Allay Airway Delay!

Predicting Flight Delays to Reduce Wasted Customer Time

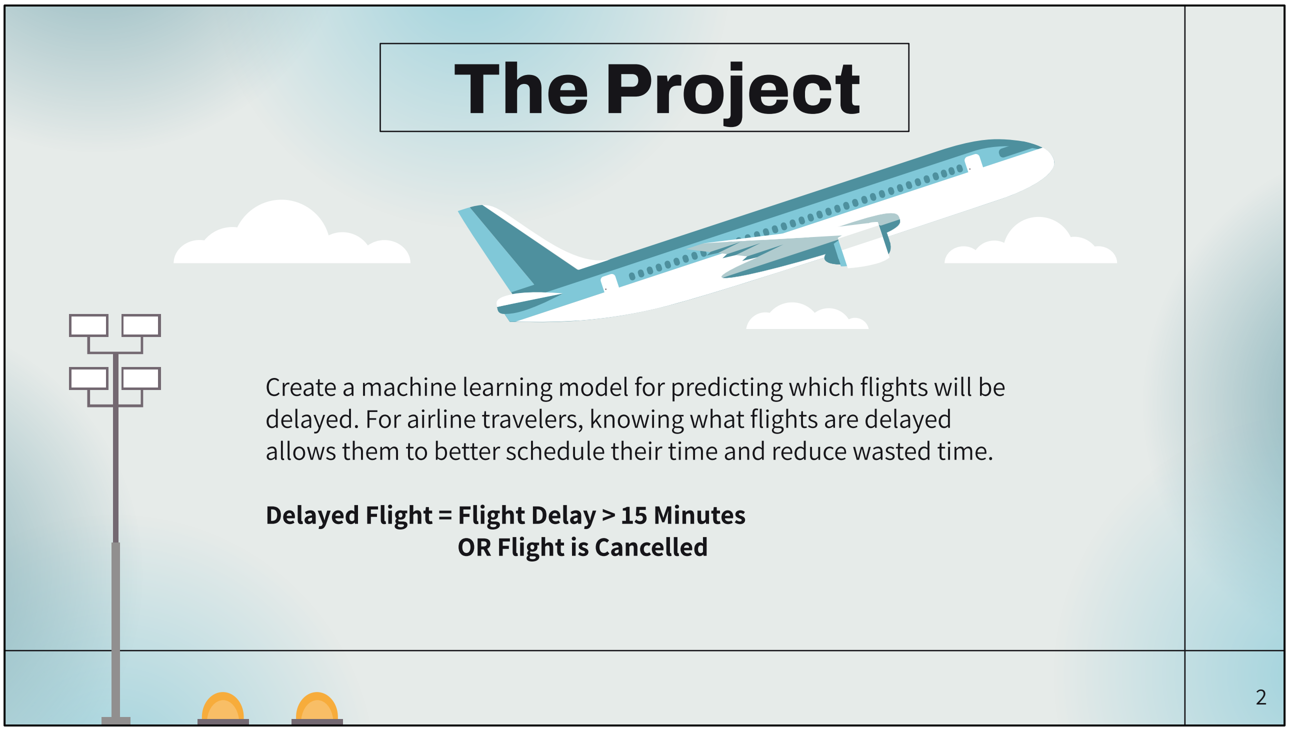
Team 13
Sparks and Stripes Forever
Nashat Cabral
Deanna Emery
Nina Huang
Ryan S. Wong

Nash

Hello! We are Team 13, also known as Sparks and Stripes Forever, and we are here to make flying a little less miserable for our users.

# Hello! We are Team 13, also known as Sparks and Stripes Forever, and we are here to make flying a little less miserable.

# One of the most common frustrations experienced in an airport is discovering that your flight has been delayed. You put in all that time and effort to be on-time, but it's your plane that's lagging behind. Wouldn't it be nice if air travellers like us had the ability to know whether their flight was delayed ahead of time?



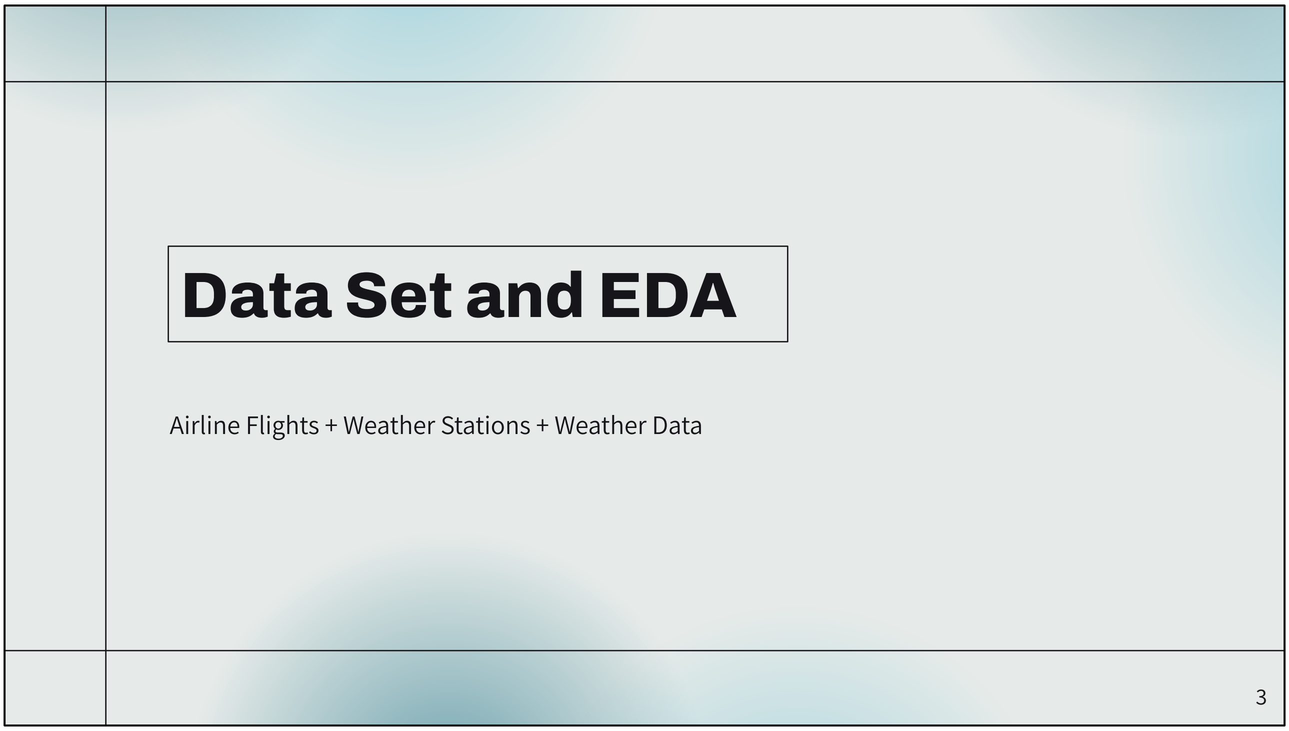
The Project

Create a machine learning model for predicting which flights will be delayed. For airline travelers, knowing what flights are delayed allows them to better schedule their time and reduce wasted time.

Delayed Flight = Flight Delay > 15 Minutes
OR Flight is Cancelled

Nash

As a reminder, our goal is to create a machine learning model that will predict whether or not a flight will be delayed by 15 minutes or more given information from 2 hours prior to the scheduled departure time. Our aim is improve the experience of air travellers, reducing the amount of time they waste waiting for a delayed flight with high precision.

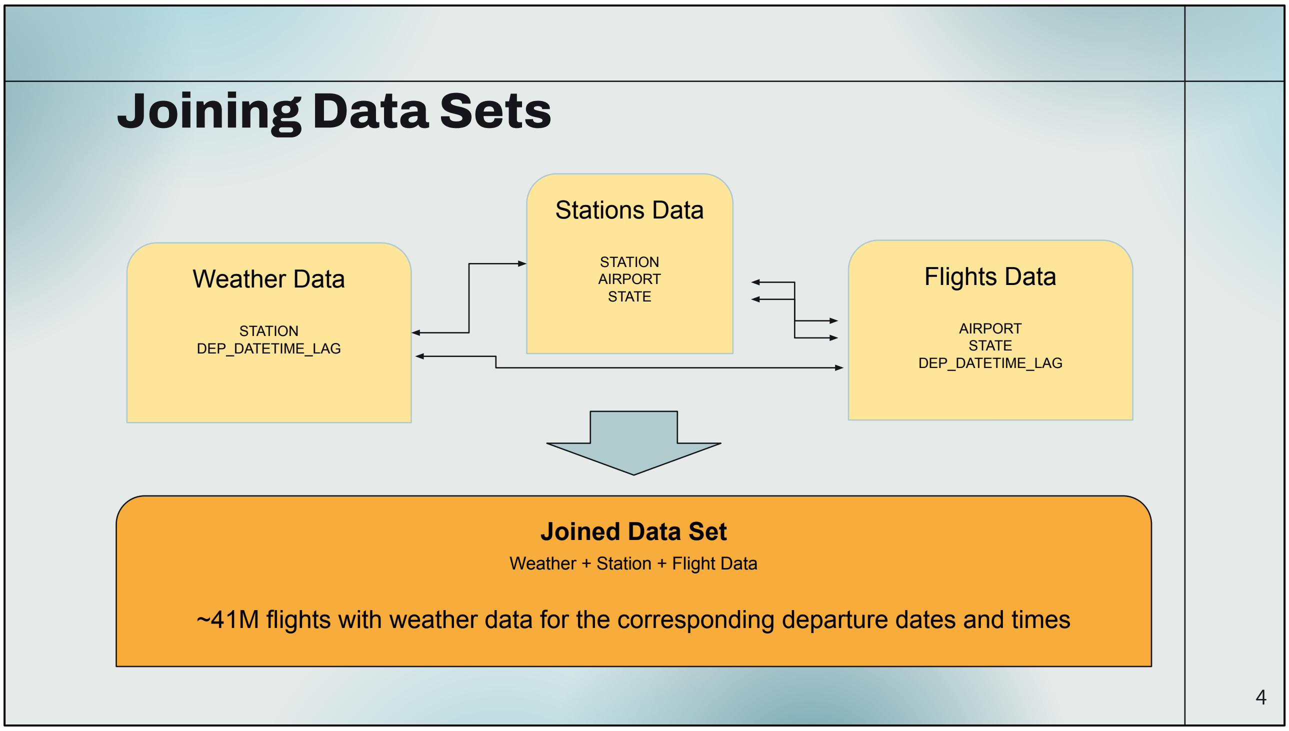


Data Set and EDA

Airline Flights + Weather Stations + Weather Data

Deanna - 20s

We started with three raw, disparate data sets, including airline data for over 70 million flights from 2015 to 2021, weather station data for hundreds of weather stations, and weather data accumulated across several million weather reports. This data was raw, messy, and bloated with irrelevant features, but we had to make it work.



