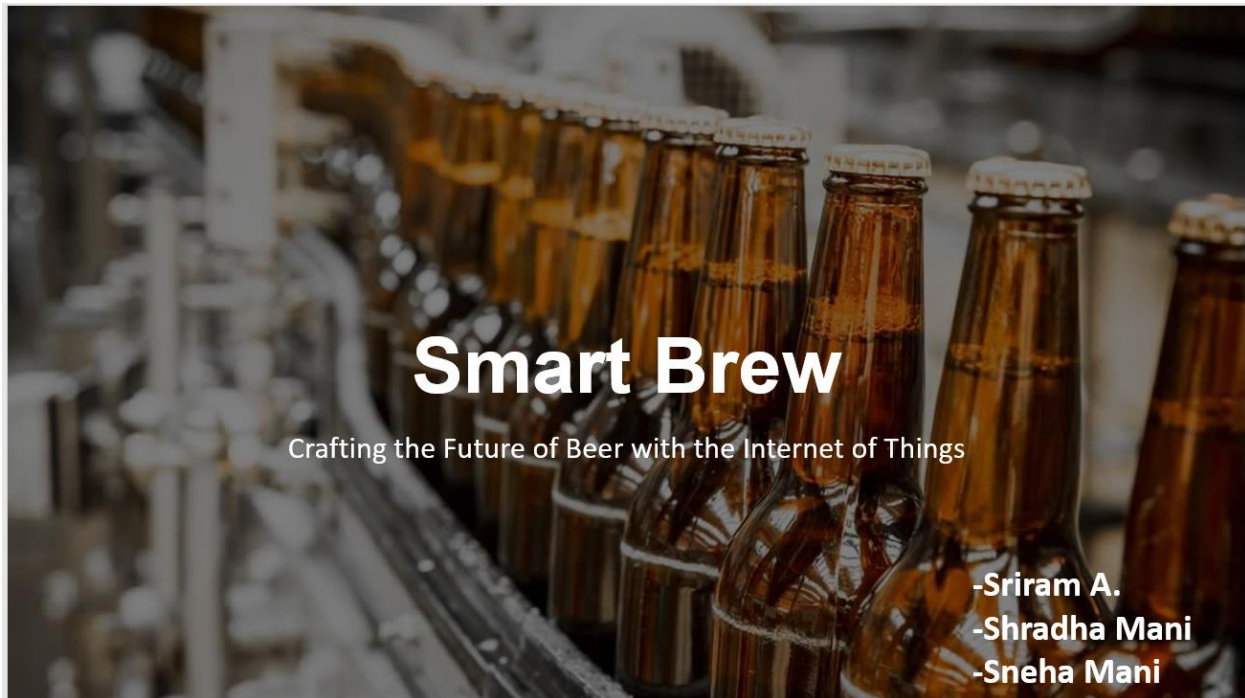


# CIS 8395 Project



## Smart Brew

Crafting the Future of Beer with the Internet of Things

-Sriram A.  
-Shradha Mani  
-Sneha Mani

## Index

1. Project Proposal
2. Architecture Overview
3. Data Source
4. Storage
5. ETL
6. Visualization
7. Product
8. Areas of Impact
9. Challenges Faced
10. Future Scope

## Need for Smart Brew

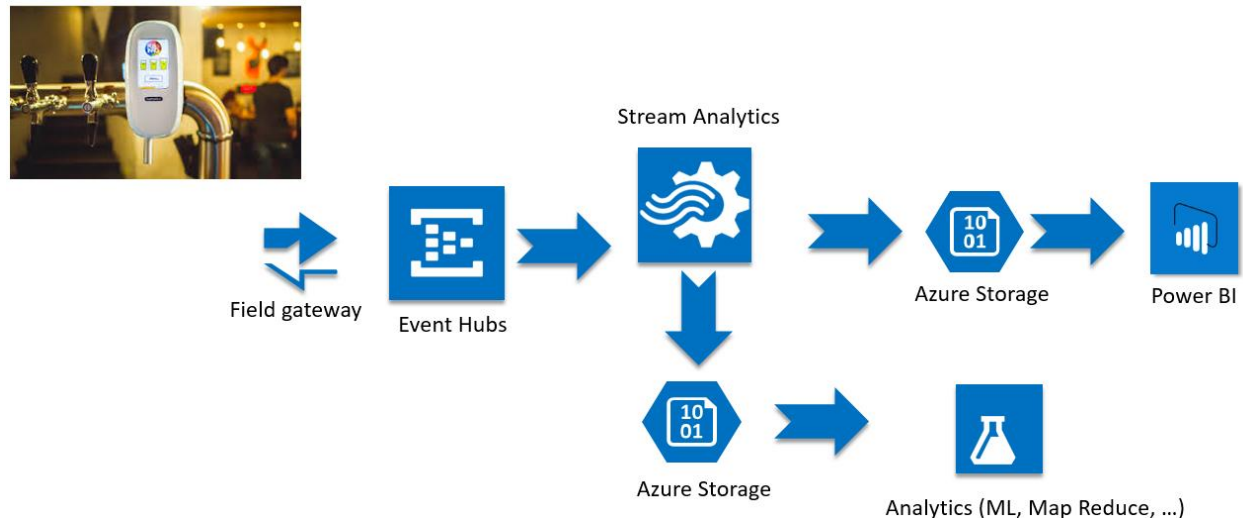
1. The ingredients to create a beer remain the same, however it's subtle shifts in ratios, varieties, temperature and time create that create wildly different results.
2. The temperature can influence the types of flavors the yeast produces and how fast your fermentation can go.
3. In some cases, if you don't vent your tank of CO2 the tank could blow up
4. Hence it is needed for the process to become more precise

## 1. Project Proposal

- Brewing beer is one of the most profitable businesses in the United States as evidenced below:
- Americans spent more than \$37 billion on beer at Nielsen-measured retail stores. Only \$12.5 billion was spent on water, the most consumed beverage globally.
- Beer sales also outpace those of wine and spirits—the other two categories in the adult beverage space.
- Specifically, dollar sales growth in the craft realm ranged between just over 15% to 18% from 2013 through early 2016, well above the 1.3%-3.5% posted by the overall beer category.
- Since temperature is one of the most important factors for the taste of beer we decided to use technology to create the next best brew.
- Our system is simple and easy to integrate and also easy to debug with a nice user interface for the owner to monitor his brews and their respective sales and comments from different channels, all in real-time.



## 2. Architecture Overview



- Our system consists of the following -
  - IoT to monitor temperature, pressure and humidity, in a brewery.
  - Process the data using Databricks for real time analysis.
  - Monitor the results using Power BI and an iPhone App.
- We will be using Microsoft Azure as our PaaS, Power BI for our visualization and Swift for our iPhone App.
- Hardware/Sensing Layer – This is the very first layer of the IoT architecture. Also, it consists of the hardware device which is connected to the Network layer. Wireless Sensor Networks (WSN) and radio-frequency identification (RFID) are considered as the two main building blocks of the IoT. The Arduino Microcontrollers are directly connected with the sensors. The Arduino microcontrollers are also connected to the Raspberry Pi which is also connected to the internet using Ethernet or WiFi. It transmits the data collected from the sensor in real-time to the server.
- Network/Gateway Layer – This layer acts as the bridge in between Hardware /Sensing Layer and Management Layer. This layer receives the digitized data and routes it over wired LANs, Wi-Fi, or the Internet for further processing. There is a different protocol which can be used to communicate between IoT gateway and servers like MQTT, AMQP, COAP, and HTTP. We have discussed different protocols used in the next section.
- Application Layer – This is the final layer of the IoT analytics architecture. This layer uses the data processed by the management layer.

