

## Real-time ML Journey MicrosoftML + Revoscale

#### Hiram Fleitas

Senior Customer Engineer Microsoft Data & Al

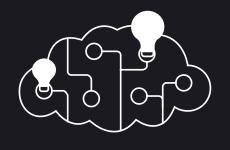
<u>badges</u>

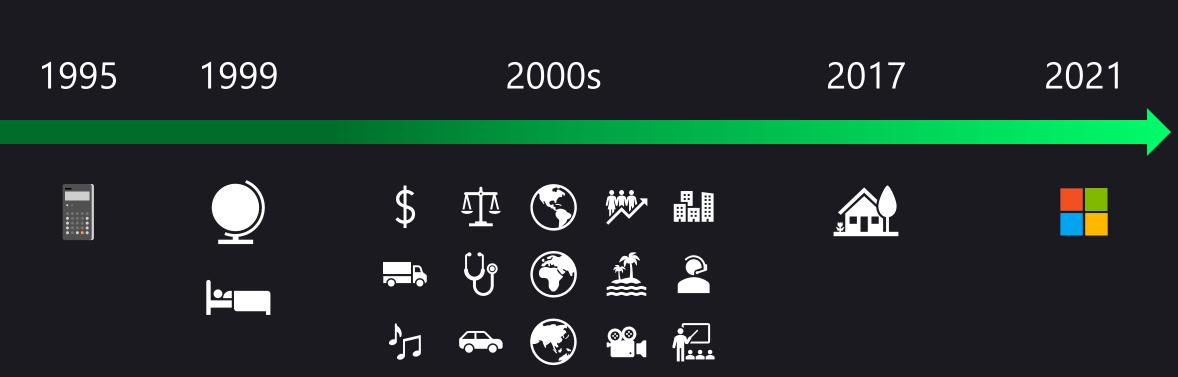


### About Me

aka.ms/hiram

- 1. Machine Learning
- 2. Synapse Analytics
- 3. Power Platform
- 4. DevOps





Legal, Medical, Retail, Travel Insurance, ISV, HR, Cruise Line, PnC Insurance, plus more...

































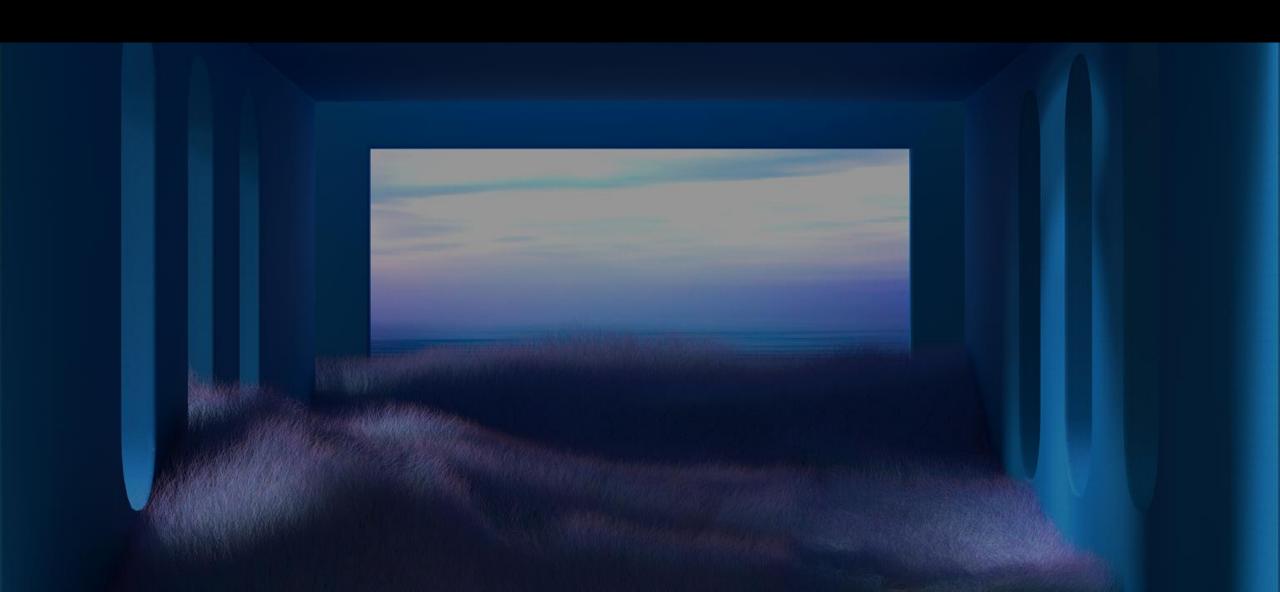


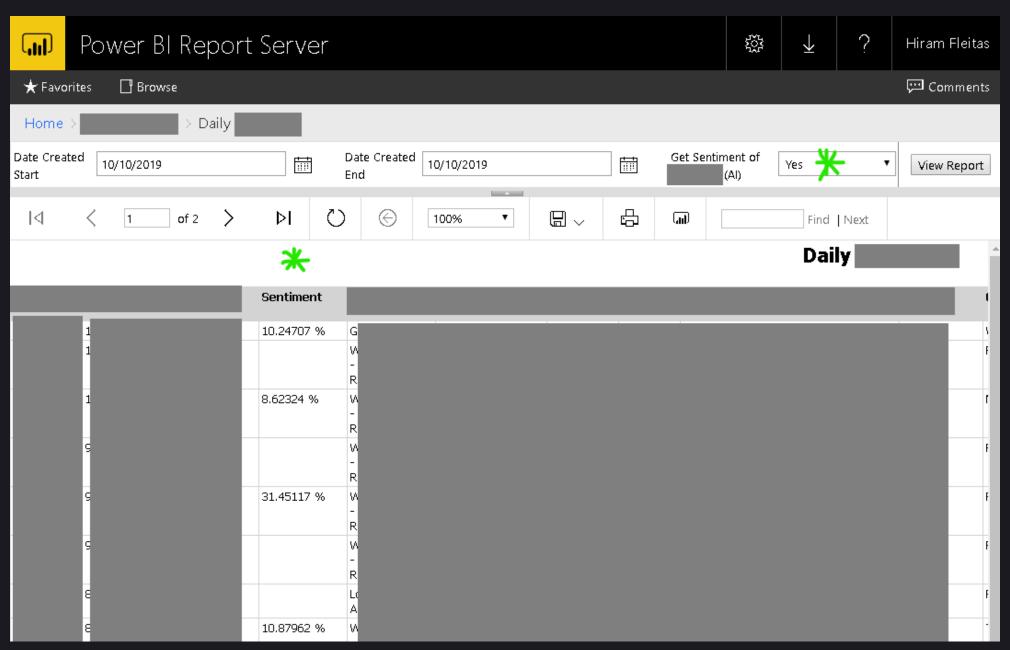
### Agenda

with impact

- 1. Sentiment \$30M
- 2. QnA Chat Bots \$200K
- 3. Forms \$2M
- 4. Capital Modeling \$500M
- 5. Length of Stay \$40M
- 6. Search by Photo \$200K