Joining Data Sets

Stations Data

STATION
AIRPORT
STATE

Weather Data

STATION
DEP_DATETIME_LAG

Flights Data

AIRPORT
STATE
DEP_DATETIME_LAG

Joined Data Set

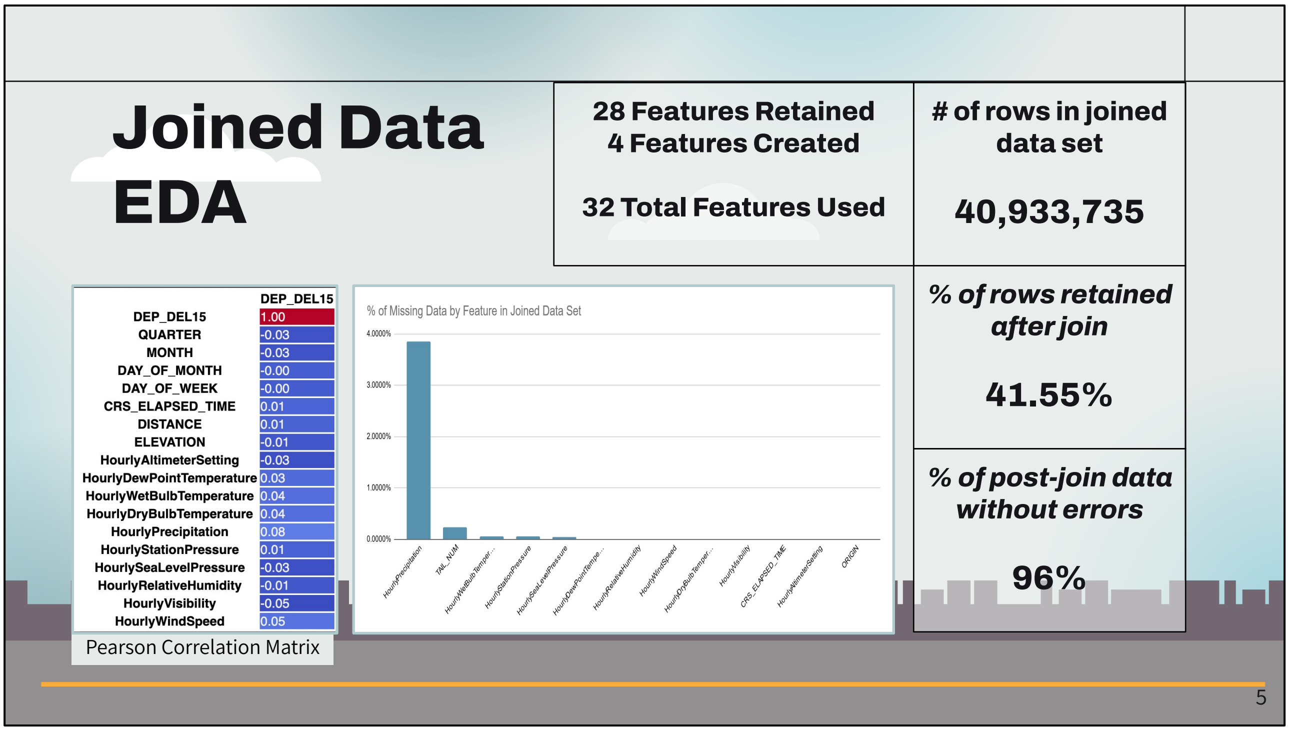
Weather + Station + Flight Data

~41M flights with weather data for the corresponding departure dates and times

4

Deanna - 1:15

The first crucial step to enable our analysis was to join the various data sources together. We connected the weather stations with the flights by associating airports with any nearby weather stations, which was done using the airport codes and states. Then, in the flights data, we created a new feature, DEP_DATETIME_LAG, which represents the timestamp two hours prior to the flight departure time, rounded down to the hour. Similarly, we created a corresponding time feature in the weather data with the record time rounded UP to the nearest hour. These two time features allowed us to connect flights and their weather stations with the corresponding weather reports from at least two hours prior in order to prevent any data leakage.
At this point we had multiple records of weather data for each flight from each of the nearby weather stations. Aggregating all of these records proved too time intensive, so instead we selected the two closest and the furthest weather stations from each airport and averaged their weather data for each time interval. This allowed us to maximize our data coverage to account for some stations that had missing data.
Finally, we converted all times to UTC so that we can better track flights that cross time zones.
The join took a total of 11 minutes to run from start to finish on 4 cores for the full dataset. The resulting data contained approximately 41 million records.



Joined Data EDA

28 Features Retained
4 Features Created

32 Total Features Used

# of rows in joined data set

40,933,735

% of rows retained after join

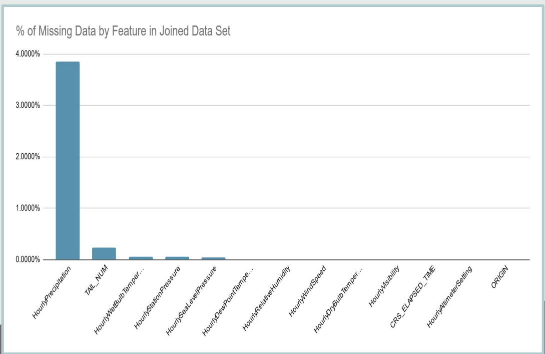
41.55%

% of post-join data without errors

96%

	DEP_DEL15
DEP_DEL15	1.00
QUARTER	-0.03
MONTH	-0.03
DAY_OF_MONTH	-0.00
DAY_OF_WEEK	-0.00
CRS_ELAPSED_TIME	0.01
DISTANCE	0.01
ELEVATION	-0.01
HourlyAltimeterSetting	-0.03
HourlyDewPointTemperature	0.03
HourlyWetBulbTemperature	0.04
HourlyDryBulbTemperature	0.04
HourlyPrecipitation	0.08
HourlyStationPressure	0.01
HourlySeaLevelPressure	-0.03
HourlyRelativeHumidity	-0.01
HourlyVisibility	-0.05
HourlyWindSpeed	0.05

Pearson Correlation Matrix

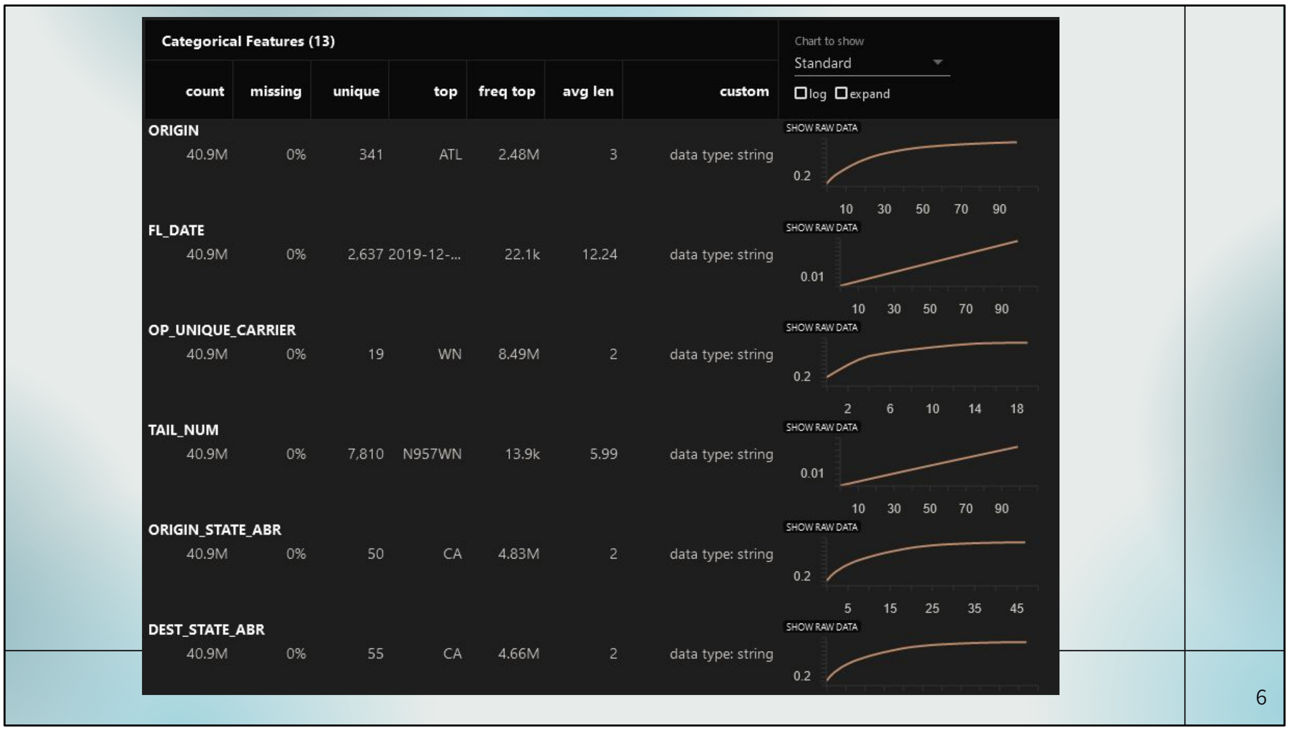


Nash

Our initial EDA on the raw data showed that each of the three datasets were rife with missing data and extraneous features too numerous to state here. With our joined data set, we narrowed our focus by picking or creating 32 relevant features to use in our joined data set. Fortunately, our joined data set was both sufficiently large and considerably clean: 40.9 million rows with 96% of them being completely free of missing values.