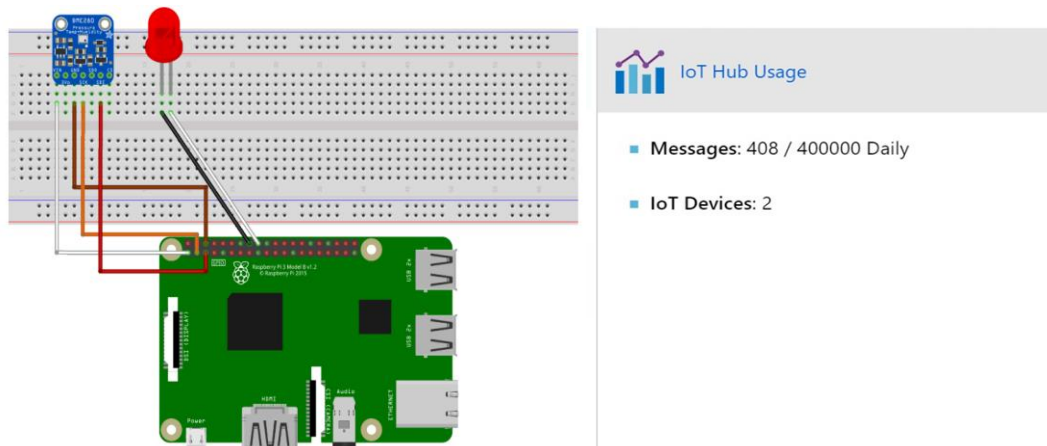
### 3. Data Source

For our project we will be getting data from two sources

- A. IoT Devices
- B. Recipe Data
- C. Brewery Sales Data

#### A) Getting Data from IoT devices:

For this project, we will use Azure raspberry-pi-web-simulator.



If the project is implemented on a large scale we will get data from n number of devices  
However, for our project we will be getting data from 2 IoT devices

There are three areas in the web simulator.

- 1. Assembly area - The default circuit of Pi.
- 2. Coding area - An online code editor to code with Raspberry Pi.
- 3. Integrated console window - It shows the output of the code.

```
Message sent to Azure IoT Hub
>
{"temperature_C":25.816221053486263,"humidity":72.20460527195279,"pressure_hPa":10.95357027
>
Sending message: {"messageId":80,"deviceName":"2","deviceId":"Iq3dGo1c2+q/xo1I7PSl4PEGBP5te
>
Message sent to Azure IoT Hub
>
```

## B) Getting Data for the Brewery:

To get the brewery data we have taken Kaggle dataset on brewery

The dataset consist of 75,000 homebrewed beers with over 176 different styles. Beer records are user-reported and are classified according to one of the 176 different styles. These recipes go into as much or as little detail as the user provided, but there's are least 5 useful columns where data was entered for each: Original Gravity, Final Gravity, ABV, IBU, and Color

### Recipe Data

Columns	Values	Type
BeerID	Record ID	Numeric
Name	Name	String
URL	Location of recipe webpage at <a href="https://www.brewersfriend.com">https://www.brewersfriend.com</a>	String
Style	Type of brew	String
StyleID	Numeric ID for type of brew	Numeric
Size(L)	Amount brewed for recipe listed	Numeric
OG	Specific gravity of wort before fermentation	Numeric
FG	Specific gravity of wort after fermentation	Numeric
ABV	Alcohol By Volume	Numeric
IBU	International Bittering Units	Numeric
Color	Standard Reference Method - light to dark ex. 40 = black	Numeric

BoilSize	Fluid at beginning of boil	Numeric
BoilTime	Time wort is boiled	Numeric
BoilGravity	Specific gravity of wort before the boil	String
Efficiency	Beer mash extraction efficiency - extracting sugars from the grain during mash	String
MashThickness	Amount of water per pound of grain	Numeric
SugarScale	Scale to determine the concentration of dissolved solids in wort	String
BrewMethod	Various techniques for brewing	String
PitchRate	Yeast added to the fermentor per gravity unit - M cells/ml/deg P	Numeric
PrimaryTemp	Temperature at the fermenting stage	Numeric

### C) Sales Data

Since we do not have readily available sales data for our dataset mentioned above, we will be generating the data for 3 months for the majority of the categories they will fall under using a normal distribution curve given a mean value and a standard deviation.

This data is based on certain statistics we have obtained such as -

1. The craft beer industry has total sales of 26 Billion Dollars as of 2017.
2. This amount is then divided into the top 5 categories.
3. We further divide this by 12 to get each month's sales and split it based on the ratio of the sales the top 5 categories contribute to.
4. Once we have each month's data we divide that by the number of elements which belong to that category and take that to be the mean value and assume an appropriate standard deviation.
5. We use a random number generator to generate a random number for this standard deviation and then add this to the mean value of the category to get the final value.

6. For 74000 rows we will have 3 values, represented by 3 extra columns, month 1, month 2 and month 3.

The sales data is generated using the equation below:

$$Total_j = \sum_1^3 \mu_{j,sales} + f(\sigma_{j,sales})$$

$j \in \{American\ Pale\ Ale, IPA, Ale, Stout\ etc.\}$

$\mu_{sales}$  = mean sales value of the category

$\sigma_{sales}$  = standard deviation of the sales of the category

$f(x)$  = random function generator which chooses a random  $\sigma$  value




## 4. Storage



IoT solutions can generate significant amounts of data depending on how many devices are in the solution, how often they send data, and the size of payload in the data records sent from devices. Data is often time series data and is required to be stored where it can be used in visualization and reporting as well as later accessed for additional processing.

1. **Durability.** Data stored in archive level is kept for equal to or longer than six months. Therefore, this tier is great for long-term back-up purpose.
2. **Scalability.** It meets the exponential increase of companies data storage requirements. Able to deal with massive data and improve companies productivity.
3. **Availability.** With Microsoft data centers spread out over the world and redundancy, risks are controlled and damages from natural disasters are prevented. Replicate data ensures the availability for all time.
4. **Confidentiality.** The solution guarantees that who are not supposed to have access to the data will not be allowed to read or modify data.
5. **Integrity.** Consistency assurance. Data consistency can be guaranteed because everything involved is validated if the object is modified. This solution allows you to access the latest version of information, regardless of time or space, and synchronize information with the rest of the team.



NAME	MODIFIED	ACCESS TIER	BLOB TYPE	SIZE	LEASE STATE
 lotBrew-02-2019-04-26-19-07					- ...
 lotBrew-02-2019-04-26-19-09					- ...
 lotBrew-02-2019-04-26-19-11					- ...

