Optimized Warehouse Management System Web Application

a final report,
submitted in complete fulfillment.

For the purpose of validating participation in

The IBM Hack Challenge 2020,

conducted by the SmartBridge.

Submitted by:

Samrat Chaudhuri.

Team name:

The Czar.

Contents

Chapter 1: Introduction

1.1. Overview

The entire setup of the Web Application has been created to assist the user, which in our particular case, will be the owner or employees of the food delivery chain. The projected web application contains 4 essential parts or tabs.
The essential tabs gradually and almost with utmost certainty, help the user glide through the desired process of predicting future number of order, also keeping in mind some additional benefits, such as keeping a check on a sound mental health, staying updated with the events in and around the world in general and the business in particular.
The 4 tabs of the web application are as follows,

- **About tab :** This is the first tab that the user is navigated to once he or she decides to open and use the web application.
- **Assistance tab :** This is the second tab that the user might use. Contains the chat bot, named The Czar bot.
- **Predictions tab :** This is the third and the most important part of our web application. This part is the reason behind building this web application. Uses backend ML model to predict outcomes.
- **Dashboard tab :** This is the last yet most visualy appealing tab in the web application. Contains charts, graphs and tables depicting the relations between target and non target attributes.

1.2. Purpose

The purpose of the web application is ultimately, to predict the number of orders, for a particular center and a particular deliverable consumable good, not only for a given week but also for the upcoming 10 weeks, for anticipated storage purpose. Although the main function of the web application is prediction, there are numerous other add on services added onto the very web app to make it more user friendly and to keep a user engaged for as long as possible.
The varying purposes of each of the tabs are mentioned as follows,

- **About tab :** The about tab is the first tab that our user is navigated to once the user opens

the web application. The About tab, as the name suggests, provides the user with a variety of information through the various widgets that it has. The widgets being, as follows,

Our Motivation Widget, gives the user, an insight to the cause or thought that went behind the creation of such a web application. It also goes onto inform the user about the current global standing of India in the hunger index.
Essential info Widget, provides the user with essential data required during the prediction, i.e. center id and meal id. It becomes impractical for the user to remember these unique datas or to carry large documents for the same, thus for the convenience of the user these have been hosted on the web app.
News Widget, provides the user with news regarding the various happenings in the world in general, and food news in particular.
Youtube Widget, provides the user with colourful food related videos, solely for the user's entertainment.

- **Assistance tab :** This tab contains the chat bot, named The Czar bot. The Czar bot has been programmed to answer all the queries of the user, both through an output displayed in a text form and through audio form. Apart from a few general queries and greetings, the Czar bot is also capable of,
  - intercepting and appropriately reciprocating to both positive and negative feedback, since mental health is important.
  - answering queries, related to food business in general and governmental policies regarding the same, through a smart document understanding system embedded within the Czar bot.
  - giving updated stock lists of the Czar's warehouse items.
  - telling jokes and giving tips, month wise, for the user to understand and better utilise the seasonal changes to his/her benefit, and store sufficient stocks without the risk of spoilage.

- **Prediction tab :** This tab has an Machine Learning Regression algorithm model running in the back end. The purpose of this tab is to give predictions in the form of number of orders, for a particular center and consumable item, for a given week and 10 week ahead.

- **Dashboard tab :** This tab contains the visua representations of the dependencies between the target and non target attributes. A better insight to the relationship between the attributes and their variations accordingly, through graphs and tables is provided.

Chapter 2: Literature Survey

2.1. Existing Problem

A food delivery service has to deal with a lot of perishable raw materials which makes it all, the most important factor for such a company is to accurately forecast daily and weekly demand. Too much inventory in the warehouse means more risk of wastage, and not enough could lead to out-of-stocks - and push customers to seek solutions from your competitors. The replenishment of majority of raw materials is done on weekly basis and since the raw material is perishable, the procurement planning is of utmost importance.
So, the need of the hour is a method to predict the prospective demand amounts in the not so distant future so that the supply chain can stock and meet every incoming demand of the customer.