## Sentiment \$30M





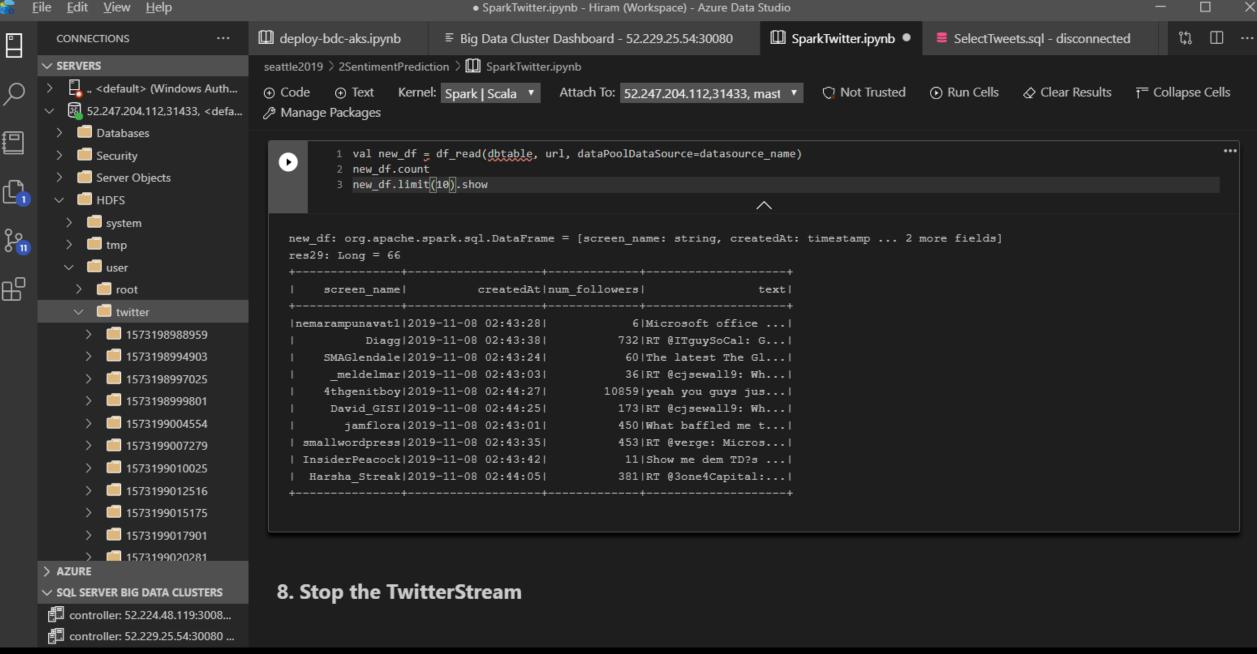
```
create or alter proc GetCognitiveAPIQuoteSentiment
as
    set nocount on;
    declare @py nvarchar(max);
    set @py = N'import requests, pprint as pr
from pandas.io.json import json normalize
apikey = "mykey"
api = "https://eastus2.api.cognitive.microsoft.com/text/analytics/v2.0/"
url = api + "sentiment"
df = jsondocs
headers = {
     "Ocp-Apim-Subscription-Key": apikey,
     "content-type": "application/json"
response = requests.post(
     url,
     headers = headers,
     data = df.iloc[0][0].encode()
rds = response.json()
df2 = json normalize(rds, "documents")
pr.pprint(rds)
print(type(df2),df2,sep="\n")
```

Learn more:

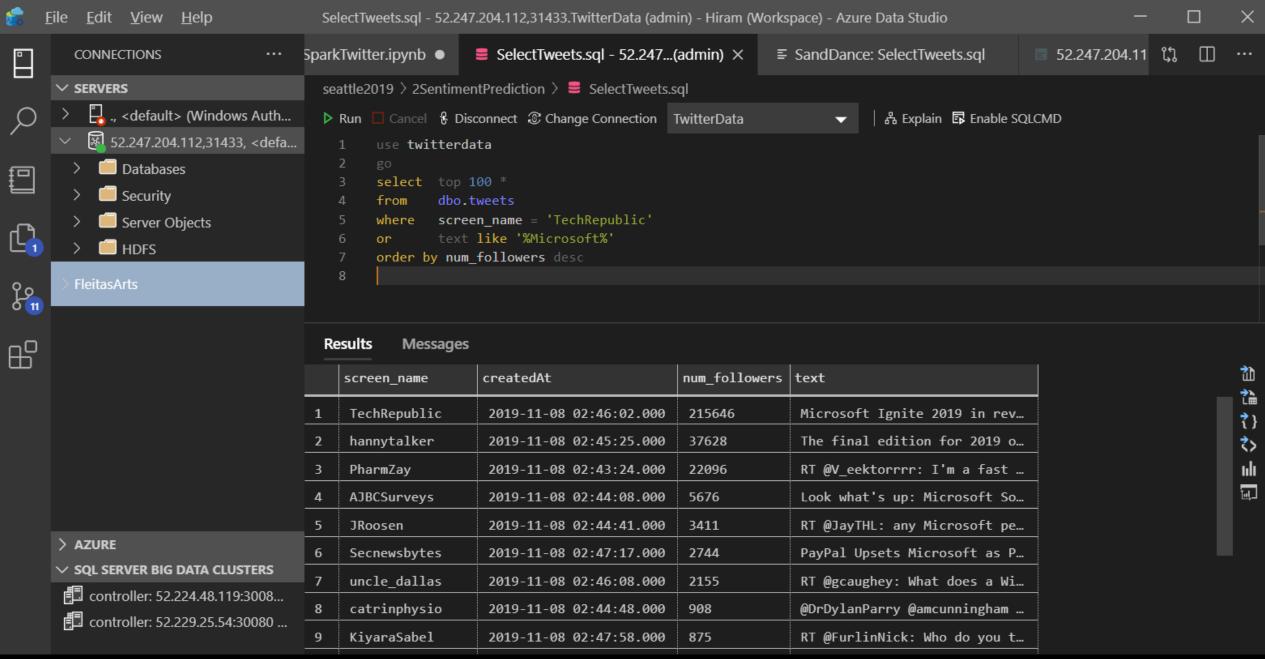
github.com/hfleitas/seattle2019/blob/master/2SentimentPrediction/Trollhunters.ipynb

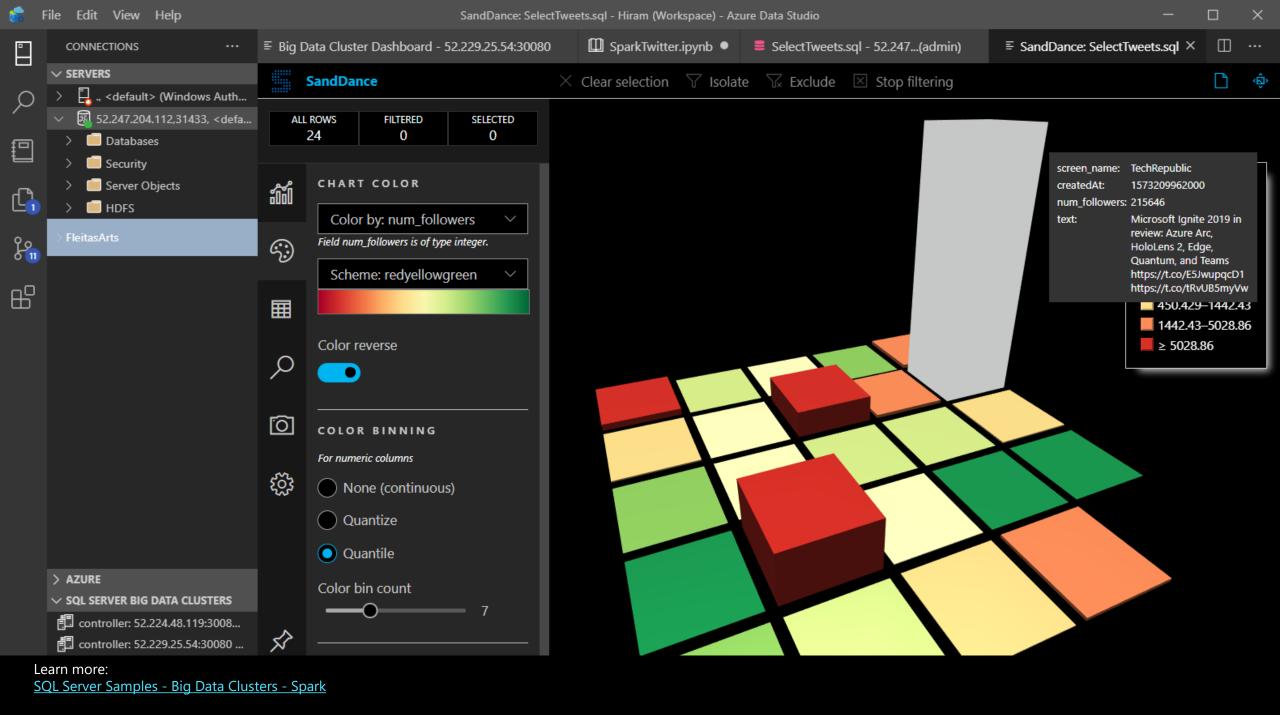
```
drop table if exists apiresults;
    create table apiresults (id int, score float);
    insert into apiresults
    exec sp_execute_external_script @language = N'Python'
       ,@script = @py
        "@input data 1 = N'select * from JsonQuotes'
        "@input data 1 name = N'jsondocs'
        ,@output data 1 name = N'df2'
    select * from apiresults;
   update q
       set q.Sentiment = a.Score
    from
           Quotes q
    inner join apiresults a
       on q.quoteid = a.id
    where q.Sentiment is null;
go
```