Output of Files stored in blob

## 5. ETL

Our reason to choose IoT Hub is well articulated by this article, a snapshot of which is provided below -

### Introducing IoT Hub device streams in public preview

Posted on January 23, 2019



Reza Sherafat, Senior Program Manager, Azure IoT

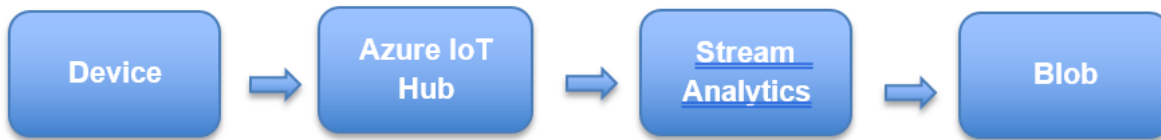
In today's security-first digital age, ensuring secure connectivity to IoT devices is of paramount importance. A wide range of operational and maintenance scenarios in the IoT space rely on end-to-end device connectivity in order to enable users and services to interact with, login, troubleshoot, send, or receive data from devices. Security and compliance with the organization's policies are therefore an essential ingredient across all these scenarios.

Azure IoT Hub device streams is a new PaaS service that addresses these needs by providing a foundation for secure end-to-end connectivity to IoT devices. Customers, partners, application developers, and third-party platform providers can leverage device streams to communicate securely with IoT devices that reside behind firewalls or are deployed inside of private networks. Furthermore, built-in compatibility with the TCP/IP stack makes device streams applicable to a wide range of applications involving both custom proprietary protocols as well standards-based protocols such as remote shell, web, file transfer and video streaming, among others.

### Transform

For transformation of streaming data from IoT sensors we are using **Azure Stream Analytics**, queries start with a source of streaming data that is ingested into Azure IoT Hub.. Stream Analytics can execute complex analysis at scale, for example, tumbling / sliding / hopping windows, stream aggregations, and external data source joins.





Send data to a monitored queue to trigger alerts or custom workflows downstream.  
 Send data to Power BI dashboard for real-time visualization.  
 Store data to other Azure storage services, so you can train a machine learning model based on historical data or perform batch analytics.

## A) Device Output

Input:

DeviceId	Time	Attribute	Value
1	2018-07-27T00:00:01.0000000Z	Temperature	50
1	2018-07-27T00:00:01.0000000Z	Temperature	50
2	2018-07-27T00:00:01.0000000Z	Temperature	40
1	2018-07-27T00:00:05.0000000Z	Temperature	60
2	2018-07-27T00:00:05.0000000Z	Temperature	50
1	2018-07-27T00:00:10.0000000Z	Temperature	100

## B) Stream Analytics Output

Solution:

```

WITH Temp AS (
  SELECT
    COUNT(DISTINCT Time) AS CountTime,
    Value,
    DeviceId
  FROM Input TIMESTAMP BY Time
  GROUP BY
    Value,
    DeviceId,
    SYSTEM.TIMESTAMP
)
SELECT
  AVG(Value) AS AverageValue, DeviceId
INTO Output
FROM Temp
GROUP BY DeviceId, TumblingWindow(minute, 5)
  
```

Output:

AverageValue	DeviceId
70	1
45	2

## C) Real Time Analytics

In the code given below, we write the logic to transform the data in **real-time** as the data is ingested from the IoT Hub. We compute the **running-average** for each sensor output for each device. Here lies the complexity of what we did -

Consider  $m$  devices each measuring the changes in  $n$  sensors for a time frame of  $t$  seconds. So for every second we receive =  $(m \times n)/t$  bytes of data streamed to the IoT hub.

Suppose there is a delay of  $s$  seconds, because of low bandwidth, during which there is no data generated by the devices, assuming both devices are in the same network, this would be  $(m \times n)/(t+s)$  bytes of data which has streamed.

So at  $t+1$  timestep the new data would be  $(m \times n)/t + (m \times n)/s$  bytes of data which means there is an extra  $(m \times n)/s$  bytes of data added to the stream in the next  $t+1$  time window.

While IoT hub might scale vertically or horizontally to account for this extra data, our code using stream analytics must also sufficiently account for this change since suddenly one blob size is much bigger than the other blob and this drastically affects the running mean.

This data is transformed and cleaned using the following code -

```
In [3]: from azureml.core import Workspace, Experiment, Run
import math, random, pickle
from azure.storage.blob import BlobBlobService
```

Run the next cell and follow the prompt to use device login to connect to Azure. Ignore any warnings about failing to load or parse files.

```
In [4]: ws = Workspace.from_config()
Found the config file in: /home/nbuser/.azure/config.json
```

We created an iPython notebook to load the sensor data from the blob which is stored in the raw form of type Apache Avro.

```

In [10]: blob_account_name = "testiot1123"
blob_account_key = "QPN7uEOB9sEfcWioW05ZnvZwV5fZNVnSLrTtrWgdDbr4b1CcXbY47rjhrLk2
KNJH3/SCivKIrN4EaaUwc1d2Rg=="
my_container = "testbob"
my_blob_name = "iottesthub"
my_data_file = "output"
https://testiot1123.blob.core.windows.net/testbob/iottesthub3/01/2019-02-14-19-1
0.csv

In [53]: import pandas as pd
import os
blob = BlockBlobService(blob_account_name, blob_account_key)
blobs = blob.list_blobs("testbob")
for blobby in blobs:
    print(blobby.name)

blob.get_blob_to_path("testbob", "iottesthub3/01/2019-02-14-19-10.csv", "output.
csv")

mydata = pd.read_csv("output.csv", header = 0, encoding="ISO-8859-1")
# os.remove(os.path.join(os.getcwd(), "output.csv"))
print(mydata.head(5))

# blob.get_blob_to_path(my_blob_name,my_container,my_data_file)

iottesthub3/01/2019-02-14-16-48

```

Read the data storages from the blob, as you can see the blob name, container and the output data format is mentioned above.

```

iottesthub3/01/2019-02-14-19-12.csv
iottesthub3/01/2019-02-14-19-56.csv
iottesthub3/01/2019-02-14-20-22.csv
iottesthub3/01/2019-02-14-20-24.csv
iottesthub3/02/2019-02-07-19-58
Obj avro.codec.nullavro.schema{"type":"record" name:"Message" \
0                                     false      NaN
1 $connectionDeviceId testbob2(connectionAuthMet... type:"sas"
2                                     false      NaN
3 $connectionDeviceId testbob2(connectionAuthMet... type:"sas"
4                                     false      NaN

namespace:"Microsoft.Azure.Devices" \
0                                     NaN
1 issuer:"iothub"
2                                     NaN
3 issuer:"iothub"
4                                     NaN

fields:[{"name":"EnqueuedTimeUtc" \
0                                     NaN
1 acceptingIpFilterRule:null}8connectionDeviceGe...
2                                     NaN
3 acceptingIpFilterRule:null}8connectionDeviceGe...
4                                     NaN