From our pearson correlation matrix (left) we can see there are not many features with strong notable correlations

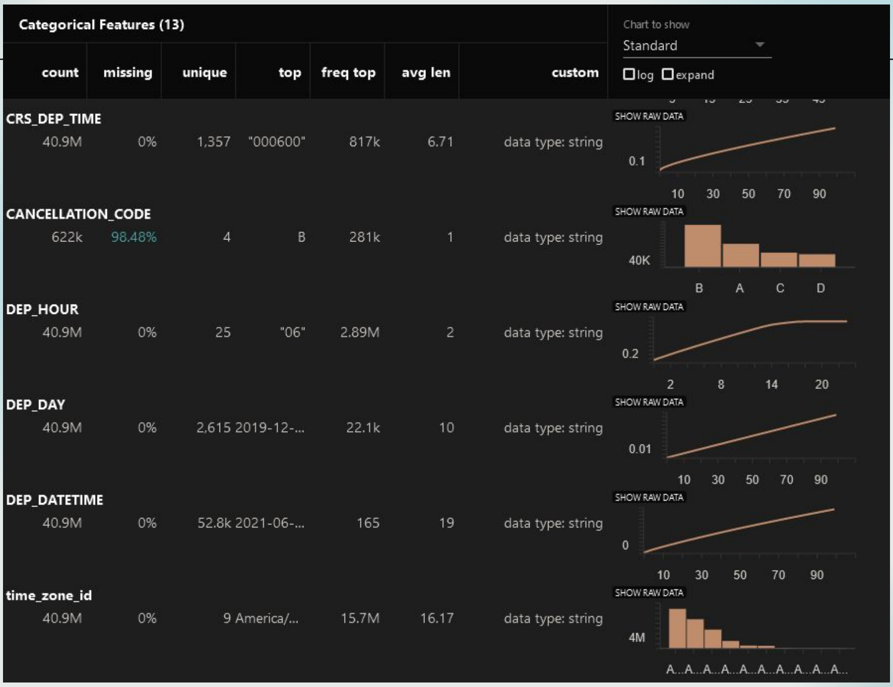
From the visuals



Categorical Features (13)

	count	missing	unique	top	freq top	avg len	custom
ORIGIN	40.9M	0%	341	ATL	2.48M	3	data type: string
FL_DATE	40.9M	0%	2,637	2019-12-...	22.1k	12.24	data type: string
OP_UNIQUE_CARRIER	40.9M	0%	19	WN	8.49M	2	data type: string
TAIL_NUM	40.9M	0%	7,810	N957WN	13.9k	5.99	data type: string
ORIGIN_STATE_ABR	40.9M	0%	50	CA	4.83M	2	data type: string
DEST_STATE_ABR	40.9M	0%	55	CA	4.66M	2	data type: string

Here we can see some data characteristics of interest on our categorical variables such as:
55 Destination States
50 Origin States

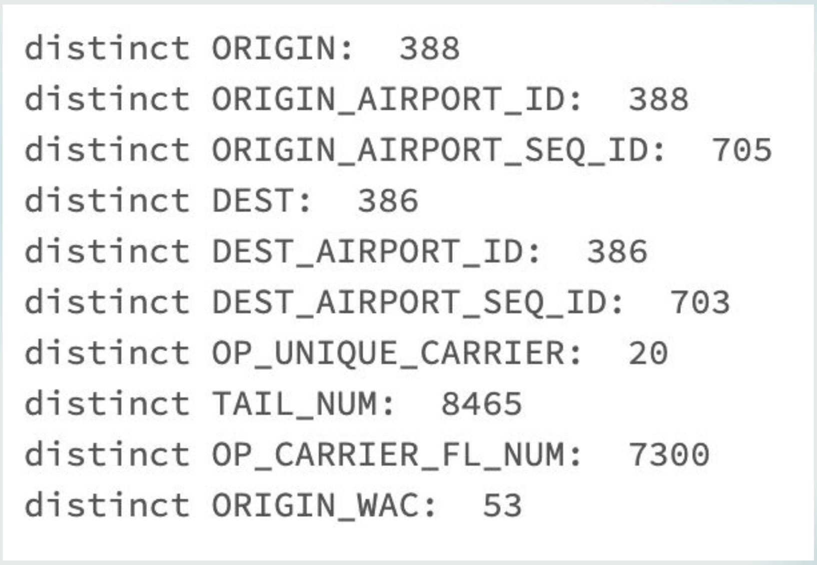


Categorical Features (13)

Chart to show: Standard

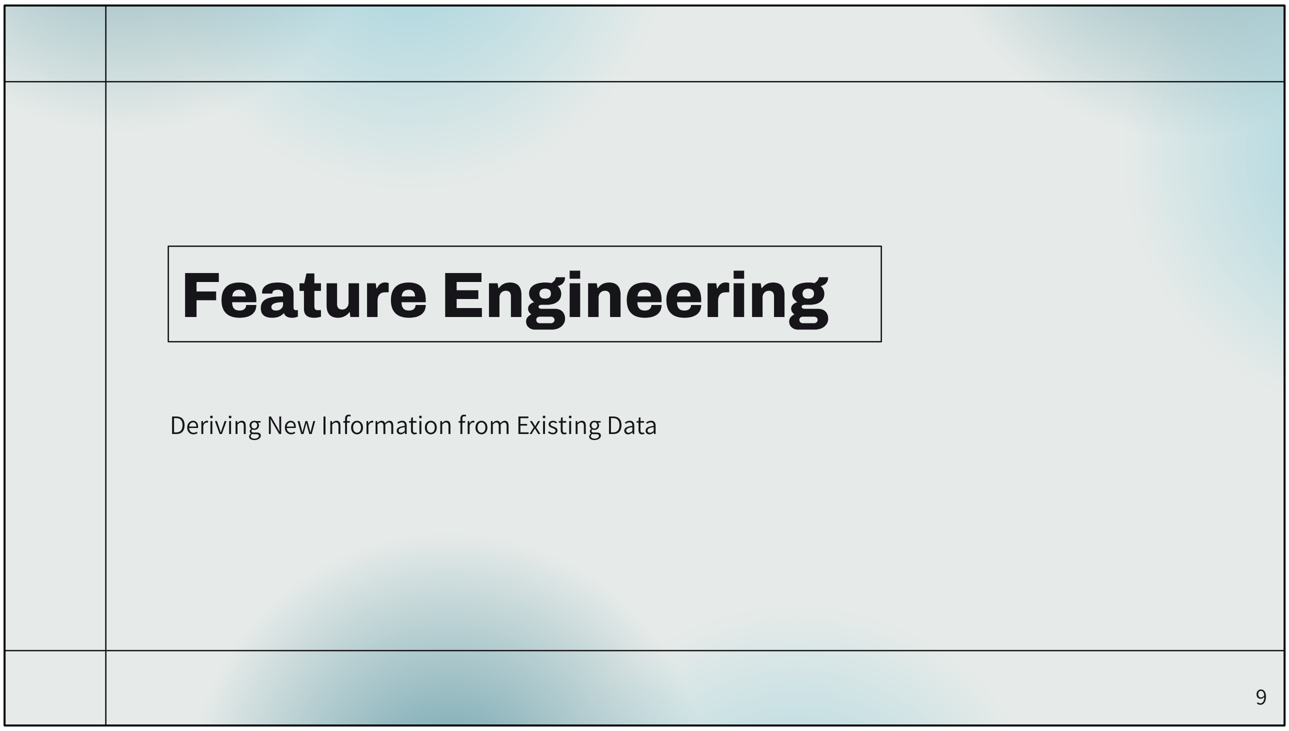
	count	missing	unique	top	freq top	avg len	custom
CRS_DEP_TIME	40.9M	0%	1,357	"000600"	817k	6.71	data type: string
CANCELLATION_CODE	622k	98.48%	4	B	281k	1	data type: string
DEP_HOUR	40.9M	0%	25	"06"	2.89M	2	data type: string
DEP_DAY	40.9M	0%	2,615	2019-12-...	22.1k	10	data type: string
DEP_DATETIME	40.9M	0%	52.8k	2021-06-...	165	19	data type: string
time_zone_id	40.9M	0%	9	America/...	15.7M	16.17	data type: string

Nash
Continued, we see
2615 Unique Days



```
distinct ORIGIN:  388
distinct ORIGIN_AIRPORT_ID:  388
distinct ORIGIN_AIRPORT_SEQ_ID:  705
distinct DEST:  386
distinct DEST_AIRPORT_ID:  386
distinct DEST_AIRPORT_SEQ_ID:  703
distinct OP_UNIQUE_CARRIER:  20
distinct TAIL_NUM:  8465
distinct OP_CARRIER_FL_NUM:  7300
distinct ORIGIN_WAC:  53
```

Lastly we have some distinct counts of key categorical features:
8465 Dist tail_number
7300 Op Carrier FL_NUM
388 FLight Origins
386 Flight Destinations

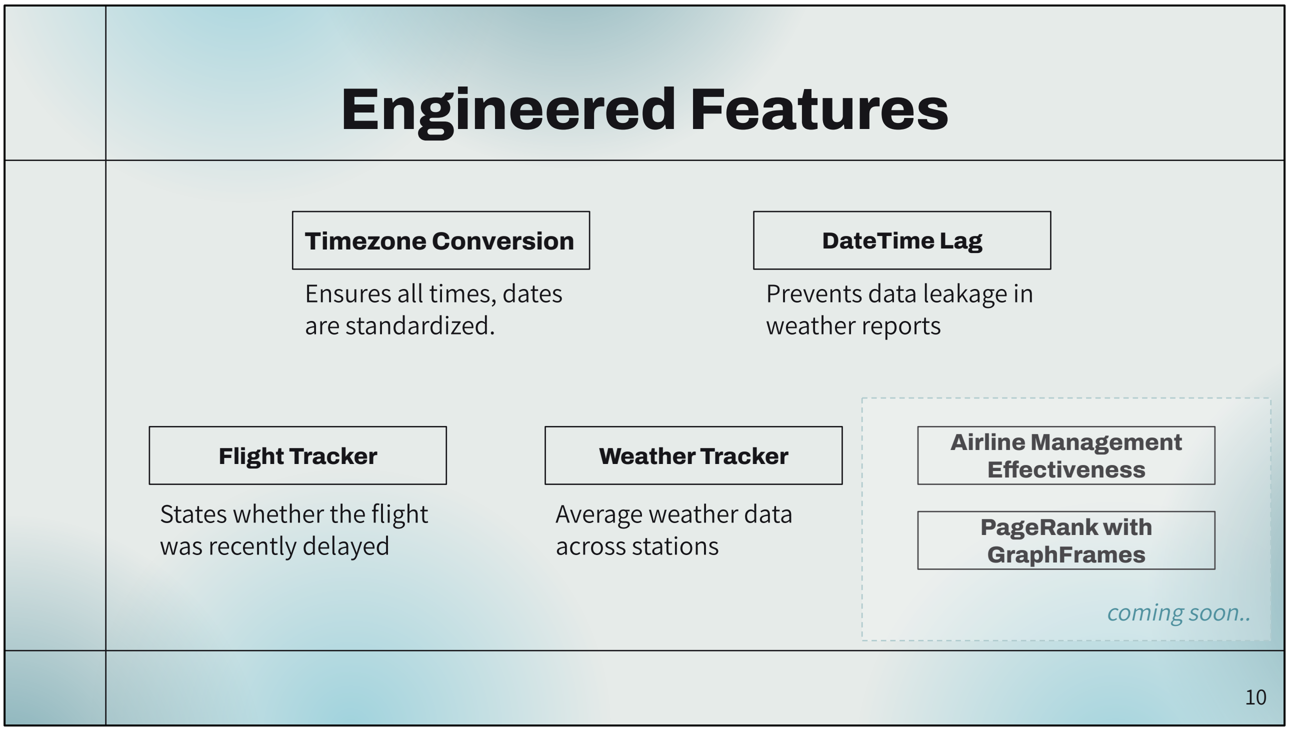


Feature Engineering

Deriving New Information from Existing Data

Nina (~23 sec)

Now we will look into Feature Engineering. While there were plenty of available features in the raw data, some of the more predictive data came from features we had to create ourselves.