Subject: Re: Developer API - Reviews Text Truncated
JUL 31, 2018   02:49PM PDT Hi Hiram,
Thanks for writing in. The API does not return full-text reviews at this point; currently, the API will return 3 review snippets (the reviews that are currently displayed at the top when sorted by Yelp Sort, which is determined by recency, user voting, and other review quality factors to help consumers make informed decisions) of 160 characters. The Yelp API cannot be configured to return alternative review excerpts.
As a reminder, all API integrations must be consumer-facing (i.e. not for B2B dashboards or analytics) and must abide by our Terms of Use ( <a href="https://www.velp.com/developers/api_terms">https://www.velp.com/developers/api_terms</a> ) and Display Requirements ( <a href="https://www.velp.com/developers/display">https://www.velp.com/developers/display</a> requirements). Lastly, all API responses cannot downloaded, stored, cached, or prefetched for more than 24 hours (except for the business ID for backend matching purposes) and scraping Yelp via any means is strictly prohibited.
Please let me know if you have other questions.
Thanks!

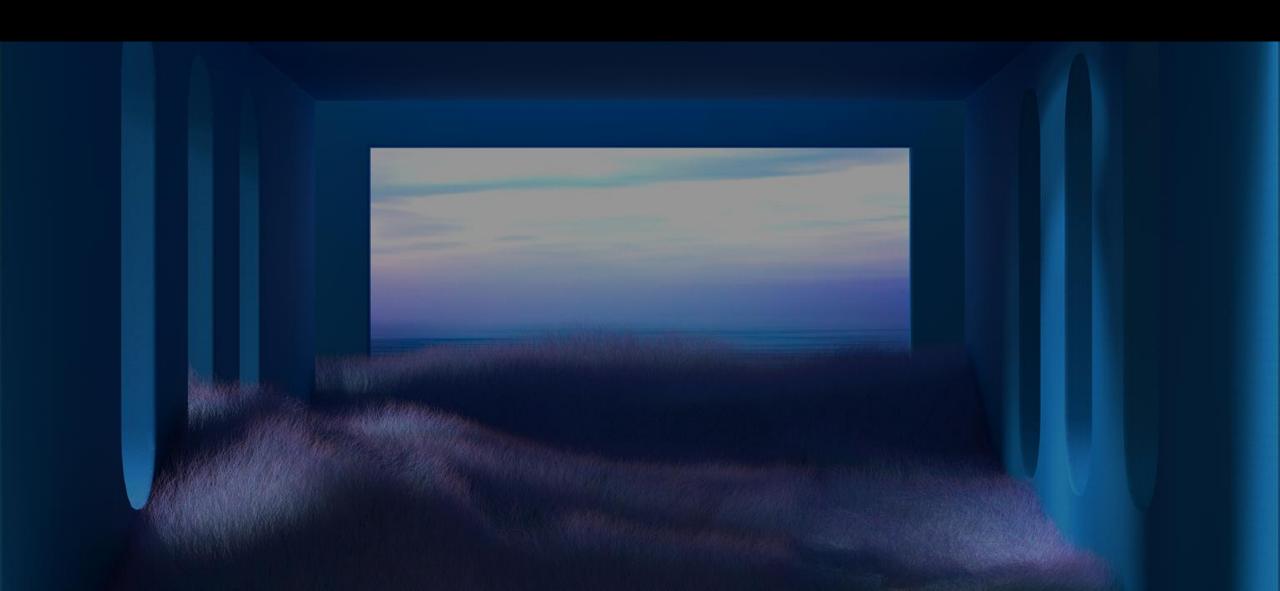


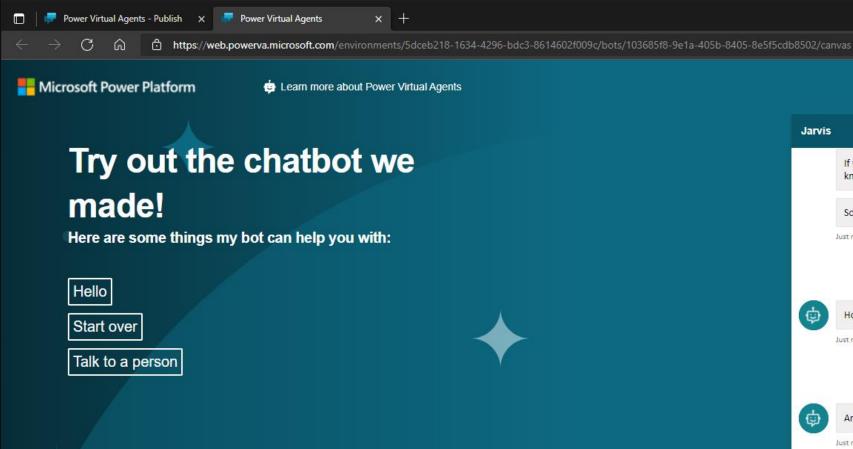
Learn more:

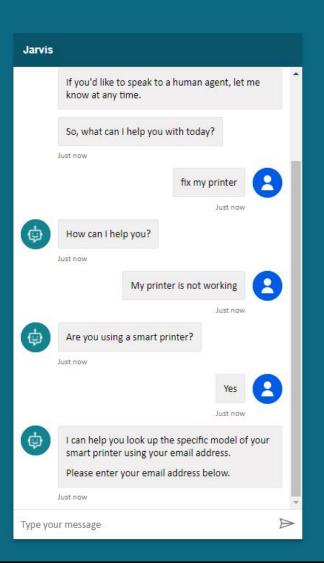




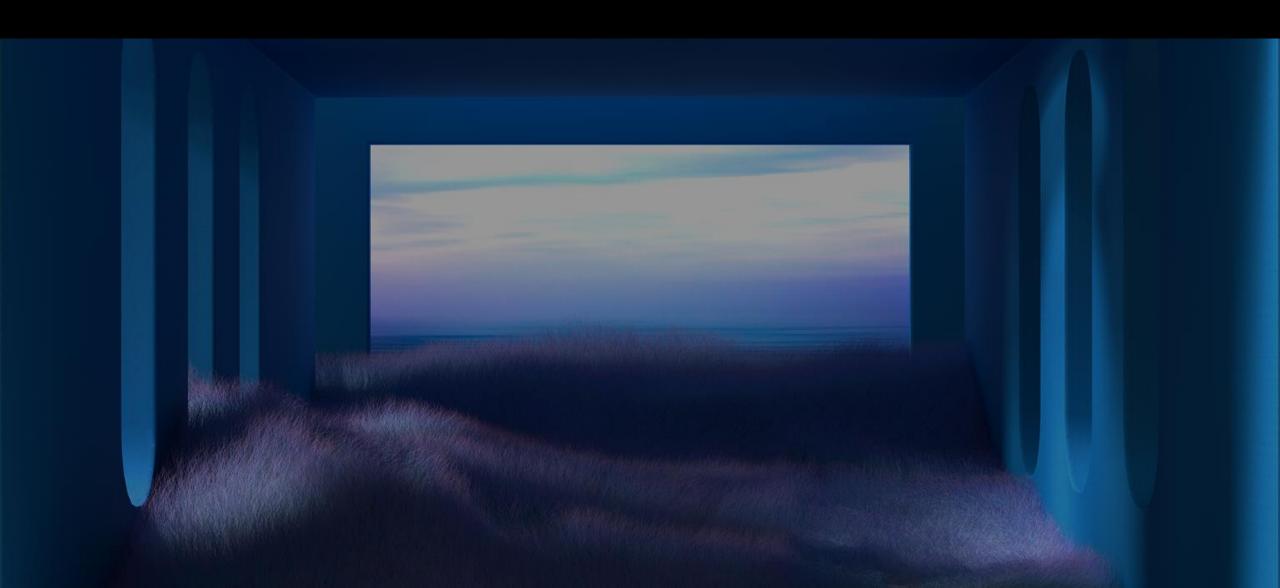
### QnA Chat Bots \$200K

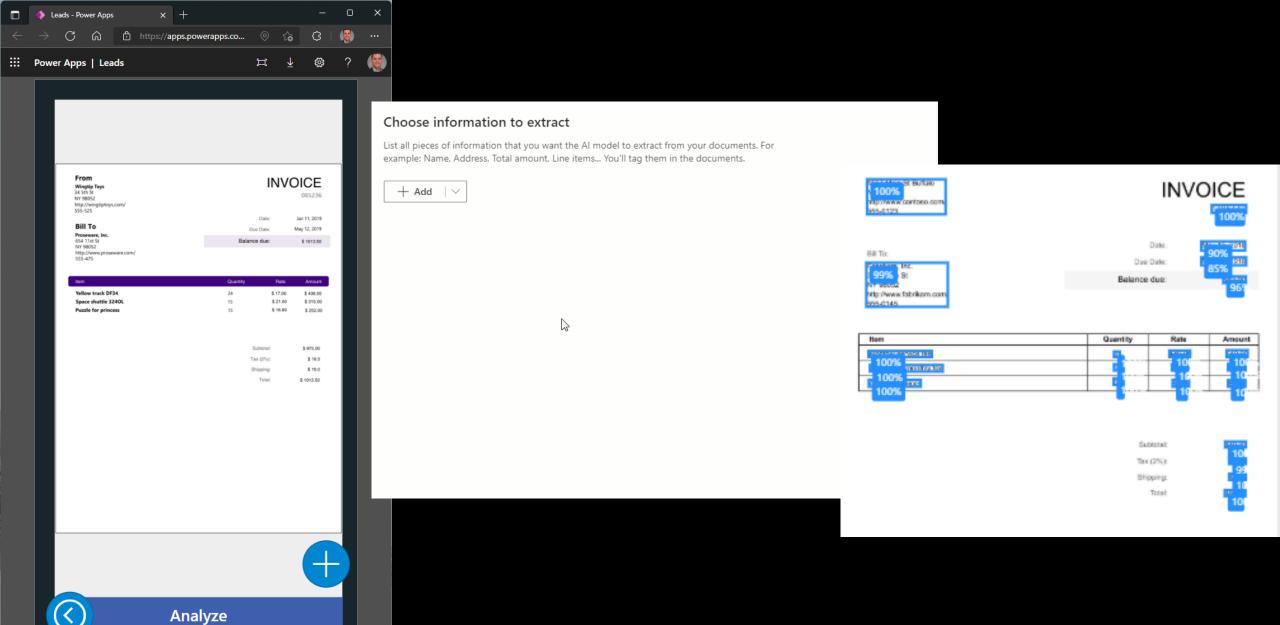




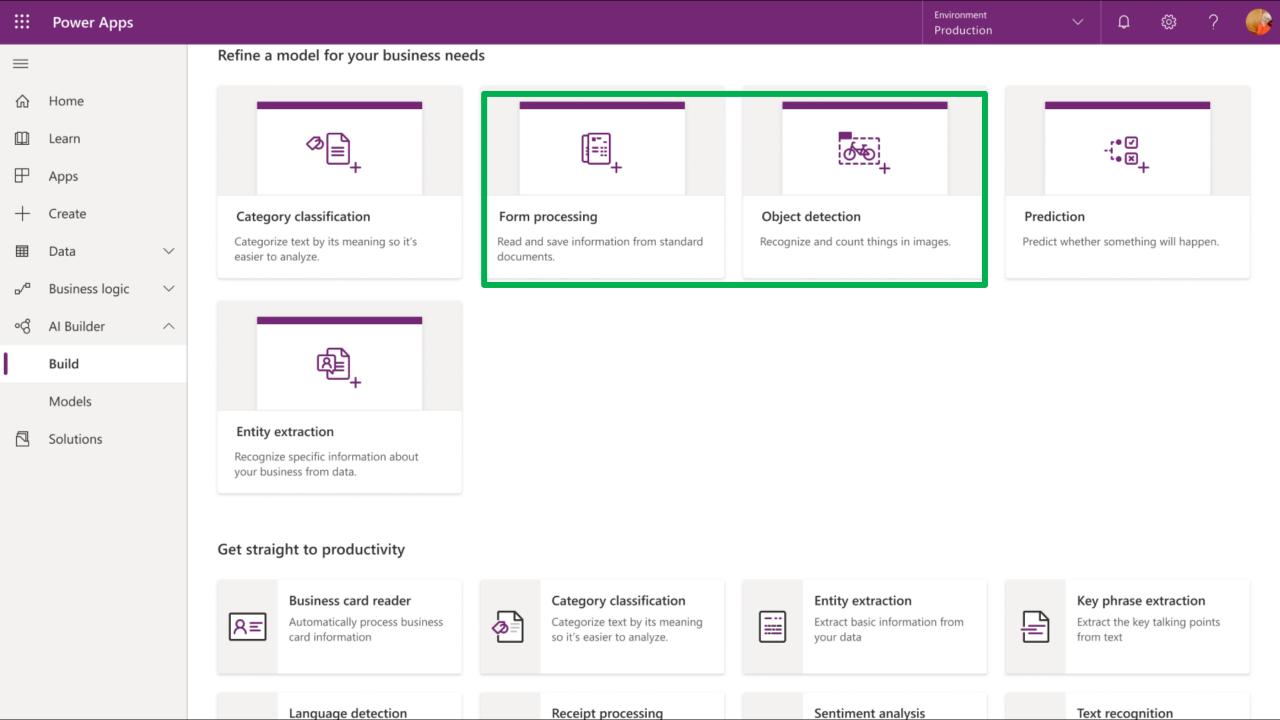


## Forms \$2M

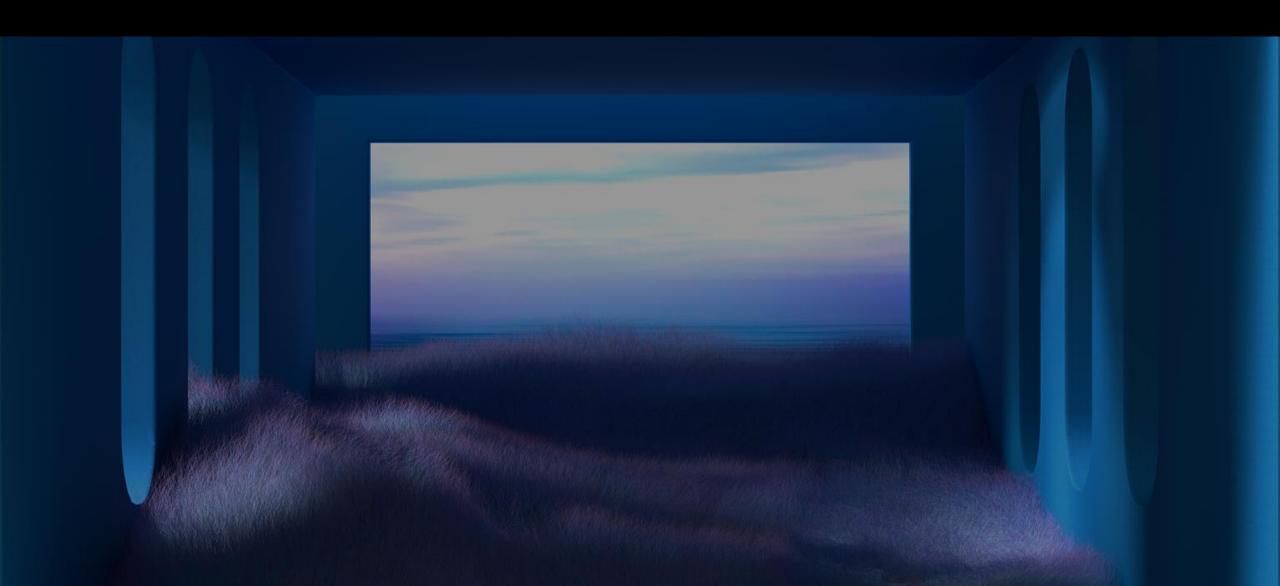




Learn more: Form Processing - Create first model



## Capital Modeling \$500M



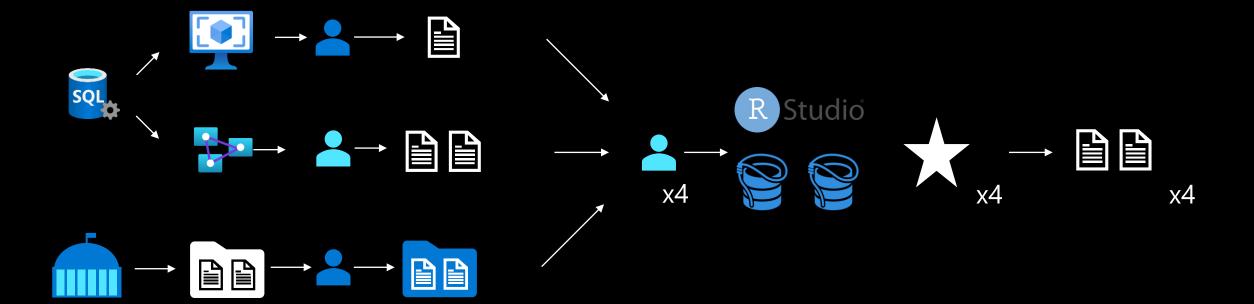
#### Before

- 7 User touchpoints
- · 4 Unique manual processes
- Many different files and folders
- Significant user downtime

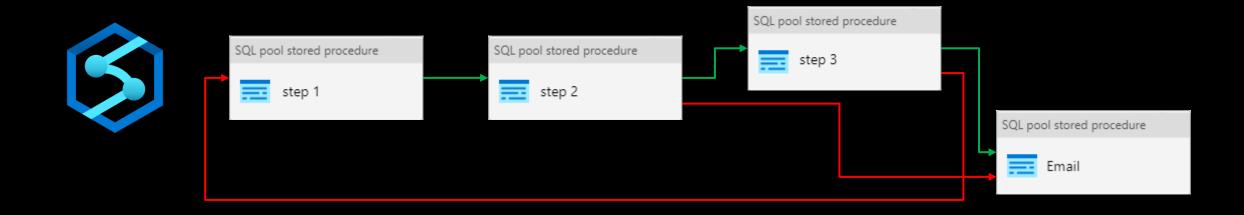
#### After

- 1 Single pipeline
- · 3 Primary stored procedures
- 1 SQL Server
- · Minimal user downtime

Before



After



#### Before

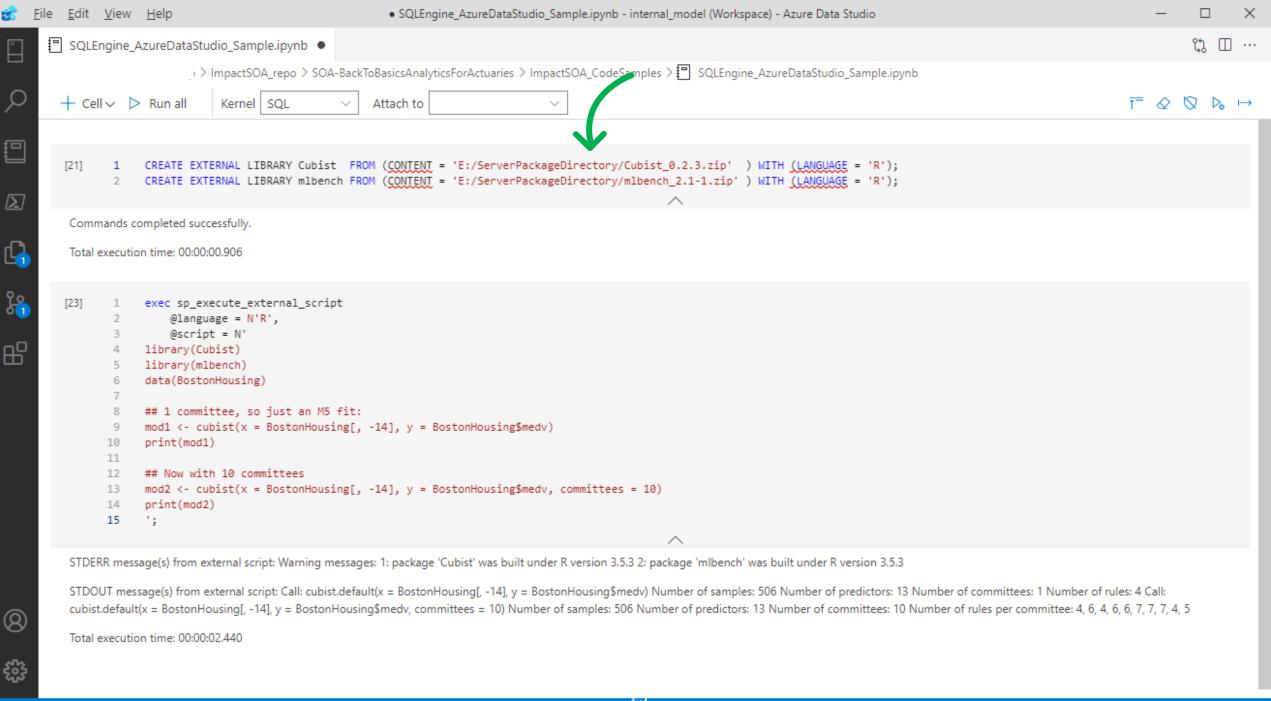
- Loop based logic
- WET (Write Everything Twice)
- Inefficient
- Frequent R troubleshooting

```
| If(sum(cashcedsd,[j])=0)=0|
| filk-cobits(y-cashcedsd,[j])=0|
| scencashPredict[,j]<-cound(predict(fitA,newdata-scenwodvars),6)
| size {
| scencashAredict[,j]<-cound(predict(fitA,newdata-scenwodvars),6)
| size {
| scencashAredict[,j]<-cound(predict(fitA,newdata-scenwodvars),6)
| size {
| scencashAredict[,j]<-cound(predict(fitA,newdata-scenwodvars),6)
| size {
| scencashCredict[,j]<-cound(predict(fitA,newdata-scenwodvars),6)
| size {
| scencashCredict[,j]<-cound(predict(fitD,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitD,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitD,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitD,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitG,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitG,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitG,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict[,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars,6),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars,6),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars,6),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars,6),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars,6),6)
| scencashCredict(j,j]<-cound(predict(fitIntxp,newdata-scenwodvars,6),6)
|
```

#### After

- Functional programming
- DRY (Don't Repeat Yourself)
- · Little need for R troubleshooting