2.2. Proposed Solution

According to the UN's Food and Agricultural Organization, a massive one third of the total amount of food produced globally is wasted, which is worth over $750 billion. India currently ranks 7th in terms of overall food wastage. Thus, pre predicted number of orders would significantly reduce the wastage during procurement. The software solution to the problem solution being: • IBM Watson Machine Learning instance: Helps predict the desired number of orders as per the changing independent variables, using algorithms. Connected through REST APIs and storing metadata in Cloudant DB. • News feed: Using iframe source code to embed e-newspaper in the Web Application. It helps the user get a better understanding of the food world. • Youtube Videos: Keeps the user interested in the Web App through colourful display of vibrant food items. • IBM Watson Cognos Embed: Connected through shareable preview link, this creates and hosts a live interactive dashboard, using uploaded datasets, enhances the understanding of the user. • IBM Watson Assistant: Connected through REST APIs, this helps create and host Chat bots in Web App. • NODE RED: Visual Flow editor, used to host the Web Application.

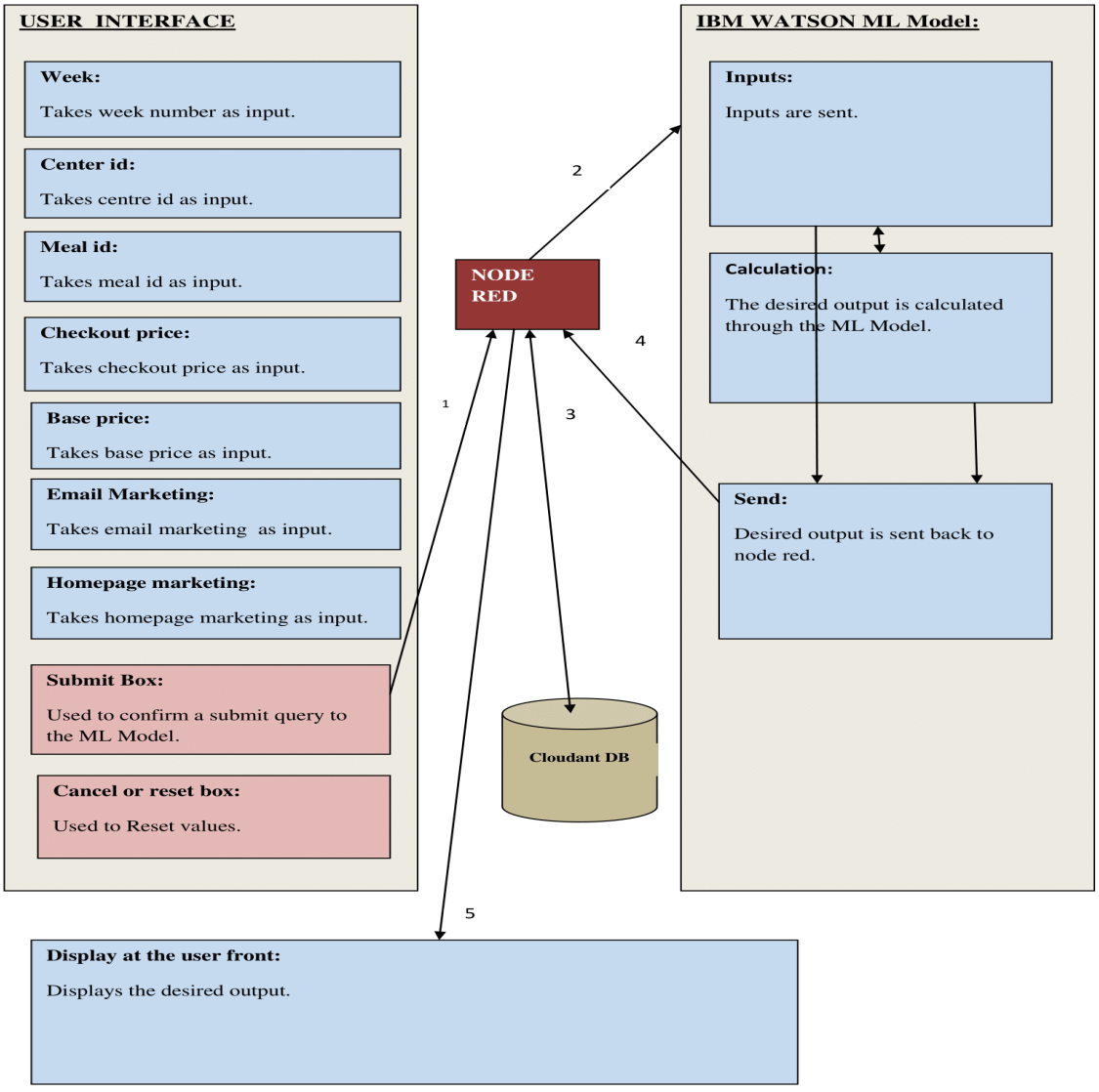
Chapter 3: Theoritical Analysis

3.1. Block Diagram

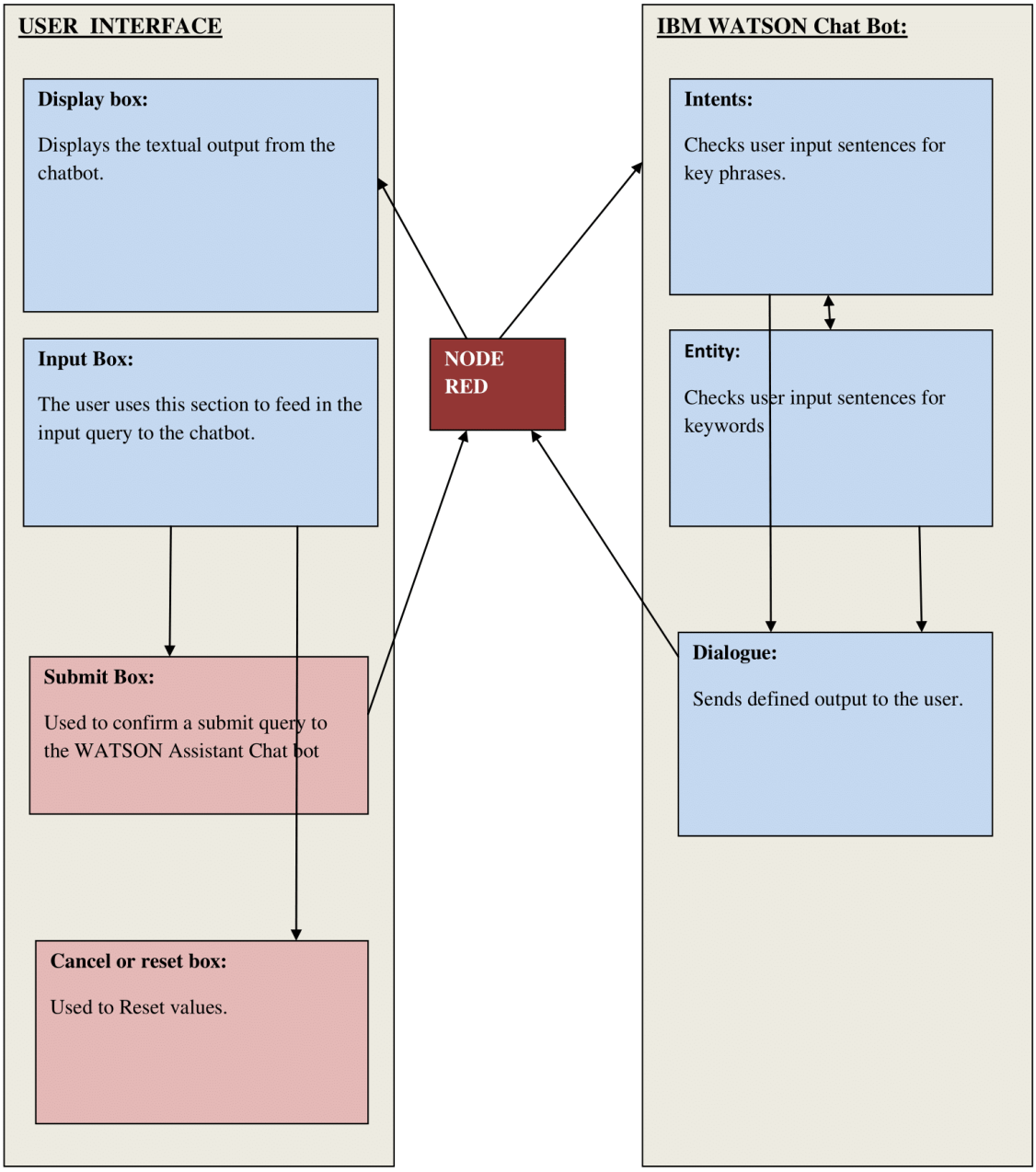
Although there are several small components in the entire web application, but the two main parts which form the bulk of the functioning and almost all of the interative user interface, are,

- The Machine Learning model running in the backend.
- The Chat bot.