```

The output from the previous step is given above and as you can see the data is an Apache Avro object and the device id and values are being interpreted as NaN.

```
In [56]: import avro.schema
from avro.datafile import DataFileReader, DataFileWriter
from avro.io import DatumReader, DatumWriter
import json
# need to use this because the data dumped by the IOT hub is of avro hadoop format
# need to convert this to csv so that we can create dataframes from this to work on

def process_blob(filename):
    reader = DataFileReader(open(filename, 'rb'), DatumReader())
    temp_dict = {}
    for reading in reader:
        print(reading)
        parsed_json = json.loads(reading["Body"])
        if not 'id' in parsed_json:
            return
        if not dict.has_key(parsed_json['id']):
            list = []
            dict[parsed_json['id']] = list
        else:
            list = dict[parsed_json['id']]
            list.append(parsed_json)
    reader.close()
```

We write a function called `process_blob(filename)` where the filename is the blob name and change the format to JSON. The output is a list of dictionaries of parsed JSON values

```
    for reading in reader:
        print(reading)
        parsed_json = json.loads(reading["Body"])
        if not 'id' in parsed_json:
            return
        if not dict.has_key(parsed_json['id']):
            list = []
            dict[parsed_json['id']] = list
        else:
            list = dict[parsed_json['id']]
            list.append(parsed_json)
    reader.close()
    # for each device store the details
    for device in dict.keys():
        deviceFile = open(device + '.csv', "a")
        for r in dict[device]:
            deviceFile.write(", ".join([str(r[x]) for x in r.keys()]) + '\n')

process_blob("output.csv")
```

```
{'EnqueuedTimeUtc': '2019-02-14T19:09:26.7780000Z', 'Properties': {'temperature
Alert': 'false'}, 'SystemProperties': {'connectionDeviceId': 'testbob2', 'conne
ctionAuthMethod': '{"scope":"device","type":"sas","issuer":"iothub","acceptingI
pFilterRule":null}', 'connectionDeviceGenerationId': '636857593577455296', 'enq
ueuedTime': '2019-02-14T19:09:26.7780000Z', 'iotHubName': 'iottesthub3'}, 'Bod
y': b'{"messageId":172,"deviceId":"Mqm3ENJk95aIWv88Fa4J3Lyzx6tDC1ddQxmGWhldgnY
=", "temperature":23.53424205913287, "humidity":77.20021532597647}'}
```

On running the above function for each of the devices we then print the structured JSON output which is given above, where the Properties, System Properties, Body etc. are all compiled into one object file.

```
In [ ]: run = experiment.start_logging()

pi_counter = 0
n_iter = 100000

# Log total number of iterations
run.log("Number of iterations",n_iter)

for i in range(1,n_iter):
    # Monte Carlo step to update estimate
    x = random.random()
    y = random.random()
    if x*x + y*y < 1.0:
        pi_counter += 1
    pi_estimate = 4.0*pi_counter / i

# Log convergence every 10000 iterations
if i%10000==0:
    error = math.pi-pi_estimate
    run.log("Pi estimate",pi_estimate)
    run.log("Error",error)
```

We then generate the sales value using a monte Carlo simulation, which takes steps based on the computed value.

This data is then combined with the sales data we have partially generated as well as stitched together with Walmart sales data.

This combined data will have relatively low correlation with the output since they are unrelated but will be enough for us to train a model on and use the model to predict values received from the application front-end.

This data can now be ingested by a Python script since it is in csv format. We use stream analytics simultaneously to process and monitor the data ingested from IoT Hub and the

live output data is visualized using Power BI which allows us to make charts ad-hoc for the stakeholders.

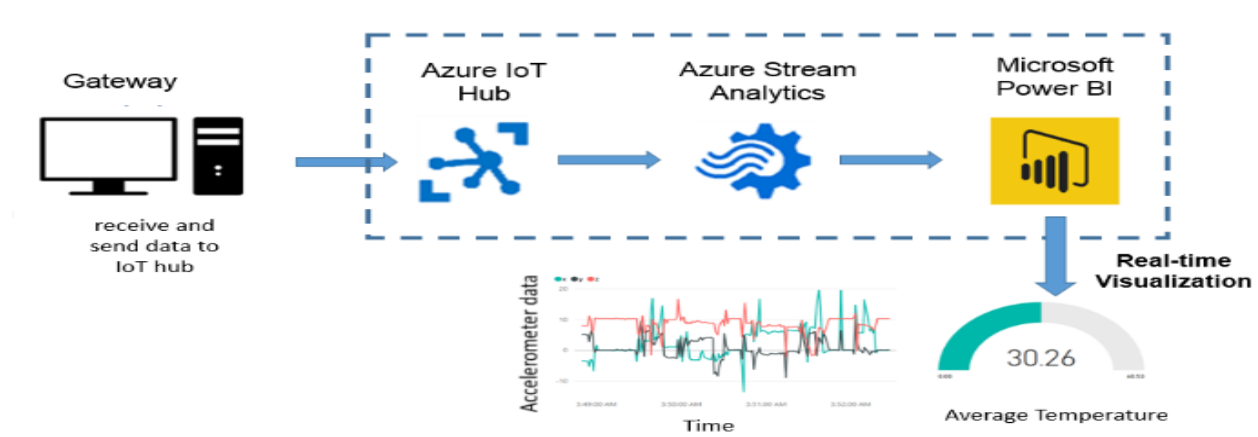
## 6. VISUALIZATION

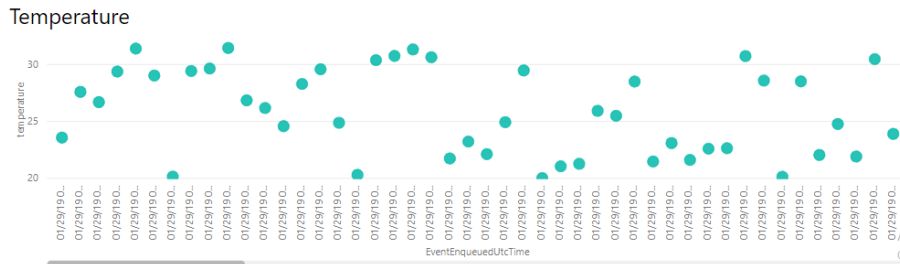
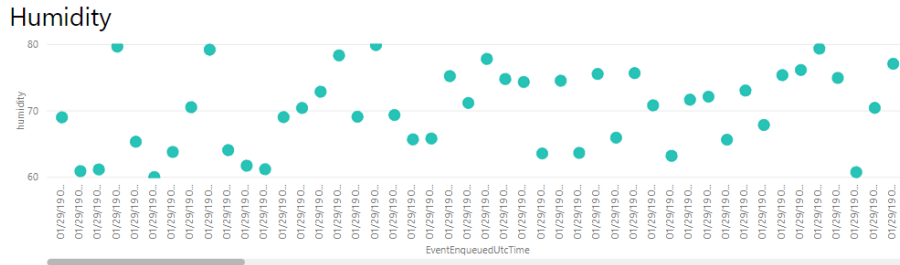
Our product has two main sections -

1. Where we monitor the raw sensor data coming from Azure IOT hub. Here the sensor data is a stream. This stream data sometimes can be held up due to a network issue or low bandwidth, in which case, in one time window the data is absent and in the next few time windows there is a flood of data. This has to be handled gracefully, including dynamic memory allocation and forced garbage collection to leave as little of a memory footprint as possible.
2. In the second section we perform basic exploration of data to find useful insights. We have used Azure IOT hub for getting IOT data, hence we are using Power BI for sensor data monitoring as it allows smooth integration with Azure IOT.
3. We have used exploratory IO for visualization as Exploratory provides a wide range of visualization types to help you explore your data and uncover hidden patterns quickly.