Engineered Features

Timezone Conversion
Ensures all times, dates are standardized.

DateTime Lag
Prevents data leakage in weather reports

Flight Tracker
States whether the flight was recently delayed

Weather Tracker
Average weather data across stations

Airline Management Effectiveness

PageRank with GraphFrames

coming soon..

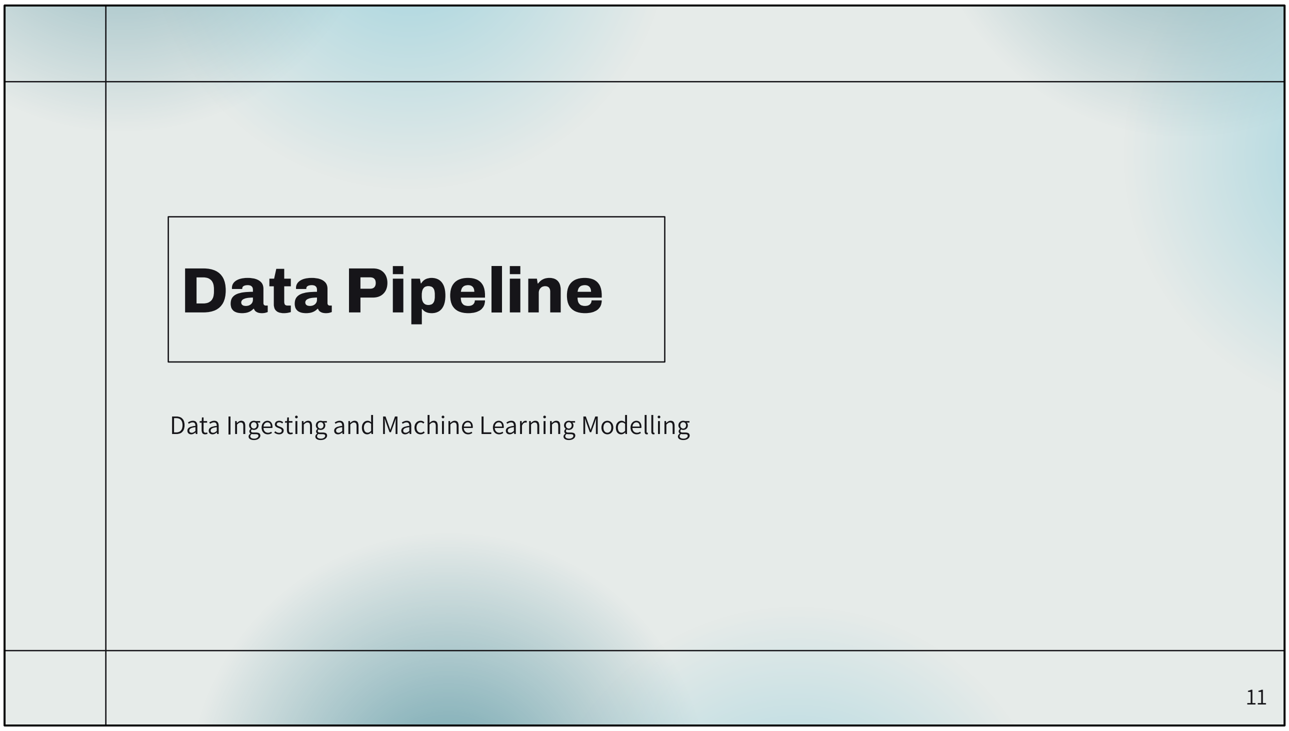
10

Nina (~1min 15 sec)
So what are these features?
We first created 2 sets of helper features which built the basis of our analysis. Deanna already went through these, but very quickly to summarize, we have local time converted to UTC, and Date Time lag for us to consistently check against data leakage

Next we created some highly predictive features, which are
- A flight tracker feature, which is an indicator feature that states whether a flight was recently delayed. We assume that a plane which was delayed will likely have its subsequent flights delayed as well.
- We also tracked weather across multiple stations and time period which Deanna previously mentioned
- Finally, we plan to look into airline management and identify which airlines have fewer delays than others.

    // done

Not presented:
- (We currently see this airline management effectiveness feature to be categorical. Specifically we will look for management effectiveness buckets based on historical delays patterns. Depending on how many buckets are identified, we may one hot encode this feature)

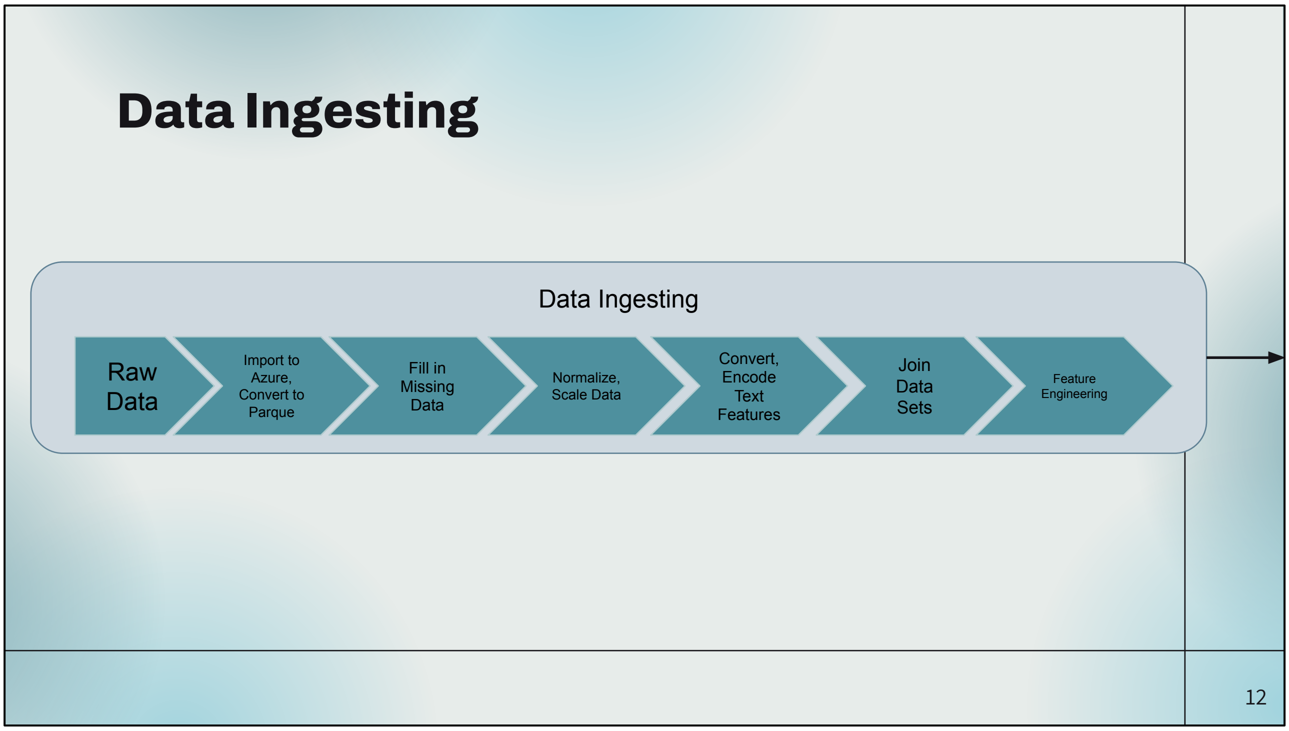


Data Pipeline

Data Ingesting and Machine Learning Modelling

Ryan ~ 13 seconds

Data joining and feature engineering are but two steps in the data pipeline that we have constructed. From start to finish, our pipeline takes in the raw data, transforms it, and then uses it in machine learning.

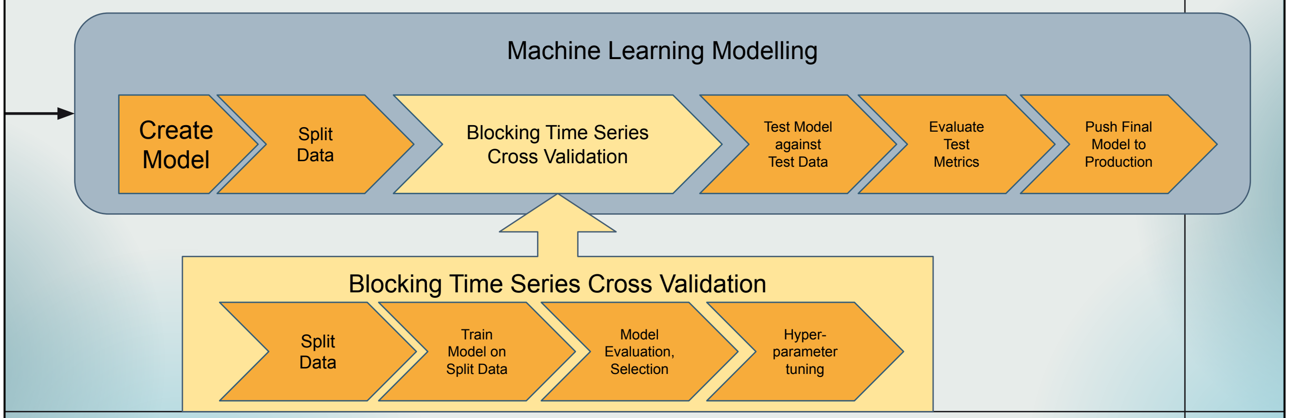


Data Ingesting

Ryan ~ 35 seconds

The first half of the pipeline focuses on ingesting and refining the data. From raw data, we save it to Azure in the Parquet format, making it highly available in a computationally efficient format. For missing data, we impute values that can be reasonably inferred, and drop rows where the missing data is ambiguous. We then convert text features with StringIndexing and one hot encode them using sparse vectors, including those features in our analysis in a space efficient manner. Then, we create new features and join the data set to finish refining the data.