The working of the ML Model block diagram,



The Chat bot working block diagram is as follows,



3.2. Software Design

Technology Stack used:

o IBM Watson Studio Machine Learning instance:

1. Server type: REST
2. Programming Language: Python
3. App: Jupyter Notebook
4. Hosted: NODE RED
5. Database: Cloudant DB
6. API Endpoints: Mentioned in service credentials

o IBM Watson Assistant:

1. Server type: REST
2. Programming Language: Python
3. App: Chat Bot
4. Hosted: NODE RED
5. Database: Cloudant DB
6. Algorithms used: Fuzzy logic, Natural Language Processing, Speech Processing.
7. API Endpoints: Mentioned in service credentials

o IBM Discovery Services:

1. Server type: REST
2. Programming Language: Python
3. App: Smart Document understanding imparted to the chat bot
4. Hosted: NODE RED
5. Database: Cloudant DB
6. Algorithms used: Fuzzy logic, Natural Language Processing.
7. API Endpoints: Mentioned in service credentials

o IBM Watson-Cognos Embed:

1. Server type: REST
2. Programming Language: Python
3. App: Dashboard
4. Hosted: NODE RED
5. Database: Cloudant DB
6. API Endpoints: Mentioned in service credentials

o NODE RED:

1. Server type: NodeJS Project (Web-APP + REST Server)
2. Programming Language: Python, HTML, PHP, CSS, Javascript.
3. App: NODE RED
4. Database: Cloudant DB

Implementation Details:

- Module 1: IBM Watson Studio Machine Learning Instance:

Creation:

The Machine Learning model is the heart of our project, since the central idea of predicting the number of orders and thereby reducing the wastage will be computed by the Machine Learning Model. Therefore, the model has be fitted with a dataset, and different pipelines have to be tested in order to select the best pipeline. The best pipeline being that which has the maximum accuracy, in our case, the maximum accuracy being that of 76.2%. The algorithm in use being **Random Forest Regressor.**

Hosting:

After selecting the desired Machine Learning Model, the next step is to host it in the Web Application. This process is accomplished through the NODE RED flow editor, using various nodes such as, Form node. http in node. http request node. Function node. Debug node. Template node. After the successful configuration of the aforementioned nodes in a required manner, the

flow needs to be deployed in order to see the web app in action.

Process flow:

The user feeds the User interface input option with essential data so as to get the requisite number of orders for subsequent 10 weeks and the given week. After the data is filled and submitted, the data is fed to the Model, using NODE RED API Connections and the subsequent output generated is displayed in the UI of the web app.

- Module 2: IBM Watson Assistant:

Objective:

For the user to have a better experience while using the Web Application, inclusion of an interactive chat bot was crucial. The Chat bot not only helps the user get a better experience through inputs of business related data, but it also gives feedback to the user and itself acknowledges feedback, shares jokes and is appreciative of human interaction, since, our goal has also been to promote mental health.

Creation:

The Watson Assistant Skill section has intents to analyze user input sentences, entities to analyze user input key words and dialogs, to reciprocate an output. It all depends upon an individual, as to how and where to drive the anticipated conversation, using fuzzy logic, NLP and SP.

The Czar bot is also endowed with a smart document understanding capability, therefore, it can understand and answer several queries generated regarding the food business, known and unknown competition and related government policies. This capability is courtesy the Watson Discovery Services.

Hosting:

The Czar bot is hosted using APIs entered in the NODE RED flow, containing the Chat bot Node, along with a variety of function, template and audio nodes, and a form node.

Process Flow:

As the user submits the enquiry through the user interface on the chat bot widget in the Web

App, the NODE RED sends the enquiry to the Chat bot and reverts back with a prompt reply along with an audio reply of the same.

- Module 3: IBM Watson Cognos Embed:

Creation:

A live interactive dashboard is created for the user to get a better understanding of the dataset. The dataset used to create the Machine Learning model, is uploaded to the dashboard and subsequent charts and flows, depicting the relation between the target attribute to the independent attributes, are designed and implemented.

There are two parts to the dashboard,

o The Basic interpretation tab is for the user to begin to understand the nature of the dependencies. o The Advanced interpretation is for a user who has already developed an understanding for the aforesaid dataset.

Hosting:

The NODE RED Template node, along with the shareable preview link of the Cognos dashboard is used to embed the dashboard into the Web Application.