### A) IOT Devices monitoring (Power BI):

We are monitoring the output from our sensors in power Bi. A person can remotely monitor any variation in the temperature value while the beer is getting brewed and if there is a huge variation he can take corrective actions accordingly. We simultaneously receive data from temperature and humidity sensors from 2 different devices connected streaming at the same time.





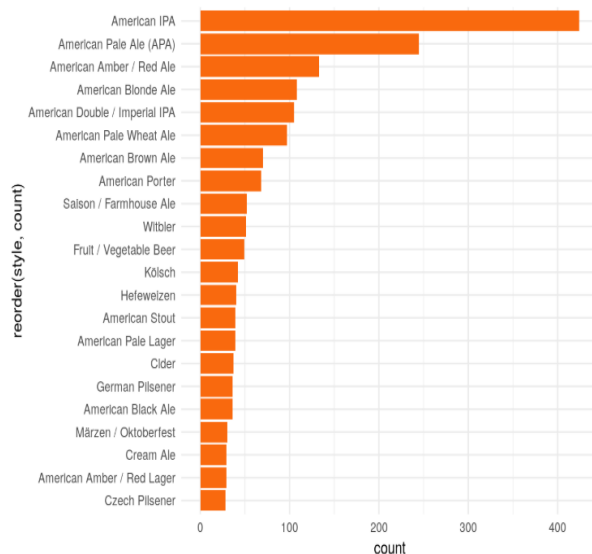
Activate Windows  
Go to Settings to activate Windows.

## B) Data Exploration (Exploratory IO):

We have used exploratory.io for basic data exploration and visualization. While exploring the data we found few insights that may help us for creating our final product. Below are the list of plots and their significance

### 1: Most Popular beer style

**Significance:** It looks like the most popular craft beers are American IPA, APA and Red Ale in the medal positions.

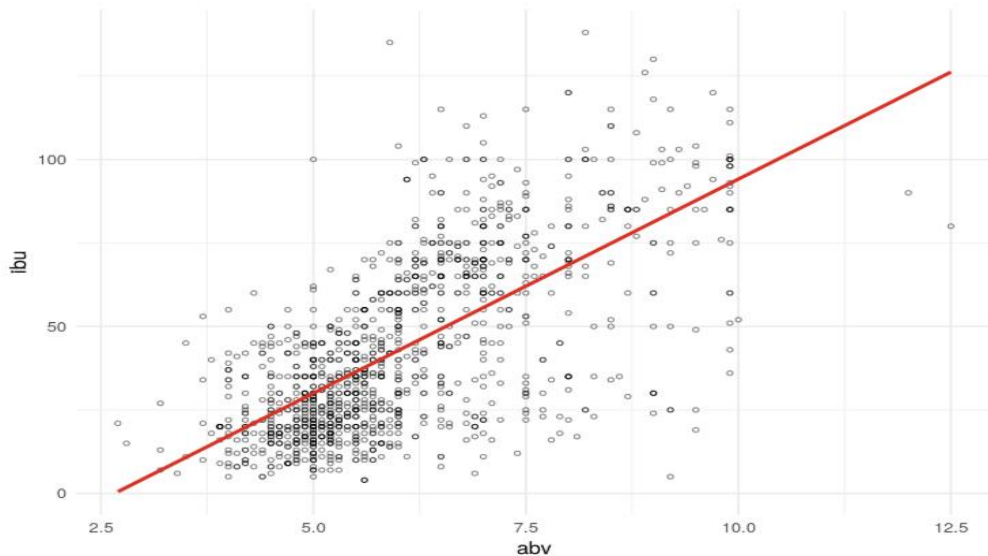


Bohemian Pilsener  
Blonde Ale  
Cream Ale  
**American IPA**  
American Pale Ale  
Belgian Blond Ale  
Belgian Golden Strong Ale



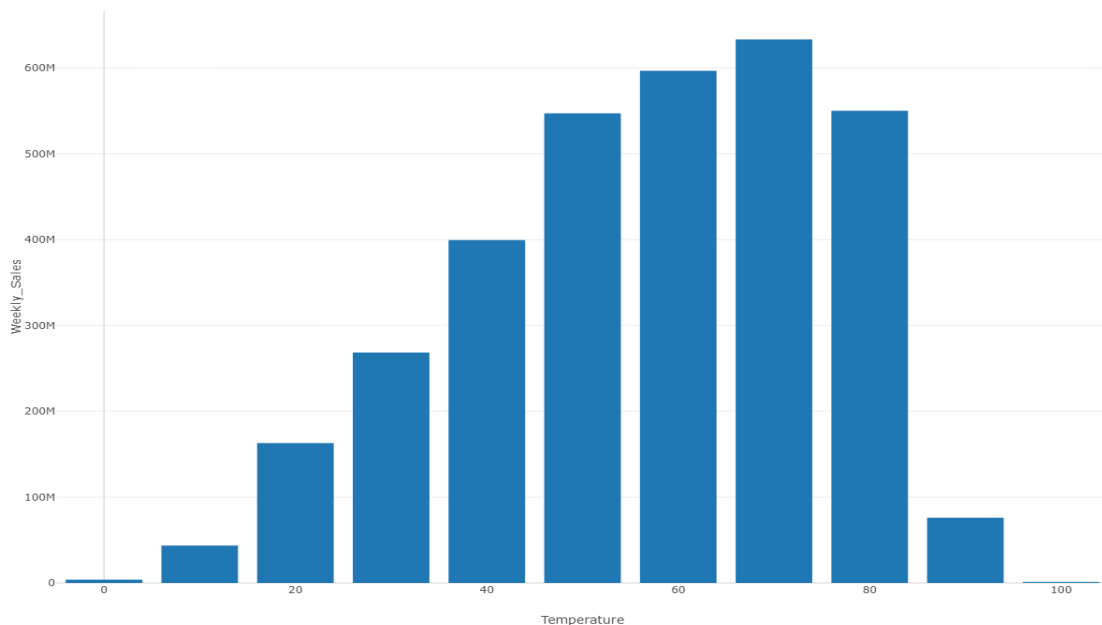
## 2: ABV vs. IBU (bitterness and alcohol content.)

**Significance:** There is a clear linear relationship between ABV and IBU where if I increase the ABV the IBU should also increase.