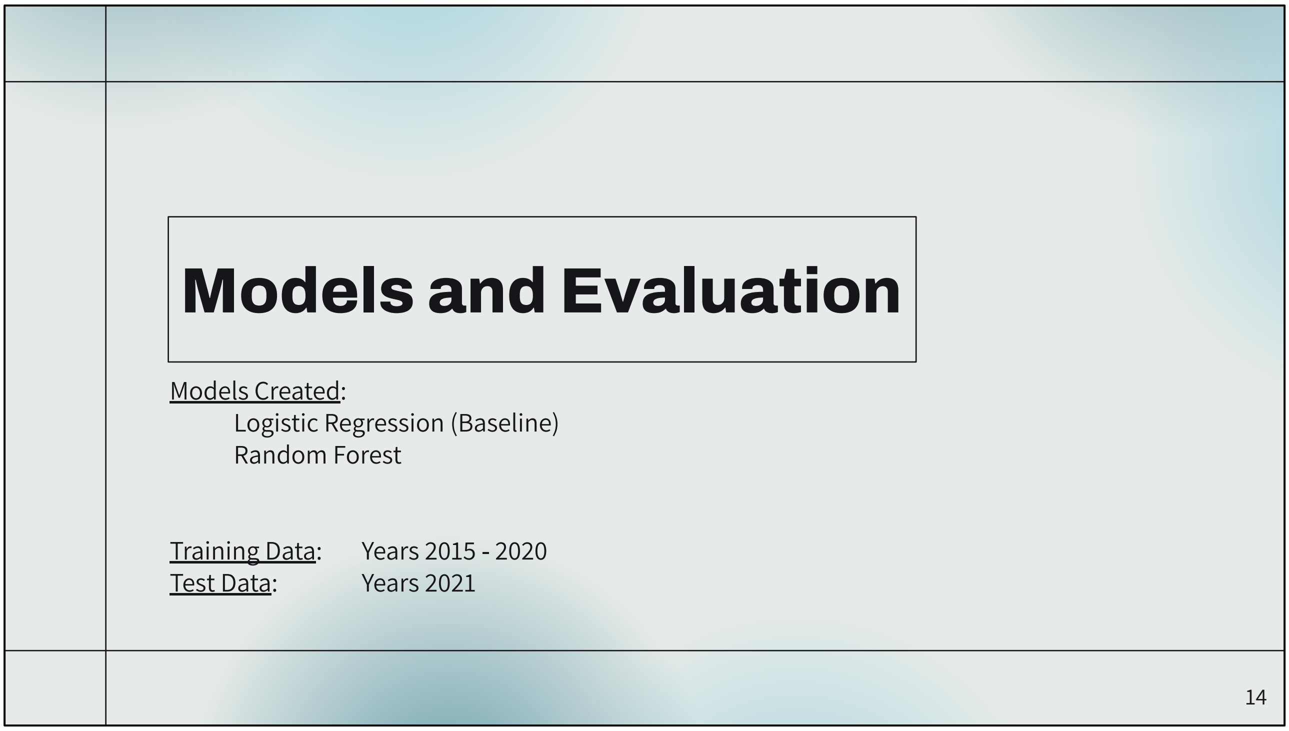
Machine Learning Modelling


Machine Learning Modelling
Create Model
Split Data
Blocking Time Series Cross Validation
Test Model against Test Data
Evaluate Test Metrics
Push Final Model to Production
Blocking Time Series Cross Validation
Split Data
Train Model on Split Data
Model Evaluation, Selection
Hyper-parameter tuning

Ryan ~ 45 seconds

The second half of the pipeline takes that refined data and uses it in machine learning modelling. We start by creating a generic model, and also split the data into a training set and a test set that we will use to test on all flights from the year 2021. We will then utilize a custom-built process called Blocking Time Series Cross Validation- which will be discussed in the next section- in order to create the best model possible. We then apply that best model onto a data set it has never seen before in order to evaluate its potential real-world accuracy. We now have our best model and an idea of how well it will perform in the real world.

All this being done, I sure am happy that we did not have to code all of this from scratch thanks to tools like MLLib, DataBricks, Spark, and Azure.



Models and Evaluation

Models Created:
- Logistic Regression (Baseline)
- Random Forest

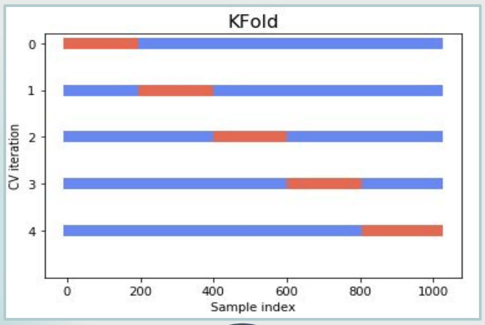
Training Data:	Years 2015 - 2020
Test Data:	Years 2021

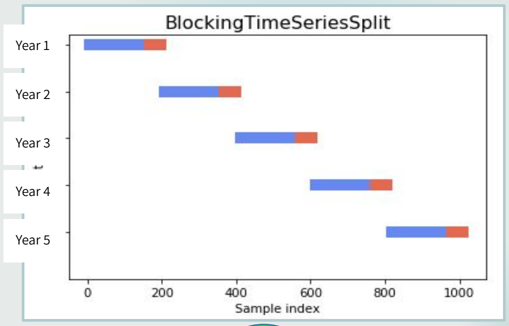
Ryan ~ 35 seconds

After we have ingested our data, we can start building and evaluating our models. We started with a logistic regression as a baseline model to compare other models against, since that model is easy to put implement despite poor performance with binary categorization tasks. We also implemented a random forest model, and were hoping to present it tonight, but it is still training and evaluating as we speak.

For evaluation purposes, we will be training our models on data from the years 2015 through 2020, and then evaluating it on unseen data from the year 2021.

Blocking Time Series Cross Validation

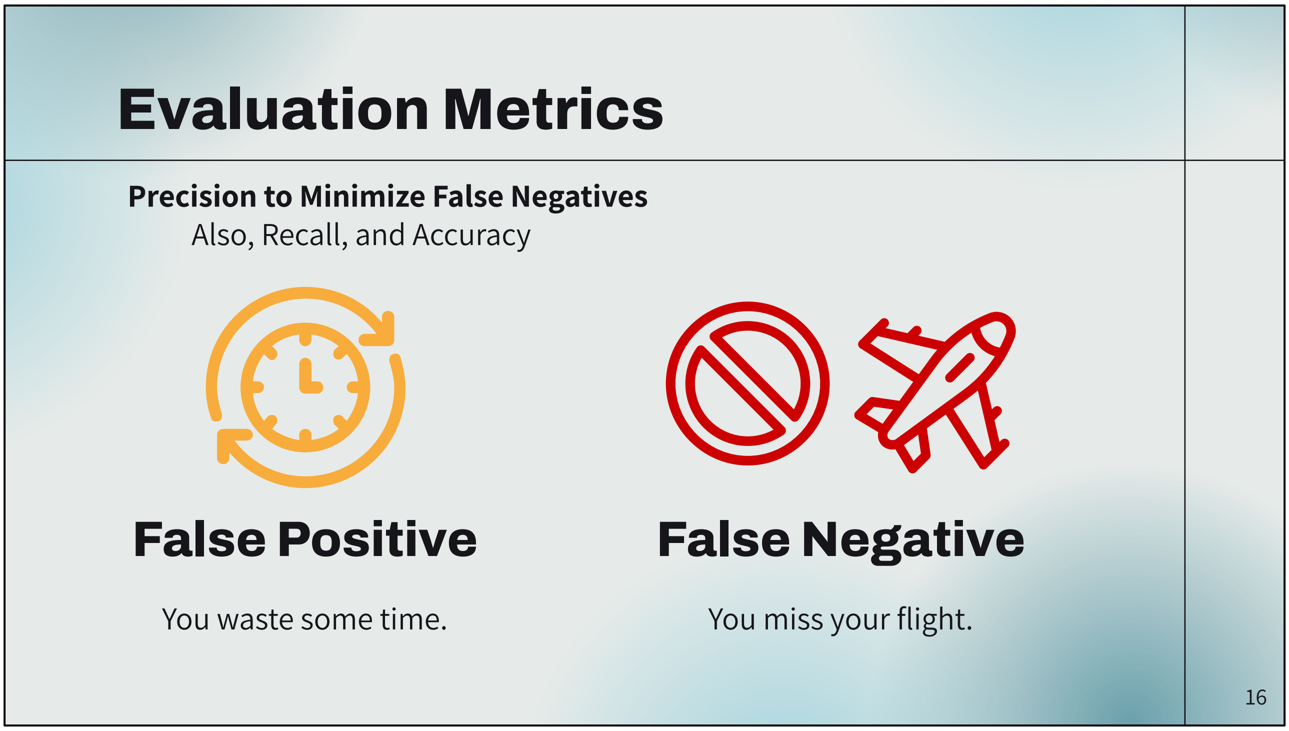




Images From: *https://hub.packtpub.com/cross-validation-strategies-for-time-series-forecasting-tutorial/*

Ryan ~ 1 minute

Since we are evaluating time series data, we need to use time series cross validation for accurately training and evaluating our models. We could not use the standard KFold cross validation because it is flawed with time series: KFolds introduces time gaps in the data, tests on data occurring before the training data, and it leaks data when the model memorizes future data it should not have seen yet. As such, we chose to build our own version of cross validation called BlockingTimeSeriesSplit,. Blocking Time Series Split will split the training data by year, builds a model for each year, trains that model on data from the first 70% of that year, and then tests that model on the data from the latter 30%. The model with the best metrics when tested is chosen as our best model to be evaluated against the 2021 test data. This allows us to cross validate our time series data without the complications of KFolds.



