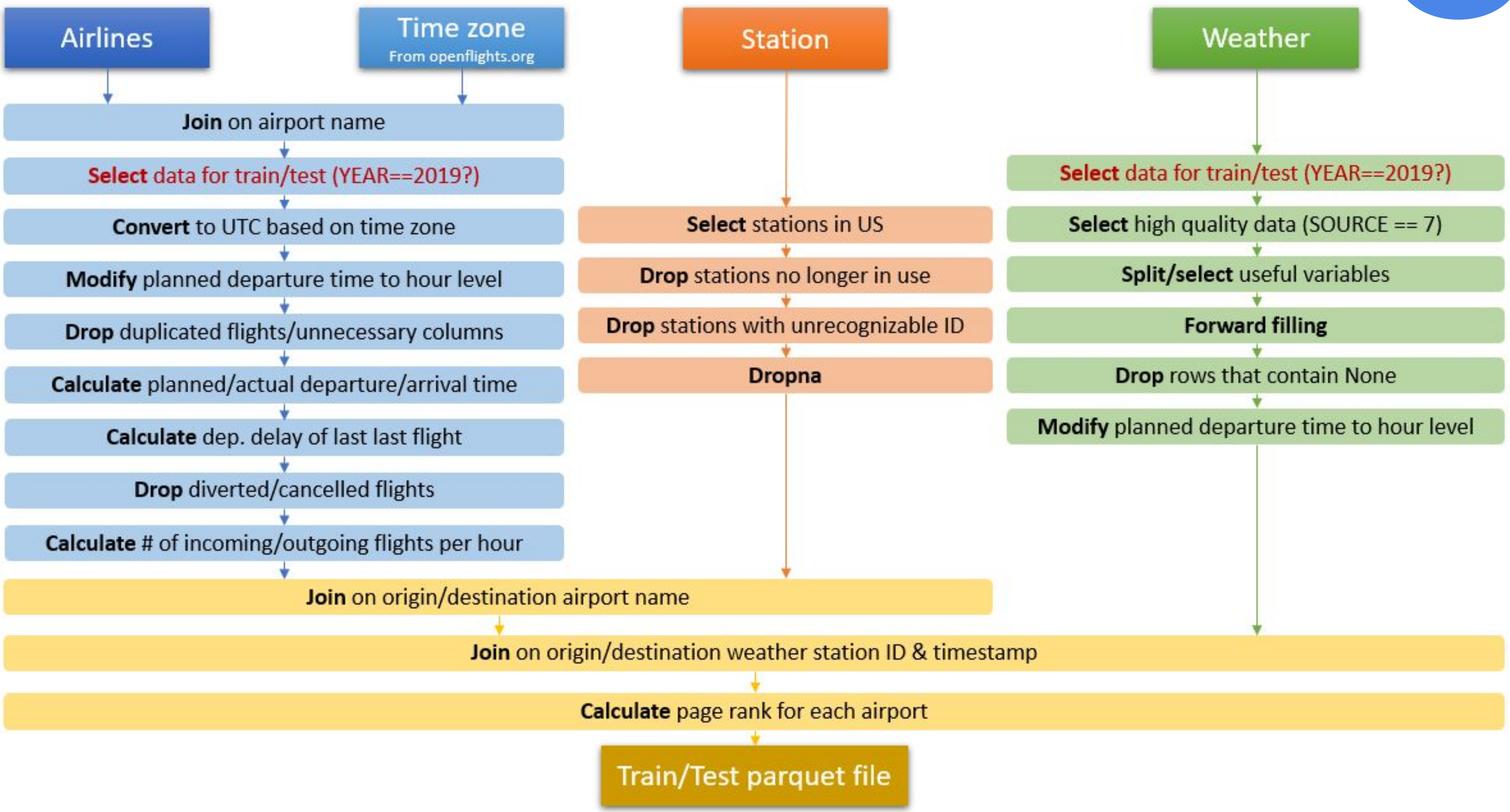
Feature Engineering

WND_x&y | Decomposition of wind direction (cyclical variable)




```python
# Handle Cyclical Variables
weather_data = weather_data.withColumn('WND_Direction_x', cos(col('WND_Direction')*np.pi/180))
weather_data = weather_data.withColumn('WND_Direction_y', sin(col('WND_Direction')*np.pi/180))
```

Pipeline



Airlines
Time zone
From openflights.org
Station
Weather
Join on airport name
Select data for train/test (YEAR==2019?)
Convert to UTC based on time zone
Modify planned departure time to hour level
Drop duplicated flights/unnecessary columns
Calculate planned/actual departure/arrival time
Calculate dep. delay of last last flight
Drop diverted/cancelled flights
Calculate # of incoming/outgoing flights per hour
Select stations in US
Drop stations no longer in use
Drop stations with unrecognizable ID
Dropna
Select data for train/test (YEAR==2019?)
Select high quality data (SOURCE == 7)
Split/select useful variables
Forward filling
Drop rows that contain None
Modify planned departure time to hour level
Join on origin/destination airport name
Join on origin/destination weather station ID & timestamp
Calculate page rank for each airport
Train/Test parquet file

Features Selection

Objective: Limit the input features to the important once.