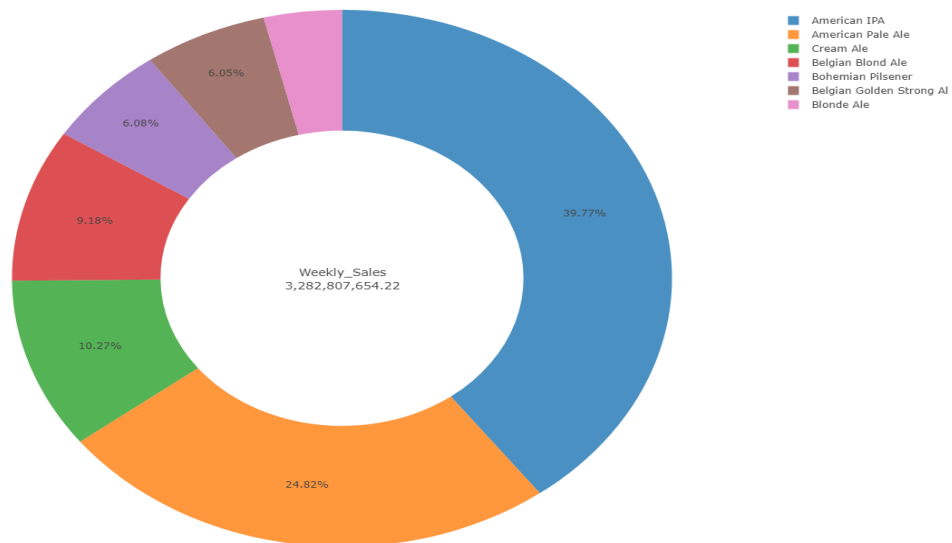
## 3: Effect of temperature on weekly sales

**Significance:** This curve seems to tend more towards a bell curve. We see that between the temperatures of 50 and 80 degrees the sales average around 600 million dollars. But surprisingly, as the temperature got higher the weekly sales dipped drastically by 550 million dollars, which indicates a very high preference to other drinks. Or a more rational explanation is our dataset did not contain too many points of data at 90 degrees since the temperature does not go that high as often.



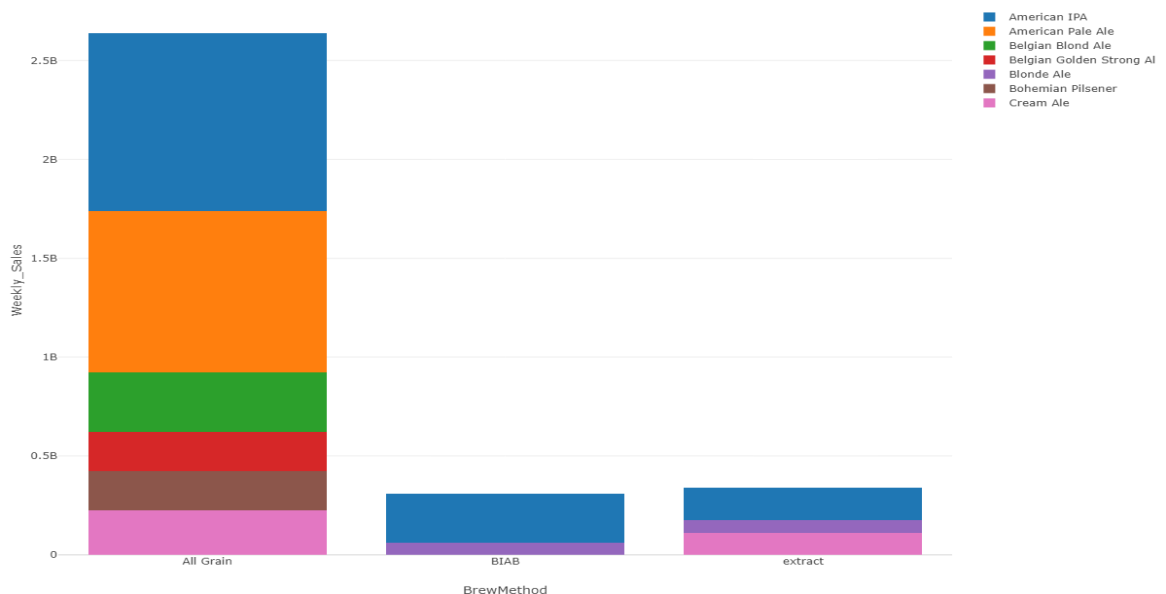
#### 4: Distribution of various styles of beer with respect to sales

**Significance:** We see that the American IPA is the most favoured type of craft beer with the American Pale Ale coming a close second. Together they account for nearly 65% of the sales in craft beer.



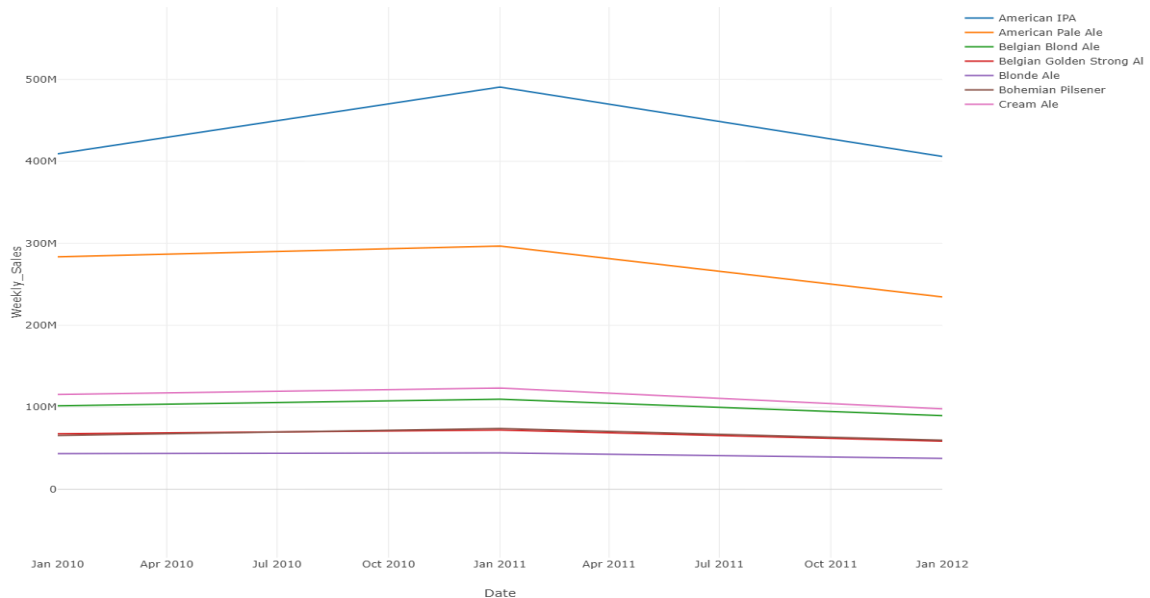
#### 5: Effect of various beer style on weekly sales by the different brew methods

**Significance:** Since we are interested in brewing the best beer, which is also the most efficient, it is interesting to see how the top beers are brewed. As we can see most of them are All Grain, with a very small percentage going to brew in a bag and extract brewing.



## 6: Time duration effect on weekly sales by different styles of beer

**Significance:** We can see over the two years the sales have been pretty consistent with a very small variation taking place at 2011.