Evaluation Metrics

Precision to Minimize False Negatives
Also, Recall, and Accuracy

False Positive

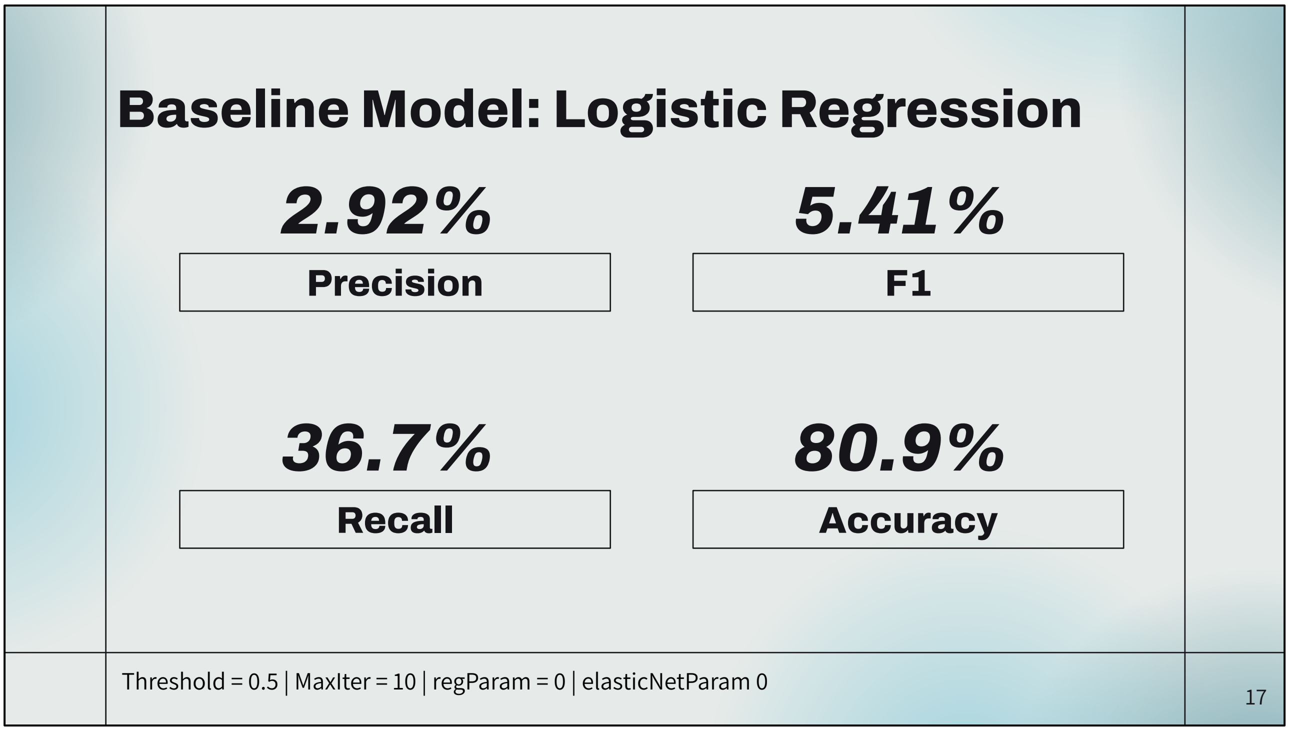
You waste some time.

False Negative

You miss your flight.

Nash

When evaluating our model, we opted to focus on Precision as our primary metric, with Recall and Accuracy being secondary metrics. We based this decision on our goal to minimize the amount of false negatives our model creates. Thinking about our use case of informing users of flight delays A false positive would mean that your plane is late but you're on-time, therefore you waste a little time. But a false negative would mean the plane is on-time but you are not, so you would miss your flight entirely. We want to minimize that worst case scenario for our users.



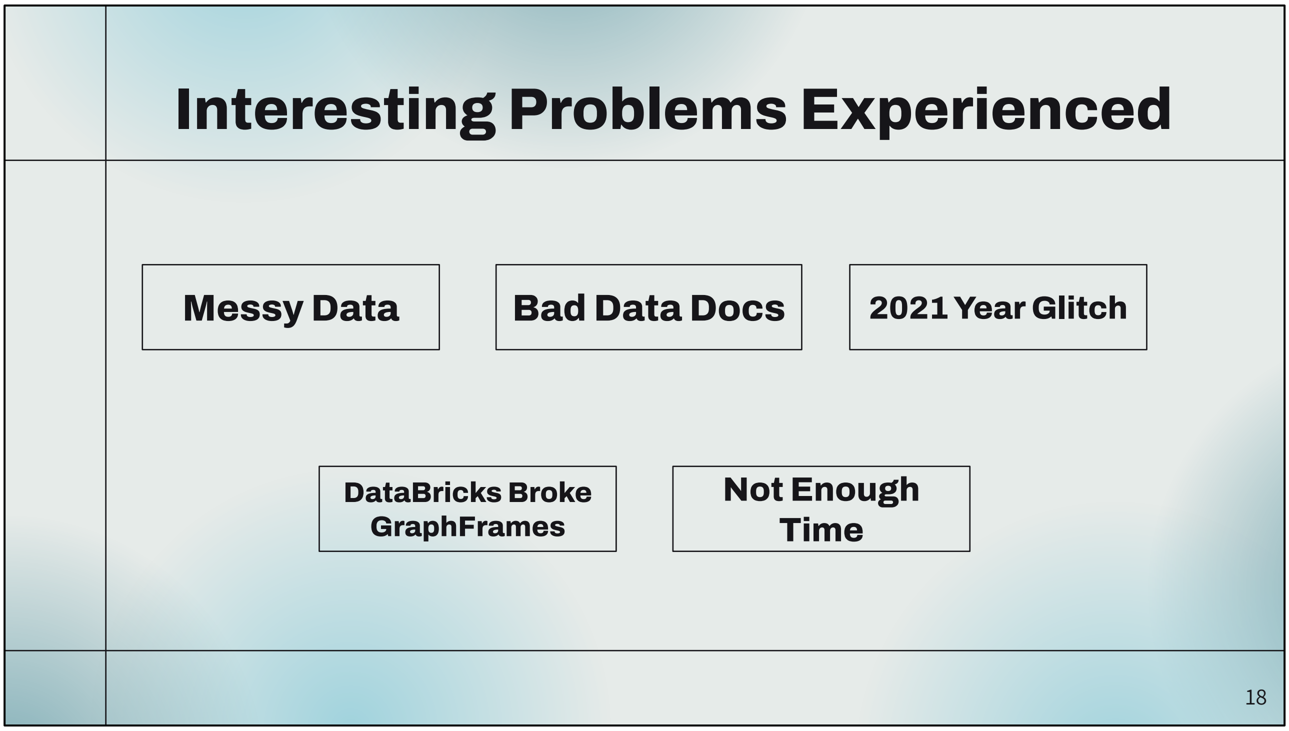
Baseline Model: Logistic Regression

2.92%	5.41%
Precision	F1
36.7%	80.9%
Recall	Accuracy

Threshold = 0.5 | MaxIter = 10 | regParam = 0 | elasticNetParam 0

Ryan

So, about that Logistic Regression model. After being trained, cross-validated, and evaluated against our test data, this baseline model has an astounding precision of… 2.92%. Recall and accuracy look good, but such low precision was surprising.



Interesting Problems Experienced

Messy Data

Bad Data Docs

2021 Year Glitch

DataBricks Broke GraphFrames

Not Enough Time

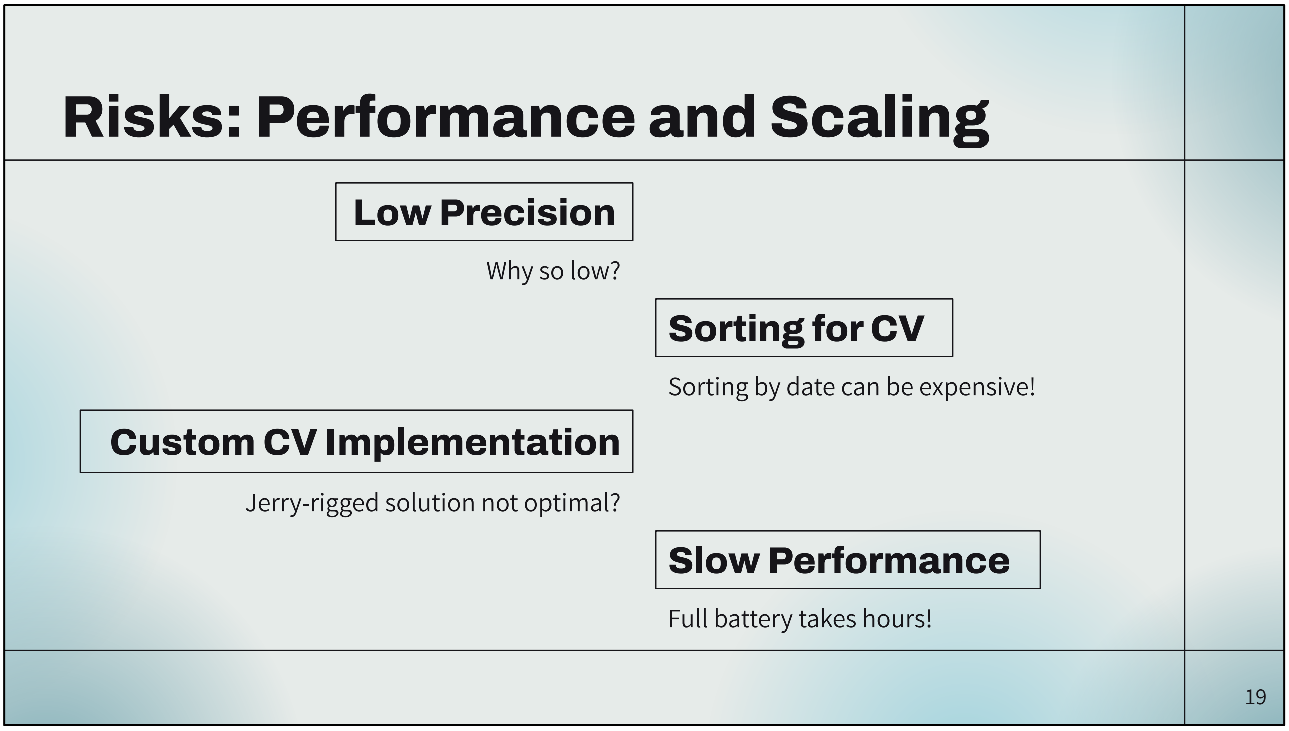
Deanna - 1 min

Throughout our analysis thus far, we've come across some strange or interesting errors!