Techniques Explored:

1. RandomForest featureImportance
2. Chi-square test
3. Lasso

```
Rf = RandomForestClassifier(labelCol="DEP_DEL15", featuresCol="features")
RfModel = Rf.fit(DF_train)
print(RfModel.featureImportances)
```

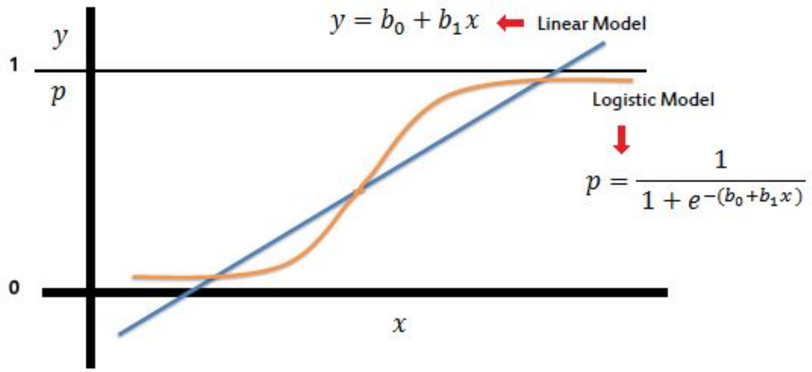
Feature	**Feature Importance**
LST_DELAY	0.546916
HOUR_OF_DAY	0.318057
OP_CARRIER_WN	0.025992
Outgoing_hour	0.022456
Tail_freq	0.020698
MA1_Altimeter_origin	0.018813
WND_Speed_origin	0.012826
MA1_Altimeter_dest	0.011495
Pagerank_DEST	0.011383
TMP_Air_origin	0.011366

Algorithms - Logistic & *Linear Regression*

Base Models

- Logistic Regression
  - Dependent Variable: DEP_DEL15
- Linear Regression
  - Dependent Variable: DEP_DELAY

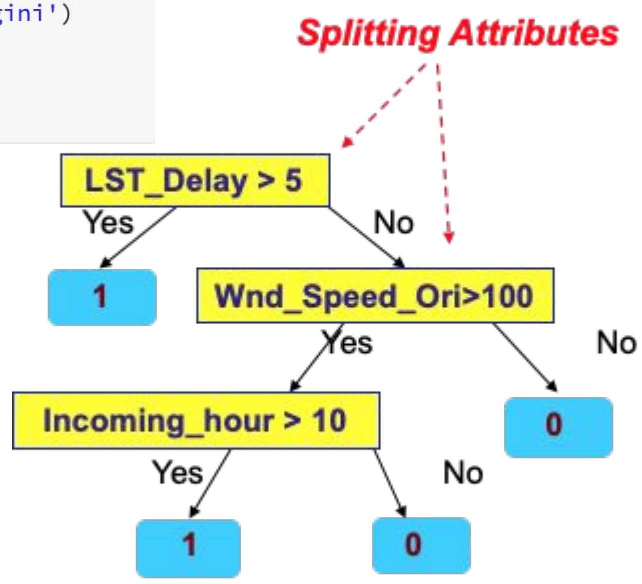
- Input Variables Selection:
  - Feature Importance
  - PCA



Algorithms - *Decision Tree*

- Non-parametric supervised learning method
    - Output: DEP_DEL15
    - Features: ['LST_DELAY','outgoing','incoming',”MONTH','DAY_OF_WEEK' ..]

```
# fit the model with training data and preidct with testing data
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'DEP_DEL15',
                            maxDepth = 10, maxBins=300, impurity='gini')
dtModel = dt.fit(trainingData)
dtpredict_train = dtModel.transform(trainingData)
dtpredict_test = dtModel.transform(testData)
```



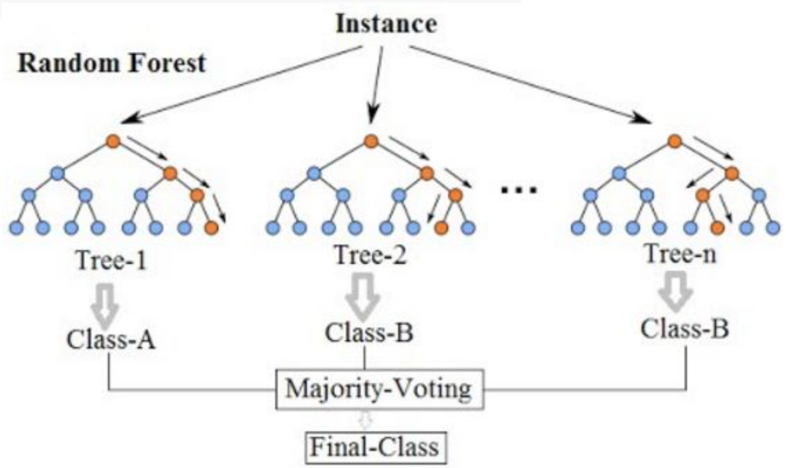
	Training Data	Test Data
Precision	0.706	0.709
Recall	0.178	0.154
F1 Score	0.284	0.253