- Module 4: Web App: User Interface:

Our Motivation:

This section contains an article depiction from a popular and trusted newspaper, used to solidify the user’s knowledge base and thereby get a better understanding of the business in general and our objective in particular.

Essential info:

Used to feed the user with essential data that he might need for computation of the predictions using the ML model, which otherwise would’ve been cumbersome to memorize.

News feed:

Provides a daily dosage of Global news in general and food related news and recipes in particular.

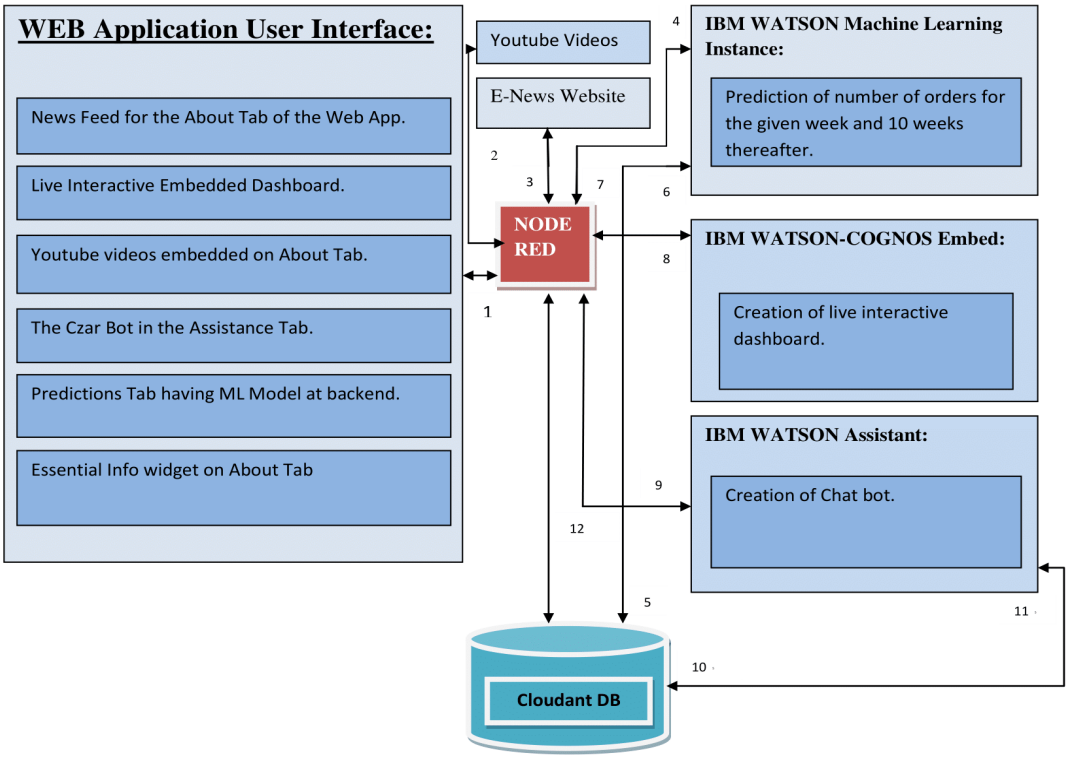
Youtube Video:

Used for users who don’t read much news and are more drawn towards audio visual representation.

Chapter 4

4.1. Flowchart

This flowchart goes on to depict the working of the entire web application.



INDEX

1. Sending Requests from User Interface to the NODE RED Flow, using input options.
2. Youtube Videos iframe Sources for embedding the videos in Web APP.
3. News feed iframe Sources for embedding the news articles in Web APP.
4. Sending essential data to the Machine Learning instance in the backend to get the desired output.
5. Computing, storing and retrieving data from Cloudant Database.
6. Sending the results back to the ML instance.
7. Sending the outputs to NODE RED flows for display using REST APIs.
8. Embedded Dashboard displayed through shareable preview link.
9. Input to The Czar bot (Chat bot) through the input options.
10. Sending query to Cloudant Database for fetching the required data for analysis and output.
11. Data sent to IBM Assistant Chat bot.
12. Computation and sending data to NODE RED flow for display using REST APIs.