As mentioned before, the raw data was quite messy.
Notably, the data documentation was not always complete or consistent with the data, making interpretation difficult.
We also came across issues early on in the join phase with airport codes in the stations dataset that didn't quite always contain Ks at the beginning of the code. We also realized, having discovered that 2021 was entirely missing from our joined dataset, that the dates and times in the airports data changed formats in 2021.

Beyond the data, we have been facing issues with the DataBricks infrastructure that prevent us from using GraphFrames on more than a single node cluster. Unfortunately, attempting to run something like PageRank on a single node would not be feasible given time constraints, so we have been exploring alternative methods.

This leads me to perhaps the largest challenge we face: it seems like we never have enough time to do everything we want to do!



Risks: Performance and Scaling

Low Precision

Why so low?

Sorting for CV

Sorting by date can be expensive!

Custom CV Implementation

Jerry-rigged solution not optimal?

Slow Performance

Full battery takes hours!

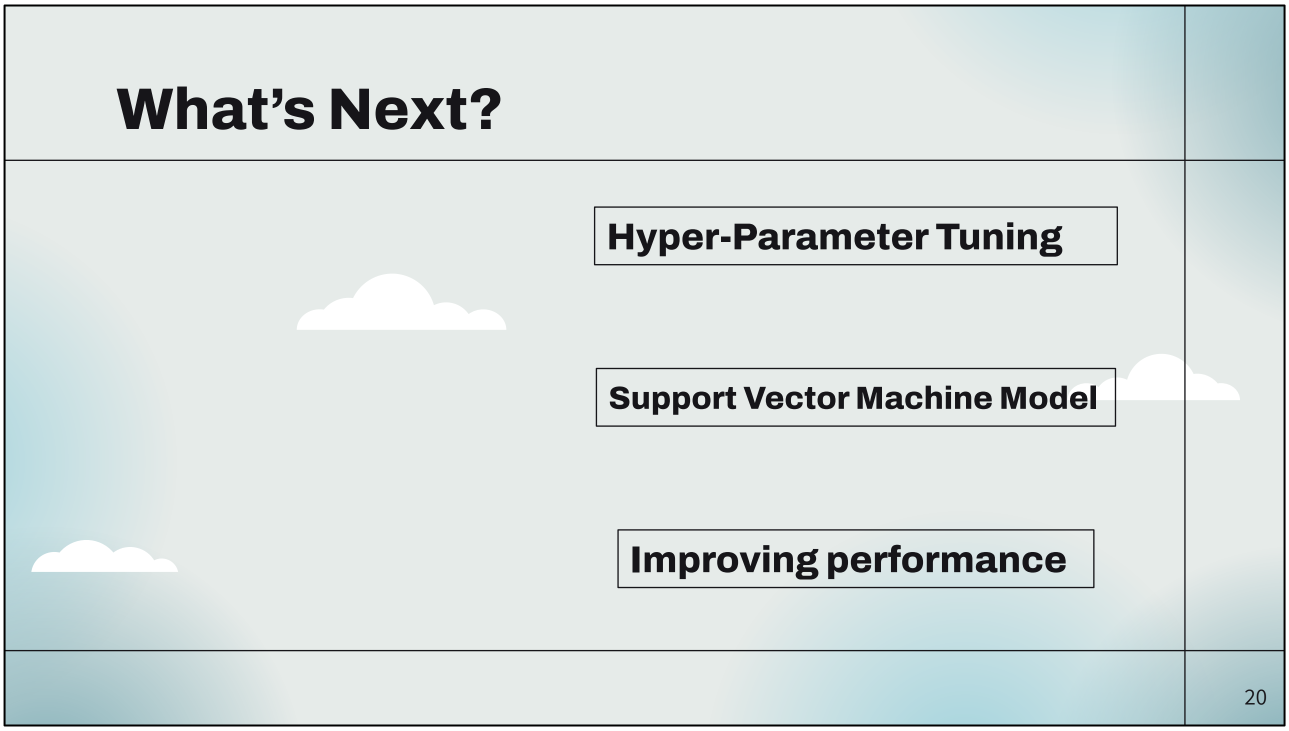
Nina (50 sec - 1 min 06 sec)
As much work as we've done so far, we are still working to fix some issues. We recognize that although we are able to fix some, others may be more difficult to fully resolve within the given timeline. Specifically this could be a concern for issues that might be inherent to the project methodology.

So what are some of these issues:
As mentioned before, the precision of our model was low. We will continue to improve it iteratively through feature selection, feature engineering and hyper parameter tuning. So more to come

Next, For our time series cross validation, the data had to be sorted before being split. We recognize this shuffle is expensive at scale

Furthermore, the libraries that we've explored so far (MLLib) did not have out-of-the-box APIs that we could leverage for our cross validation. As such we created our own user defined cross validation function, which may not be optimized for performance

Lastly, our training and testing tasks take a long time to complete. We want to highlight it because they do eat into our project timeline. We plan to improve the speed by removing features with low predictive power, and also ask our project sponsor for more resources on our clusters



What's Next?

Hyper-Parameter Tuning

Support Vector Machine Model

Improving performance

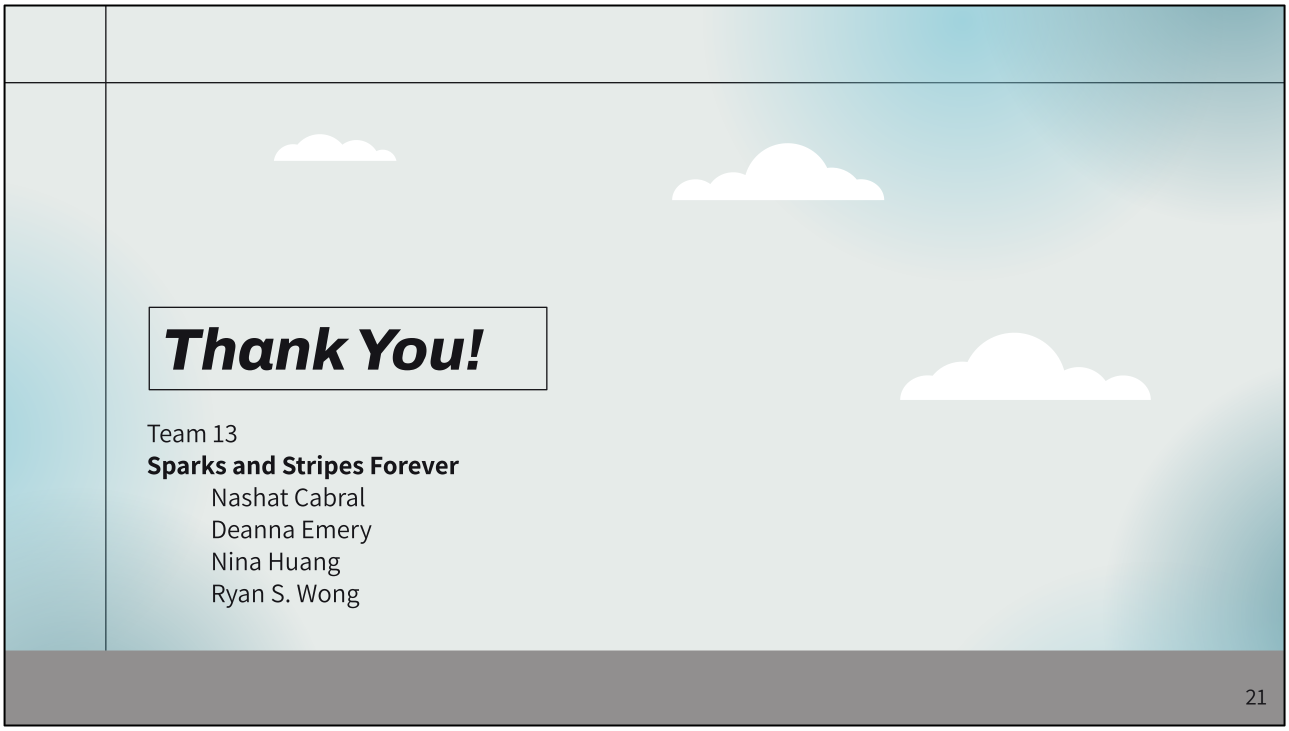
Nina (11-13sec)

With those mentioned, what is next ?

We'll begin Hyperparameter tuning to get optimal parameters for our models.

We will build Support Vector Machine model, which works well with classification tasks

And lastly, we will continuously improve our model performance through an iterative approach by looking at features, models, and ensembling techniques



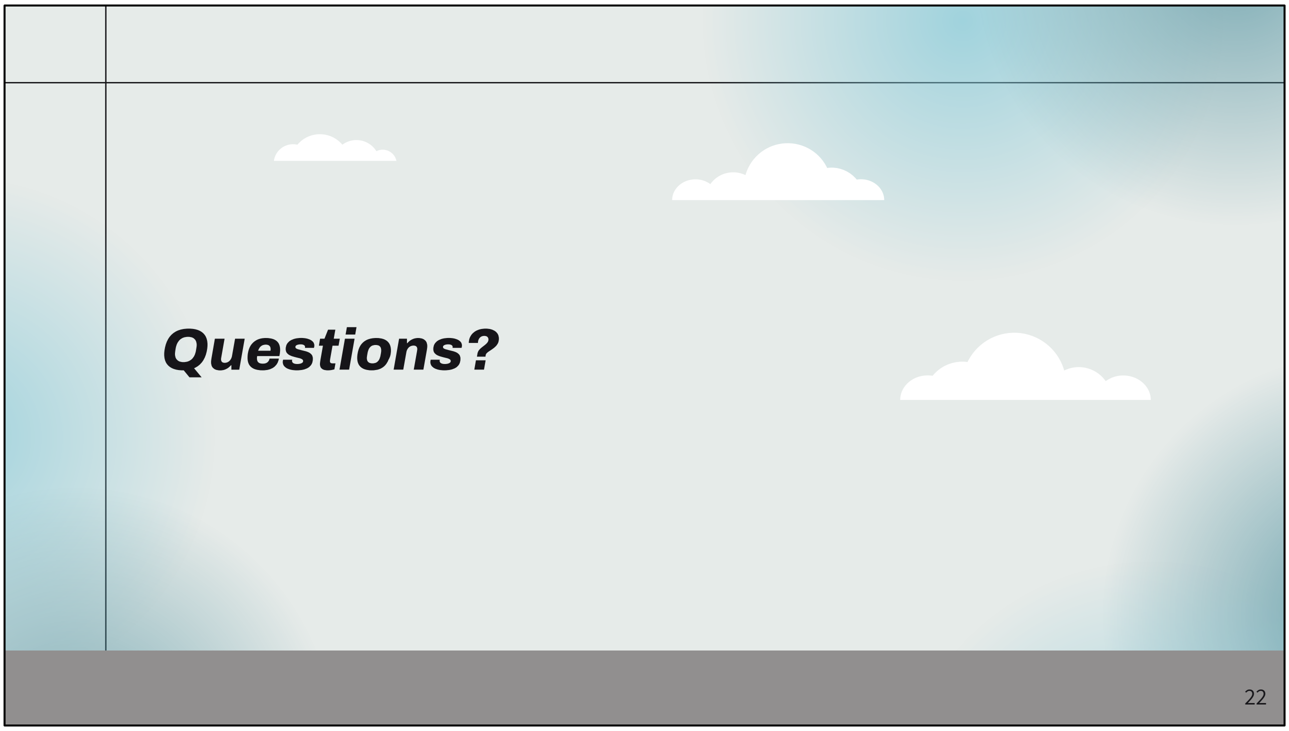
Thank You!