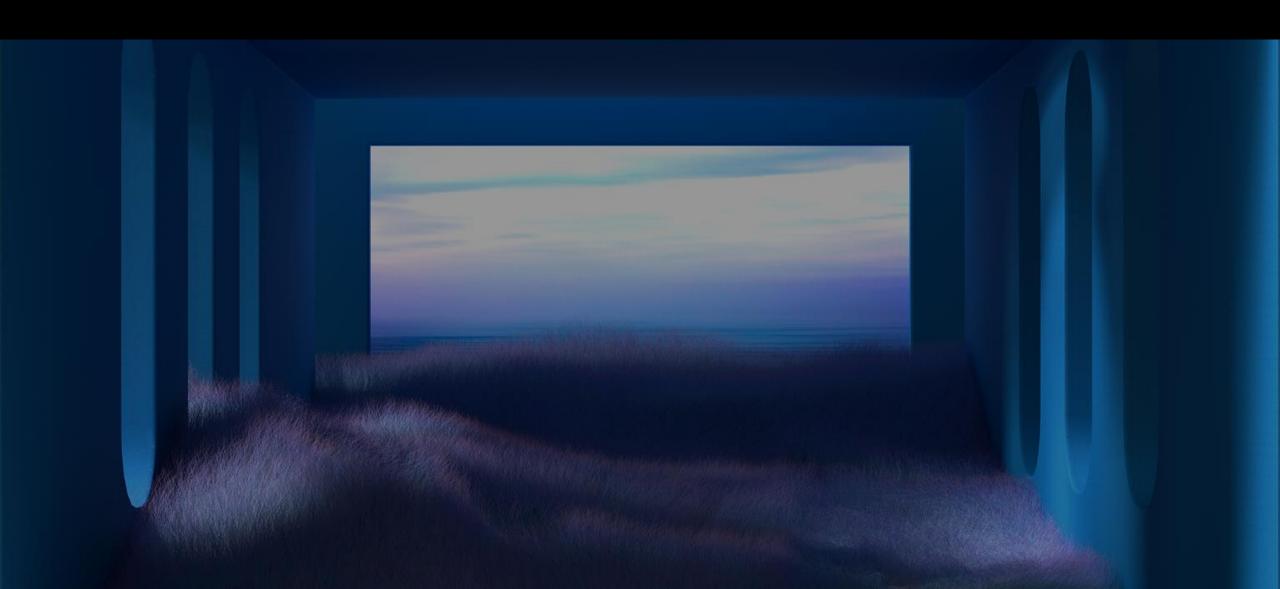
```
4 trainCubist <- function(dt, com = 15, extp = 90){Cubist::cubist(y = getY(dt),x = getX(dt),committees = getCom(dt),control =
5 predictCubist <- function(aModel, dt){stats::predict(aModel, newdata = getX(dt))}</pre>
```

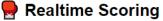


### How to get started? How to reproduce?

- 1. Identify manual data processes & transformations of high value (ROI).
- 2. Simplify by consolidating workloads (leverage cloud).
- 3. Democratize understanding by sharing knowledge.
- 4. Accelerate business value for down-stream consumers/users.
- 5. <u>aka.ms/mlsqldev</u>
- 6. <u>aka.ms/adf/azurelearn</u>
- 7. RStudio Cheatsheets RStudio