4.2. Experimental Investigation

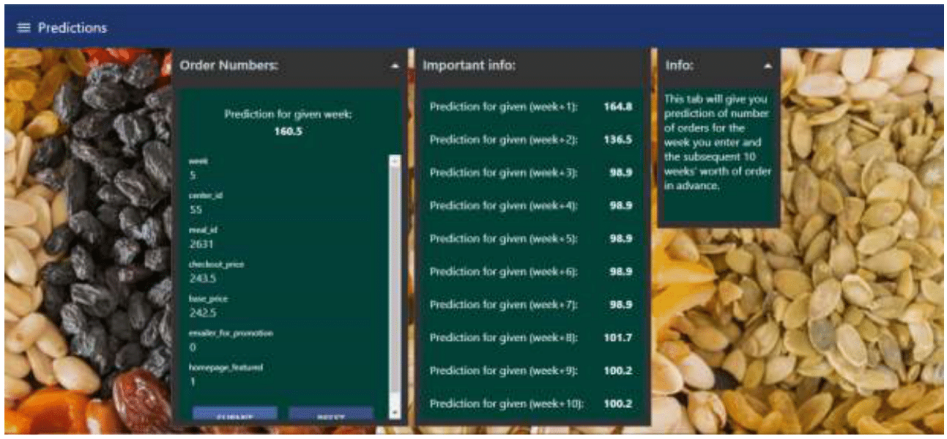
Experimental investigation done over a wide range of experimental data and subsequent output only go on to show that,

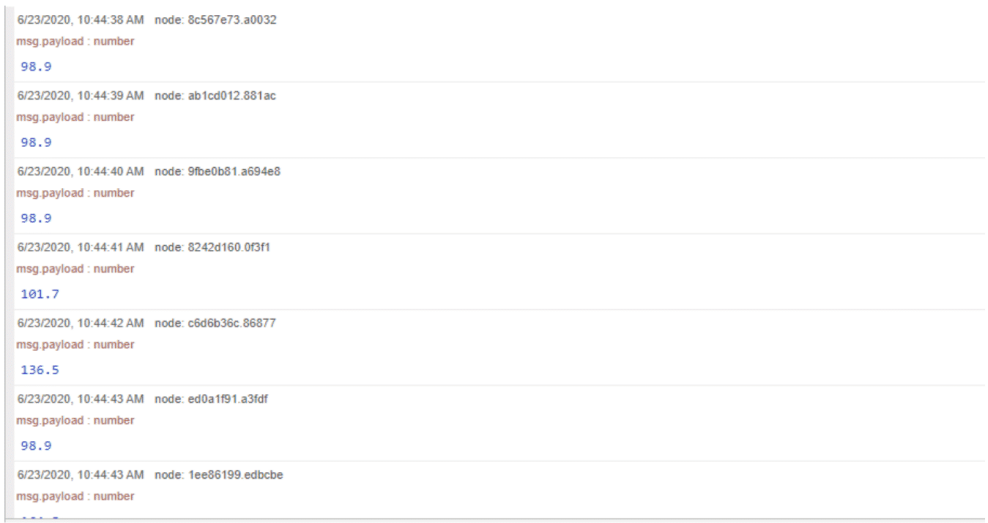
- Meal-id 2290, has the highest number of orders almost throughout the year, in almost all of the centers, whereas Meal-id 177 has the lowest number of orders in almost all of the centers and that too throughout the year.
- Center-id 10 has the highest rating amongst the other centers available for this food delivery chain, whereas, Center-id 186 has the lowest rating.
- Spending a substantial amount on advertising an item, both through email and by featuring it on the web page, increases the sales of that item by a significant margin. Thus advertising is impportant.
- There is a surge of delivery orders towards the middle or end of every month, a carefully and meticulously dedicated lineup and workforce can deal with the periodic surge gracefully.