## 7. Product

While crafting a beer there are mainly two aspects one being the temperature, humidity and pressure that needs to be monitored closely while crafting the beer and second is integrate this data and create an app that can be used for future prediction of beer sales

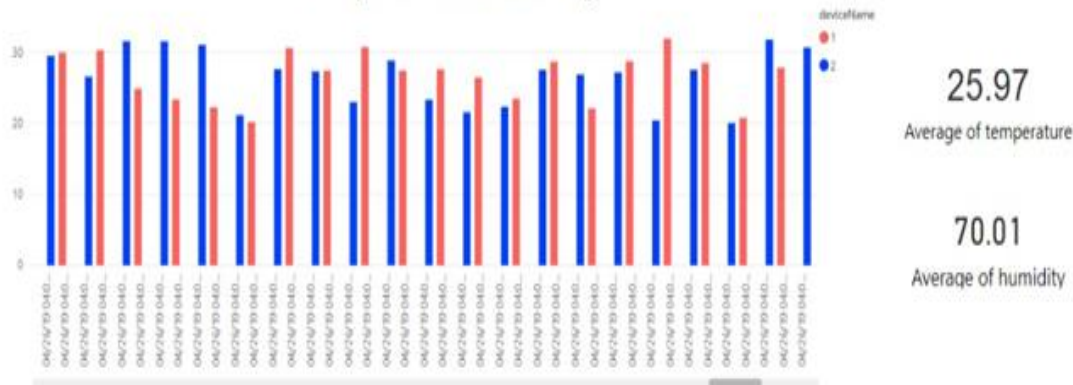
The Final Product has two parts

1. Monitoring
2. App

### Monitoring

We are monitoring the output from our sensors in power Bi. A person can remotely monitor any variation in the temperature value while the beer is getting brewed and if there is a huge variation he can take corrective actions accordingly. We simultaneously receive data from temperature and humidity sensors from 2 different devices connected streaming at the same time.

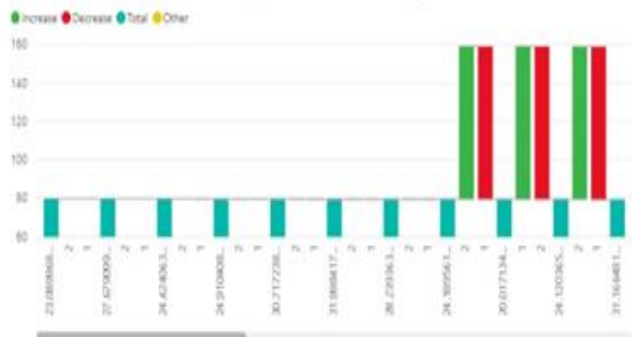
### Temperature Monitoring



### Temperature by Device



### Temperature Vs Humidity



## APP

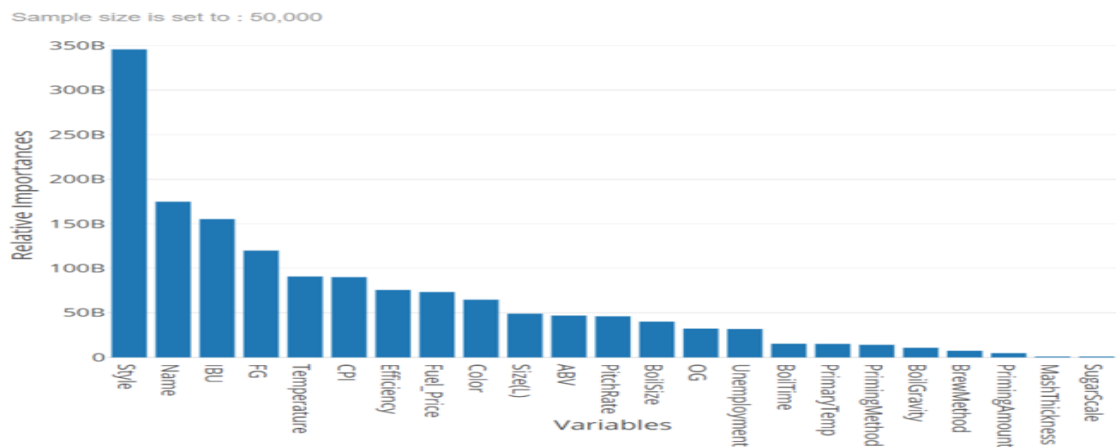
We created an app that can help brewery owner to know what is the possible return on investment based on certain parameters used for manufacturing beer.

UI > Flask Server > Multiple Linear Regression Model > Output back to UI



## Variable Importance

We created an app that takes Temperature, Color, IBU, Style, Brew Method, Primary temperature based on the variable importance plot created with respect to sales



## Machine Learning Model

The goal is prediction, or forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response.

```

49
50 regression_model = LinearRegression()
51 regression_model.fit(X_train,y_train)
52
53 print(regression_model.score(X_test, y_test))
54
55 print(X_test.head())
56
57 y_predict = regression_model.predict(X_test)
58
59 regression_model_mse = mean_squared_error(y_predict,y_test)
60
61 print("predicted data set",y_predict)
62
63 print(regression_model_mse)
64
65 print("mean square error",math.sqrt(regression_model_mse))
66
67 exists = os.path.isfile("./ls.sav")
68
69 if not exists:
70     pickle.dump(regression_model, open("ls.sav","wb"))
71
72 regression_model.predict(
73     [[24.2, 25.11, 5.33, 62.24, 7.22, 0, 0]])
74
75
76 def predict_function(temperature, primary_temp, abv, ibu, color, style, brew_method):
77
78     temp_df = pd.DataFrame({'Temperature':(temperature),
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

## Output of Machine Learning Model

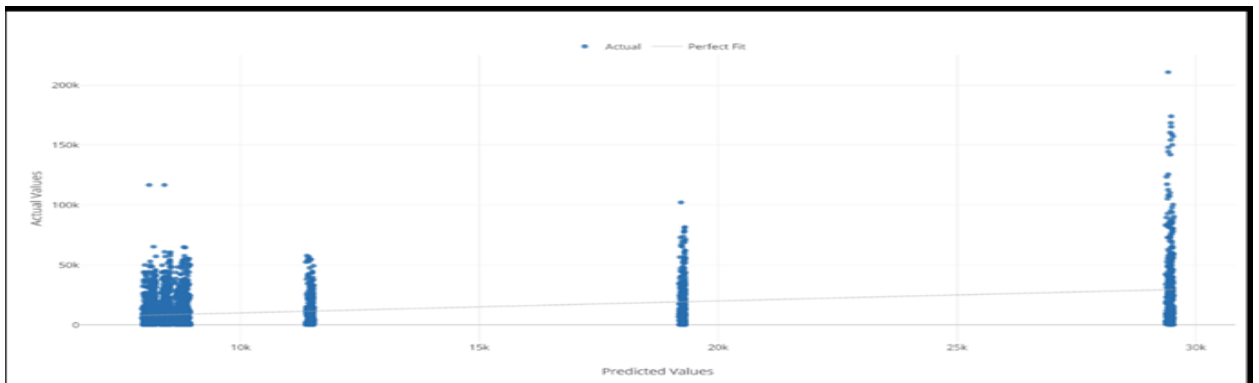
```