Team 13

Sparks and Stripes Forever

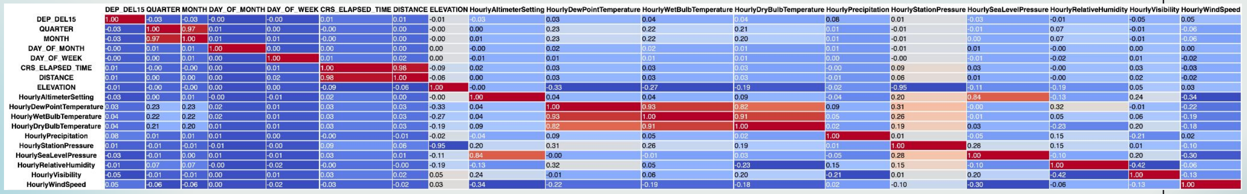
Nashat Cabral
Deanna Emery
Nina Huang
Ryan S. Wong

Thank you all for your time.

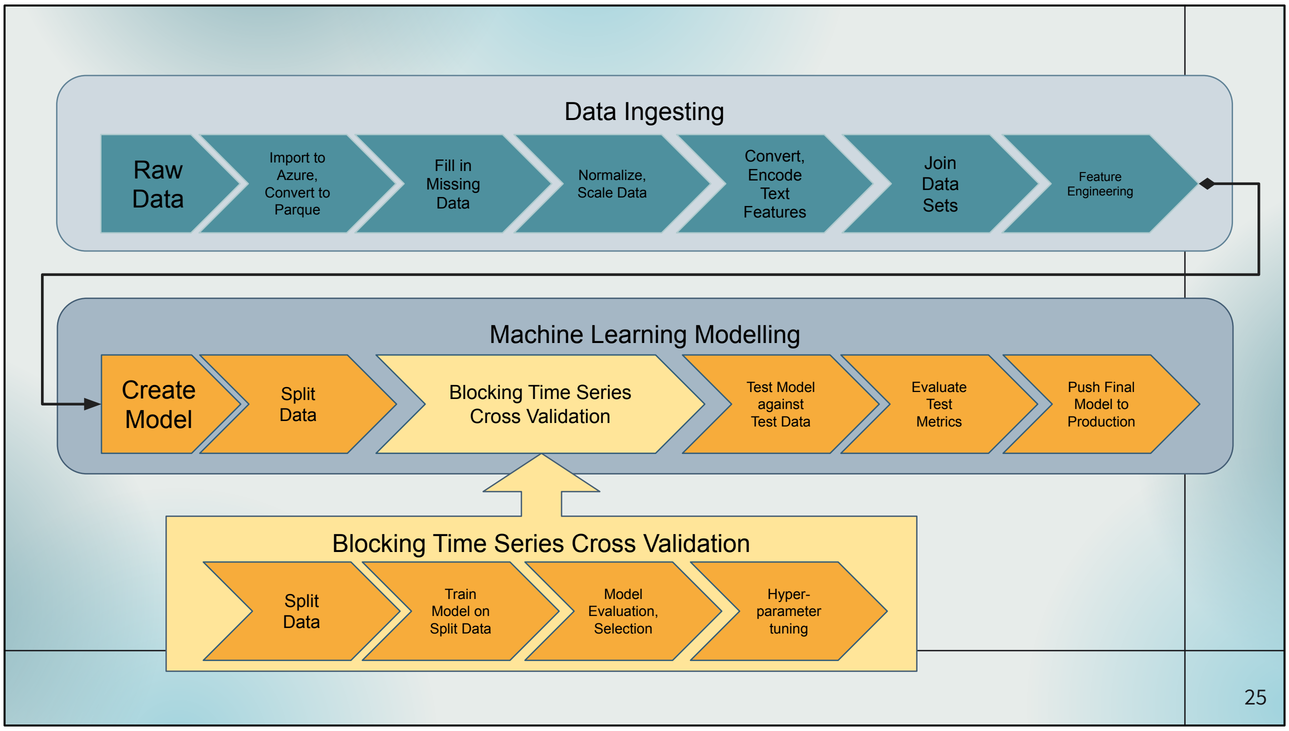


Questions?

Are there any questions?



	DEP_DEL15	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ELAPSED_TIME	DISTANCE	ELEVATION	HourlyAltimeterSetting	HourlyDewPointTemperature	HourlyWetBulbTemperature	HourlyDryBulbTemperature	HourlyPrecipitation	HourlyStationPressure	HourlySeaLevelPressure	HourlyRelativeHumidity	HourlyVisibility	HourlyWindSpeed
DEP_DEL15	1.00	-0.03	-0.03	-0.00	-0.00	0.01	0.01	-0.01	-0.03	0.03	0.04	0.04	0.08	0.01	-0.03	-0.01	-0.05	0.05
QUARTER	-0.03	1.00	0.97	0.01	0.00	-0.00	0.00	-0.00	0.00	0.23	0.22	0.21	0.01	-0.01	-0.01	0.07	-0.01	-0.06
MONTH	-0.03	0.97	1.00	0.01	0.01	-0.00	0.00	-0.00	0.01	0.23	0.22	0.20	0.01	-0.01	0.00	0.07	-0.01	-0.06
DAY_OF_MONTH	-0.00	0.01	0.01	1.00	0.00	0.00	0.00	0.00	-0.00	0.02	0.02	0.01	0.01	-0.01	0.01	-0.00	0.00	0.00
DAY_OF_WEEK	-0.00	0.00	0.01	0.00	1.00	0.01	0.02	0.00	-0.01	0.01	0.01	0.01	0.00	-0.00	-0.01	-0.02	0.01	-0.02
CRS_ELAPSED_TIME	0.01	-0.00	-0.00	0.00	0.01	1.00	0.98	-0.09	0.02	0.03	0.03	0.03	-0.00	0.09	0.03	-0.00	0.03	-0.03
DISTANCE	0.01	0.00	0.00	0.00	0.02	0.98	1.00	-0.06	0.00	0.03	0.03	0.03	-0.01	0.06	0.01	-0.00	0.02	-0.02
ELEVATION	-0.01	-0.00	-0.00	0.00	0.00	-0.09	-0.06	1.00	-0.00	-0.33	-0.27	-0.19	-0.02	-0.95	-0.11	-0.19	0.05	0.03
HourlyAltimeterSetting	-0.03	0.00	0.01	-0.00	-0.01	0.02	0.00	-0.00	1.00	0.04	0.04	0.09	-0.04	0.20	0.84	-0.13	0.24	-0.34
HourlyDewPointTemperature	0.03	0.23	0.23	0.02	0.01	0.03	0.03	-0.33	0.04	1.00	0.93	0.82	0.09	0.31	-0.00	0.32	-0.01	-0.22
HourlyWetBulbTemperature	0.04	0.22	0.22	0.02	0.01	0.03	0.03	-0.27	0.04	0.93	1.00	0.91	0.05	0.26	-0.01	0.05	0.06	-0.19
HourlyDryBulbTemperature	0.04	0.21	0.20	0.01	0.01	0.03	0.03	-0.19	0.09	0.82	0.91	1.00	0.02	0.19	0.03	-0.23	0.20	-0.18
HourlyPrecipitation	0.08	0.01	0.01	0.01	0.00	-0.00	-0.01	-0.02	-0.04	0.09	0.05	0.02	1.00	0.01	-0.05	0.15	-0.21	0.02
HourlyStationPressure	0.01	-0.01	-0.01	-0.01	-0.00	0.09	0.06	-0.95	0.20	0.31	0.26	0.19	0.01	1.00	0.28	0.15	0.01	-0.10
HourlySeaLevelPressure	-0.03	-0.01	0.00	0.01	-0.01	0.03	0.01	-0.11	0.84	-0.00	-0.01	0.03	-0.05	0.28	1.00	-0.10	0.20	-0.30
HourlyRelativeHumidity	-0.01	0.07	0.07	-0.00	-0.02	-0.00	-0.00	-0.19	-0.13	0.32	0.05	-0.23	0.15	0.15	-0.10	1.00	-0.42	-0.06
HourlyVisibility	-0.05	-0.01	-0.01	0.00	0.01	0.03	0.02	0.05	0.24	-0.01	0.06	0.20	-0.21	0.01	0.20	-0.42	1.00	-0.13
HourlyWindSpeed	0.05	-0.06	-0.06	0.00	-0.02	-0.03	-0.02	0.03	-0.34	-0.22	-0.19	-0.18	0.02	-0.10	-0.30	-0.06	-0.13	1.00


Data Ingesting
Raw Data
Import to Azure, Convert to Parque
Fill in Missing Data
Normalize, Scale Data
Convert, Encode Text Features
Join Data Sets
Feature Engineering
Machine Learning Modelling
Create Model
Split Data
Blocking Time Series Cross Validation
Test Model against Test Data
Evaluate Test Metrics
Push Final Model to Production
Blocking Time Series Cross Validation
Split Data
Train Model on Split Data
Model Evaluation, Selection
Hyper-parameter tuning