## Length of Stay \$40M





#### Proc (errors), Train, Score

```
In [3]: -- My simple proc to serialize the model bin, cause train model real time scoring errors.
        Use Hospital Py
        create or alter proc [GetRTSModelRF]
        declare @info varbinary(max);
        select @info = info from dbo.ColInfo;
        declare @info varbinary(max);
        select @info = info from dbo.ColInfo;
        exec sp_execute_external_script @language = N'Python', @script = N'
        import dill
        from numpy import sqrt
        from pandas import DataFrame
        from revoscalepy import rx_set_compute_context, RxSqlServerData, rx_dforest, RxOdbcData, rx_serialize_model, rx_write_object, RxLocalSeq
        from microsoftml import adadelta optimizer
        connection_string = "Driver=SQL Server;Server=localhost;Database=Hospital_Py;Trusted_Connection=true;"
        column_info = dill.loads(info)
                Set training dataset, set features and types.
        variables_all = [var for var in column_info]
        #variables_to_remove = ["eid", "vdate", "discharged", "facid"]
        variables to remove = ["ClaimClaimID", "ClaimDateClosed", "ClaimReportedDate"]
        training variables = [x for x in variables all if x not in variables to remove]
        LoS Train = RXSqlServerData(sql query = "SELECT ClaimClaimID, {} FROM LoS WHERE ClaimClaimID IN (SELECT ClaimClaimID from Train Id)".form
        at(", ".join(training_variables)),
                                    connection string = connection string,
                                    column_info = column_info)
                Specify the variables to keep for the training
        #variables to remove = ["eid", "vdate", "discharged", "facid", "lengthofstay"]
        variables to remove = ["ClaimClaimID", "ClaimDateClosed", "ClaimReportedDate", "lengthofstay"]
        training_variables = [x for x in variables_all if x not in variables_to_remove]
        formula = "lengthofstay ~ " + " + ".join(training_variables)
        ## Train RF Model
        dest = RxOdbcData(connection string, table = "RTS")
        model = rx dforest(formula=formula,
                            data=LoS_Train,
                            n_tree=40,
                            cp=0.00005,
                            min split=int(sqrt(70000)),
                            max_num_bins=int(sqrt(70000)),
                            seed=5)
        serialized model = rx serialize model(model, realtime scoring only = True)
        rx_write_object(dest, key_name="id", key="RF", value_name="value", value=serialized_model, serialize=False, compress=None, overwrite=False
        , @params = N'@info varbinary(max)'
        , @info = @info;
        GO
```

```
exec GetRTSModelRF;

-- Dev server: Total execution time: 00:20:17.325

In [8]: select id, (datalength(value)/1024)/1024 as MB from RTS

(1 row affected)

Total execution time: 00:00:00.010

Out[8]: id MB

RF 0
```

```
--- Real Time Scoring

-- Get the trained model

DECLARE @model_name VARCHAR(3) = 'RF'

DECLARE @model_NARBINARY(max) = (SELECT value FROM [dbo].[RTS] WHERE id = @model_name);

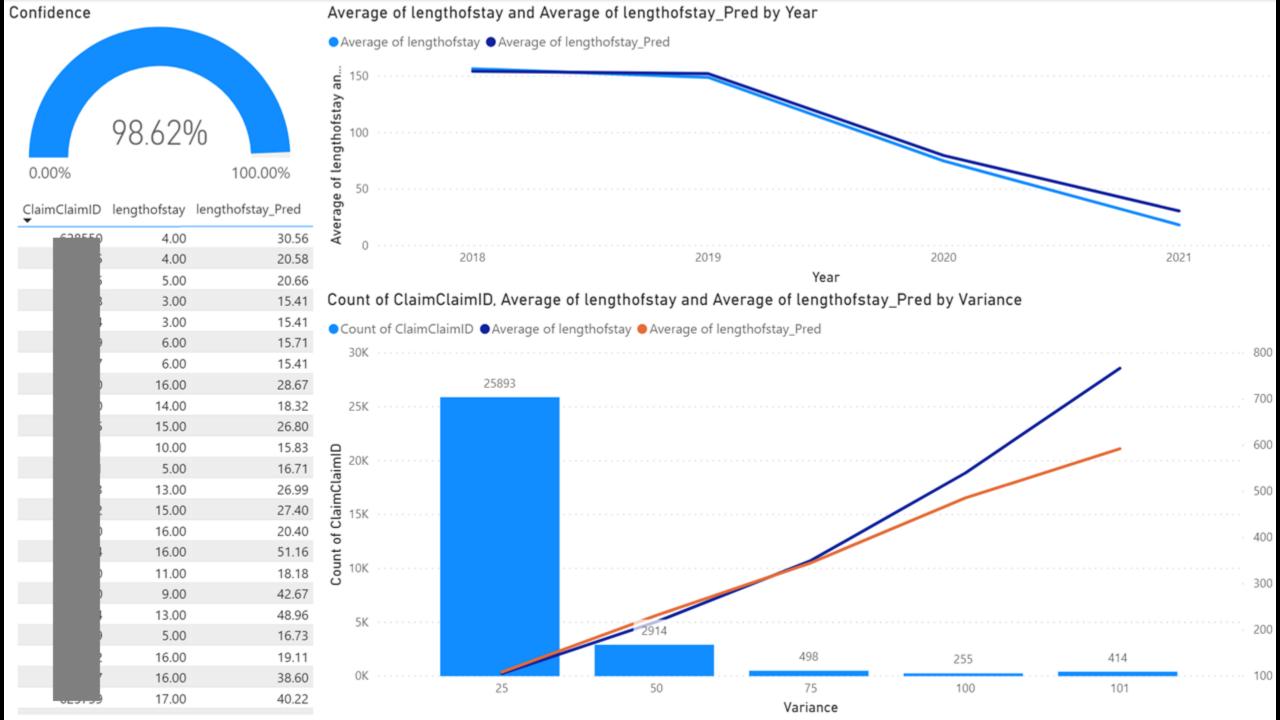
--- Real Time Scoring is meant for small scoring request, which is why we select the top 10 for this example.

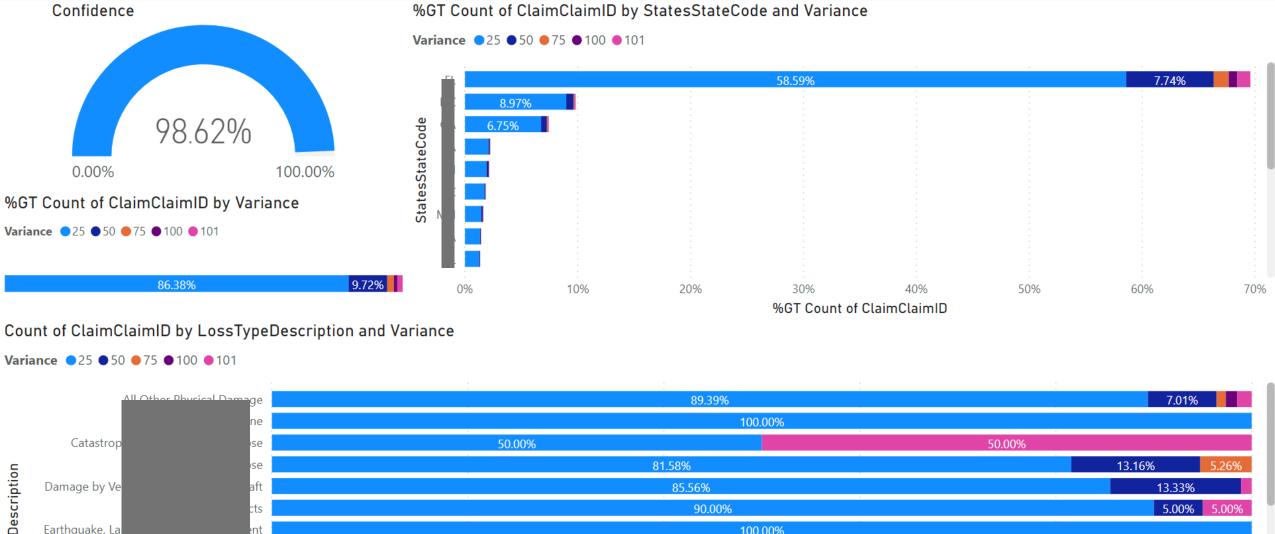
DECLARE @inputData VARCHAR(max);

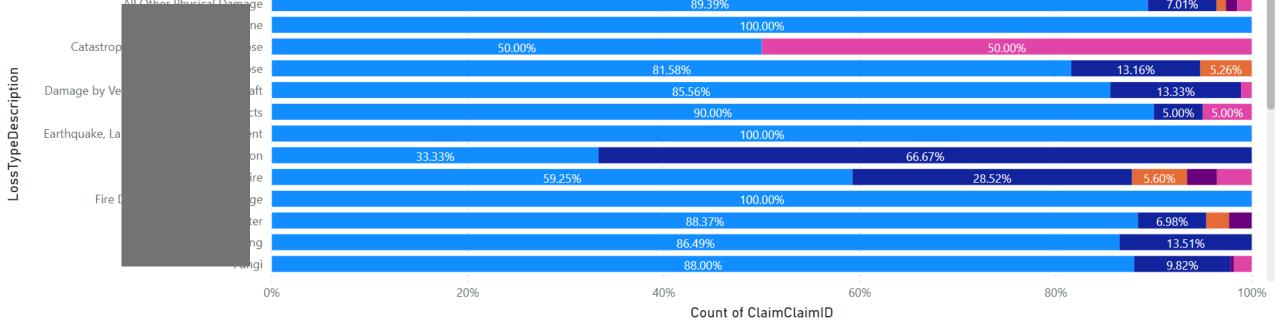
SET @inputData = 'SELECT TOP (10) ClaimClaimID, ClaimClaimStatusID, ClaimStatusDescription, StatesStateCode, LossTypeDescription, PolicyVersia PolicyVersionAttributesOccupancyType, PolicyVersionAttributesCoverageA, ClaimMoneyLosses, ClaimMoneyLAE, ClaimRoomsWithDamage, number_of_issues, lengthofstay

FROM LoS WHERE ClaimClaimID NOT IN (SELECT ClaimClaimID from Train_Id) ORDER BY ClaimClaimID';

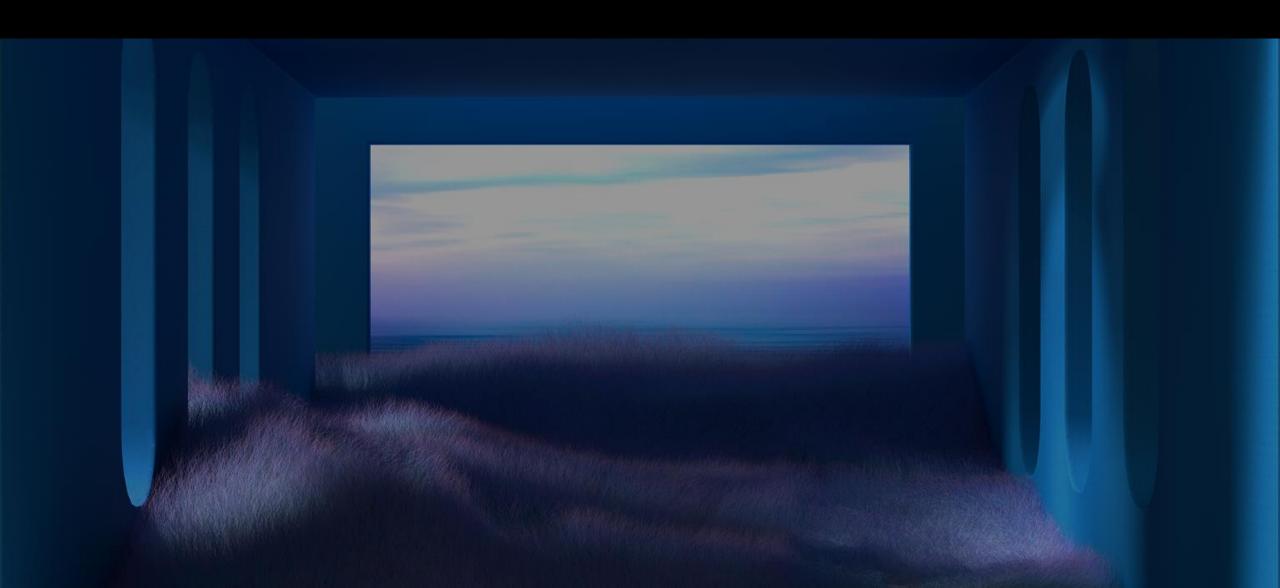
DECLARE @output_table TABLE(lengthofstay_Pred FLOAT);
INSERT @output_table EXEC [dbo].[sp_rxPredict] @model = @model, @inputData = @inputData;
DROP TABLE IF EXISTS RTS_PredictionBak;
exec sp_rename 'RTS_Prediction', 'RTS_PredictionBak';
SELECT * INTO RTS_Prediction FROM @output_table
```