new pred ABV          float64
BrewMethod      object
Color           float64
IBU             float64
PrimaryTemp     float64
Style           object
Temperature     int64
dtype: object
ABV BrewMethod Color IBU PrimaryTemp Style Temperature
0  5.3 All Grain  7.22 62.24      20.2 American IPA      92
[[4277.57888911]]

```

## Predicted Vs Actual

Since the data was not highly correlated there is a large skew between predicted and actual value



## Beer IOT Beer Awaits

**Temperature**

**ABV**

**Color**

**IBU**

**Style**

**Brew Method**

**PrimaryTemp**

**Sales**

Click for Sales

## Beer IOT Beer Awaits

**Temperature**

**ABV**

**Color**

**IBU**

**Style**

**Brew Method**

**PrimaryTemp**

**Sales**

Click for Sales



## 8. Areas of Impact

### Industry Speaks

Jayne Coffee  
Director of Operations-[Tongue & Groove](#)

The owner we spoke with cited a rule of thumb for labor costs that says it takes about 20 hours of work to make a batch of beer, regardless of the size. The going rate for a ground-level brewer at a non-union brewery is about \$12 an hour, meaning it costs \$200 in labor to make a batch. Assuming the 30-barrel batches that are standard at relatively small breweries, that means **15 cents of labor goes into a typical six-pack of craft beer.**

### Impact

1. By continuously monitoring the beer temperature and pressure it will save the brewery owners cost of manual labourers and also they can see if there is any huge variation over a span of time
2. If a brewery owner wanted to craft a new style of beer then he can use the app to predict his future sales based on Style, Alcohol by volume ,temperature.

### Stakeholders

#### DWAYNE 'THE ROCK' JOHNSON IS LAUNCHING HIS OWN BRAND OF TEQUILA

Drink it on the rocks.



#### Ryan Reynolds Buys Stake In Aviation Gin, Cashes In On Celebrity Liquor Rush



## 9. Challenges Faced

### ➤ Data

It was difficult to get data related to Brewery sales especially related to craft beer. We had to merge data with Walmart sales which was very challenging

### ➤ IoT

The data coming from the simulated sensors had to be configured properly. There is a limitation on basic subscription how much data can be processed by the IOT hub owing to this restriction we could not collect large amount of data

### ➤ Machine Learning Model

Due to disjoint data we could not find much of a correlation between the output sales and temperature. Also, the data related to beer style was not formatted properly due to which we had a hard time fitting the model

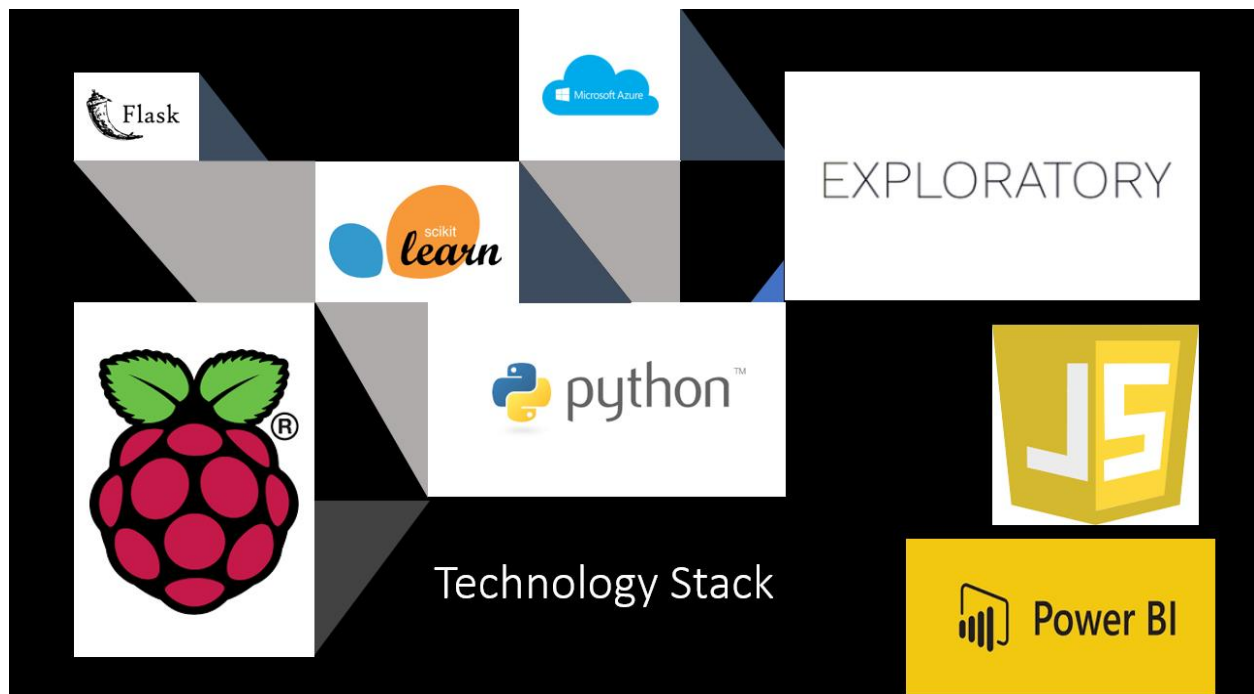
### ➤ Stream Analytics

We had to check if the query written was able to solve the issue of duplicate records. This caused a lot of inconvenience

### ➤ Integration

There were several components that were interconnected to form the entire architecture. We had to make sure that the configurations had been set properly

## Technology Stack



## Existing PoC



## 10. Future Scope

- For our project we have created a web app to predict future sales we can create an mobile app that can help brewery owners or anyone who is getting into the beer industry to get a prediction on what might possible return on investment
- To integrate it with actual sensors and connect the output to Azure IOT hub  
We can use the IOT monitoring across various domains and industries that manufacture similar product which can help them reduce labor cost.
- To integrate more data to get better accuracy and apply more machine learning models.

## 11. Project Experience

This experience has been fun to say the least. My initial dogma of not using either AWS or Azure and build my own microservices cloud deployed on Docker was met with some surprise, but on realizing that this isn't a small task, So I decided to use the cloud and build a solution with a good business use-case. My search did not take me far, as we as a team settled on using IoT to brew the ideal beer. This hit all our points on the checklist –

1. Hardware Integration.
2. Live Streaming from a physical device.
3. Unstructured data.
4. AI
5. Flask Application



The experience had us working across the different sectors of the data science pyramid, working from the descriptive analytics, identifying what happened, how is the beer produced, then diagnostic analytics, why did a sensor rating go wrong, predictive analytics, what will happen based on the values we have collected till now and then finally prescriptive analytics, which is where there is a remedy posted based on the analytics achieved.

Bringing so many moving parts together was a challenge in itself. With just the 3 of us building out an entire cloud solution, writing the machine learning code and finally an app to help owners predict the solutions.



Reinforcement learning was something I will definitely consider in the future, considering it's impact, but considering the business case we had decided to address, it did not have much of a use case here. But I will definitely consider this in the future.

This class was unique in the sense that it allowed us to run wild with our imagination and had no restrictions on technology, which was a boon and a shark tank like evaluation of the business case. Looking forward to more classes like this in the future.

## Reference

- [1] <https://www.nielsen.com/us/en/insights/reports/2017/the-state-of-the-us-beer-market.html>
- [2] <https://www.sciencedirect.com/science/article/pii/S003295929800154X>
- [3] <https://www.sciencedirect.com/science/article/pii/S0032959200002673>
- [4] <https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2050-0416.2001.tb00083.x>
- [5] <https://azure-samples.github.io/raspberry-pi-web-simulator/#GetStarted>
- [6] <https://www.kaggle.com/jtrofe/beer-recipes/data#recipeData.csv>