Algorithms - *Random Forest*

- Large number of simple trees, combined at the end of the process.
  - Output: DEP_DEL15
  - Features: ['LST_DELAY','outgoing','incoming',"MONTH','DAY_OF_WEEK' ..]

```
# fit the model with training data and preidct with testing data
from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'DEP_DEL15',
                            numTrees=20, maxDepth=15,  seed=42)
rfModel = rf.fit(trainingData)
rfpredict_train = rfModel.transform(trainingData)
rfpredict_test = rfModel.transform(testData)
```

	Training Data	Test Data
Precision	0.782	0.757
Recall	0.158	0.125
F1 Score	0.263	0.214

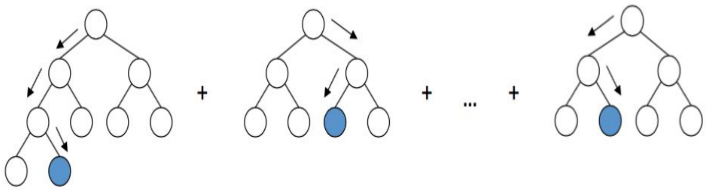


Algorithms - *Gradient Boosting Trees (GBT)*

- Combine decision trees, but start the combining at the beginning
  - Output: DEP_DEL15
  - Features: ['LST_DELAY','outgoing','incoming',''MONTH','DAY_OF_WEEK' ..]

```
# fit the model with training data and preidct with testing data
from pyspark.ml.classification import GBTClassifier
gbt = GBTClassifier(featuresCol='features', labelCol='DEP_DEL15',
                    maxDepth=15, maxIter=10, seed=32)
gbtModel = gbt.fit(trainingData)
gbtpredict_train = gbtModel.transform(trainingData)
gbtpredict_test = gbtModel.transform(testData)
```

	Training Data	Test Data
Precision	0.789	0.645
Recall	0.277	0.214
F1 Score	0.410	0.322



Algorithms Performance Summary

Actual	Predicted								
		LogReg		DT		RF		GBT	
		Delay	No Delay	Delay	No Delay	Delay	No Delay	Delay	No Delay
	Delay	104,206	2,961,920	534,065	2,474,418	474,953	2,533,530	833,324	2,175,159
	No Delay	88,712	13,372,165	221,913	13,038,834	132,561	13,128,186	222,313	13,038,434

Models	LogReg		DT		RF		GBT	
Metrics	Training	Test	Training	Test	Training	Test	Training	Test
Precision	0.54	0.539	0.706	0.709	0.782	0.757	0.789	0.645
Recall	0.034	0.039	0.178	0.154	0.158	0.125	0.277	0.215
F1 Score	0.064	0.073	0.284	0.253	0.263	0.214	0.41	0.322
Accuracy	0.815	0.813	0.834	0.829	0.836	0.828	0.853	0.830

Performance and Scalability

Learnings

- Caching and persisting parquet files
- Incremental Joins & Intermediate Tables

Limitation and Challenges

- Couldn’t evaluate model performance w.r.t. Time taken.
- Scope to get improved model performance if we explore more hyperparameter tuning via grid search.

Conclusion

1. Recommended Algorithm : Random Forest

Predicted

Actual

	Delay	No Delay
Delay	474,953	2,533,530
No Delay	132,561	13,128,186

Random Forest

	Training	Test
Precision	0.782	0.757
Recall	0.158	0.125
F1	0.263	0.214
Accuracy	0.836	0.828

2. ML Learned and Applied:

- Spark framework with RDD (distribute and parallelize computation)
- Graph algorithm and Pageranks (derived additional features for modeling)
- One hot encoding and normalization (optimized the approach)
- Logistics regression, decision trees , random forest and gradient boosting trees (achieved good result with advance algorithm)

Thank You!

Reference

1. Bureau of Transportation Statistics

   https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236

   https://www.bts.gov/topics/airlines-and-airports/understanding-reporting-causes-flight-delays-and-cancellations
2. Matching station codes to airports:

   http://dss.ucar.edu/datasets/ds353.4/inventories/station-list.html

   https://www.world-airport-codes.com/
3. UC Berkeley School of Information, MIDS W261 - Fall 2020, all course materials
4. Open flights data

   http://openflights.org
5. “Machine Learning Crash Course” https://developers.google.com/machine-learning/crash-course
6. “Labs Graph Theory -Small World Network” http://web.math.princeton.edu/math_alive/5/Lab1/Networks2.html