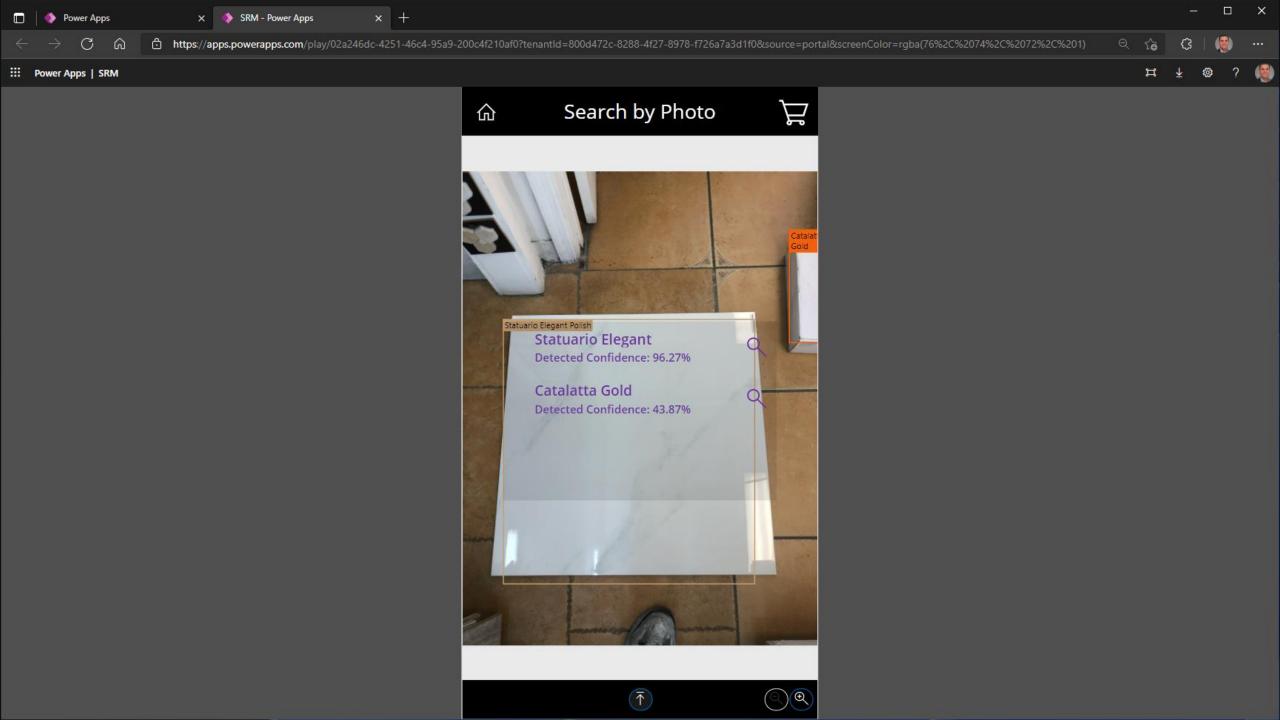


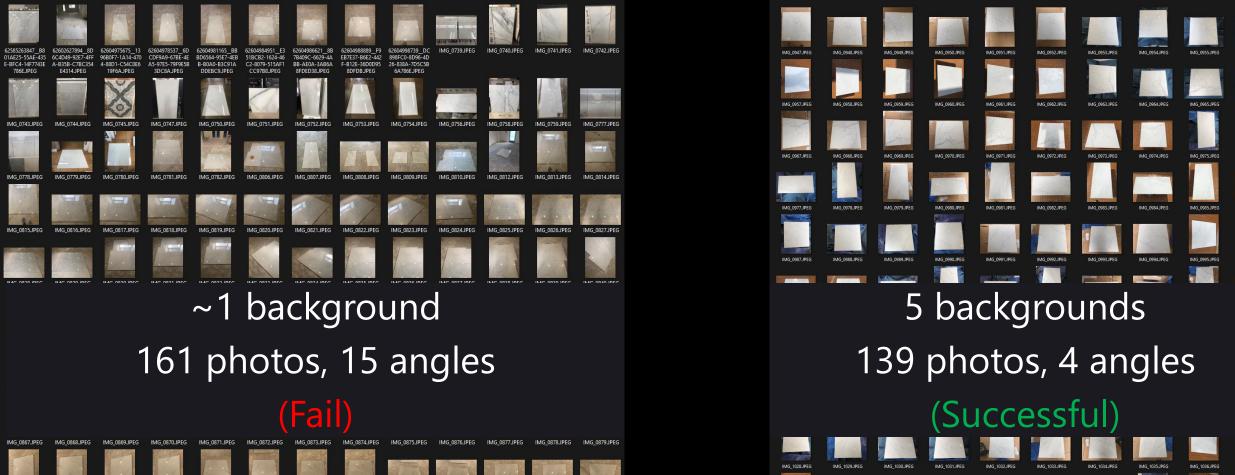


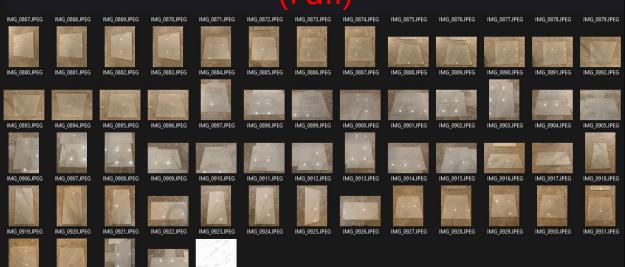


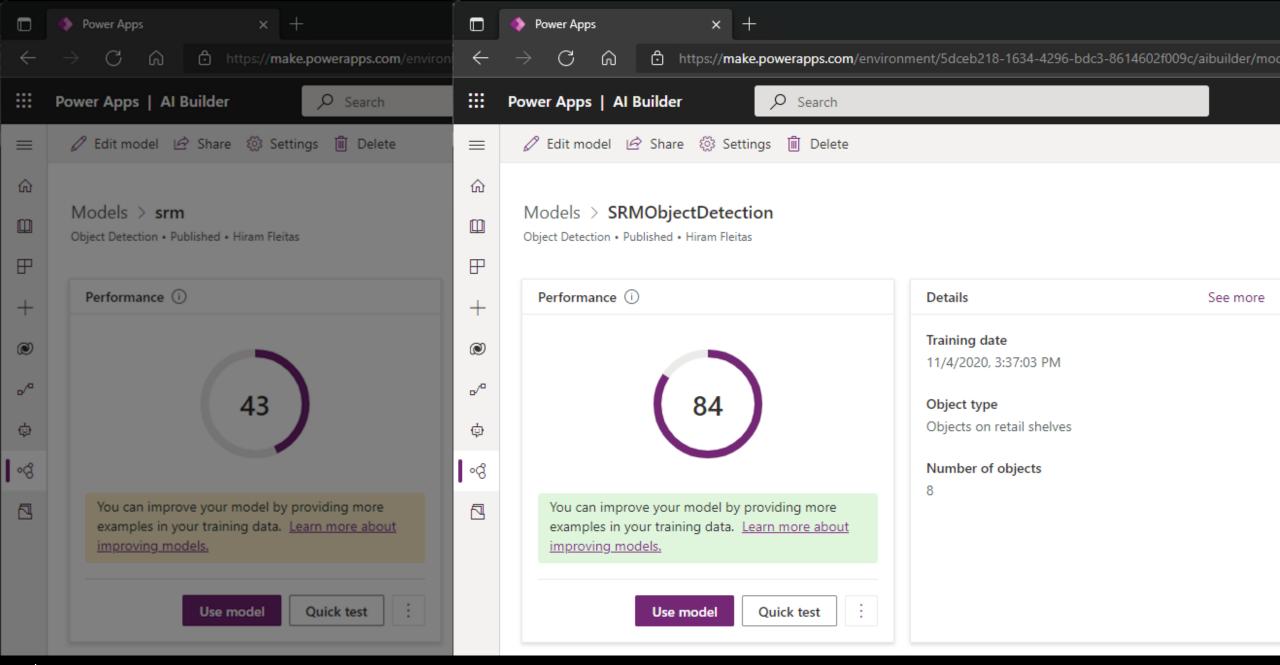
## Search by Photo \$200K











Learn more:

<u>Use more diverse images</u>

<u>Interpret performance score</u>

### Summary

- 1. Sentiment
- 2. QnA Chat Bots
- 3. Forms
- 4. Capital Modeling
- 5. Length of Stay
- 6. Search by Photo

### Additional Resources

- 1. Repo1 (SQL, Python, JS on Node, Power BI)
- Repo2 (SQL BDC, Spark/Scala, Python, Azure Cog API, Paginated Reports)
- 3. Repo3 (Synapse, DevOps, In-DB ML, R, Purview)
- 4. Video (3 minutes)
- 5. <u>Video</u> (1 hour)



## Thank you

Hiram Fleitas

aka.ms/hiram