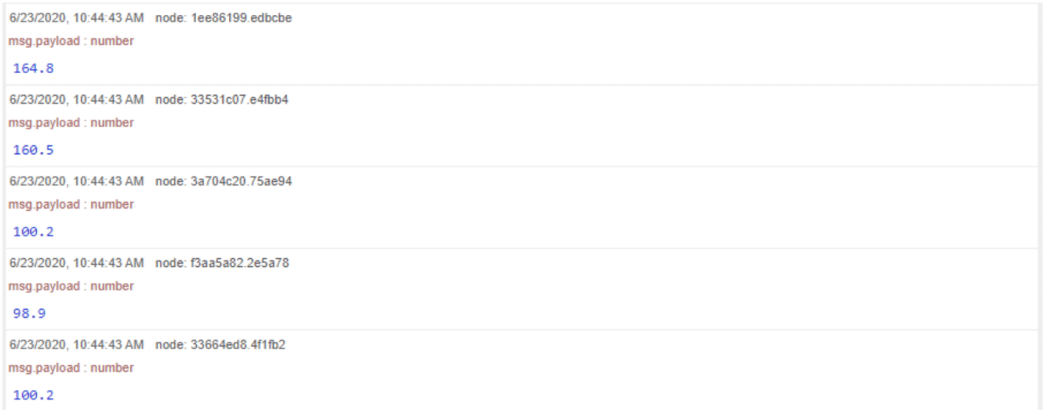
Chapter 5

5.1. Result

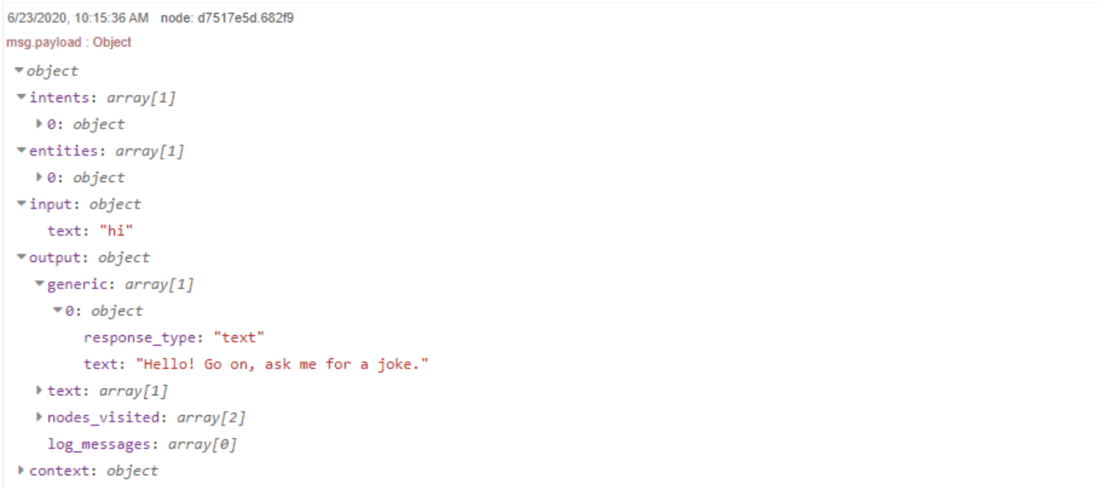
Results for Machine Learning model, from the debug section of NODE RED flow.

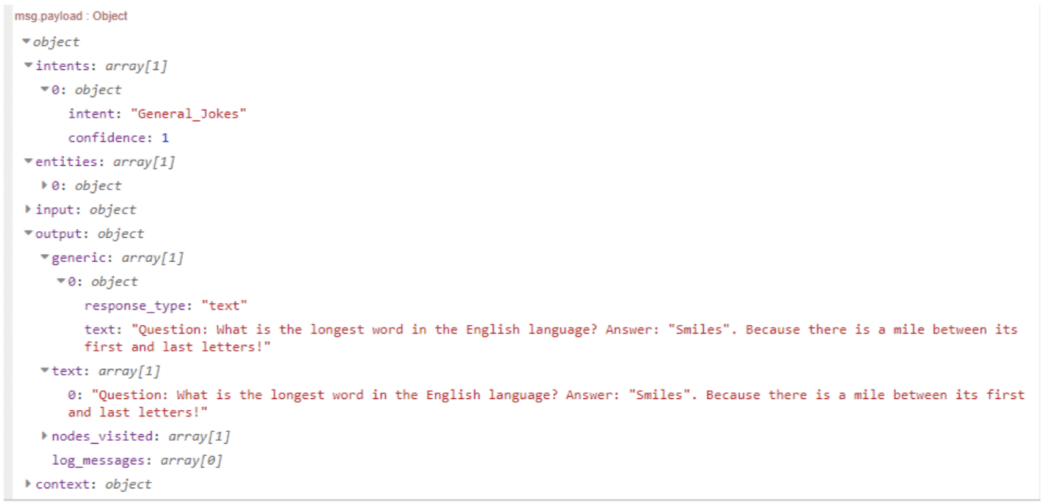


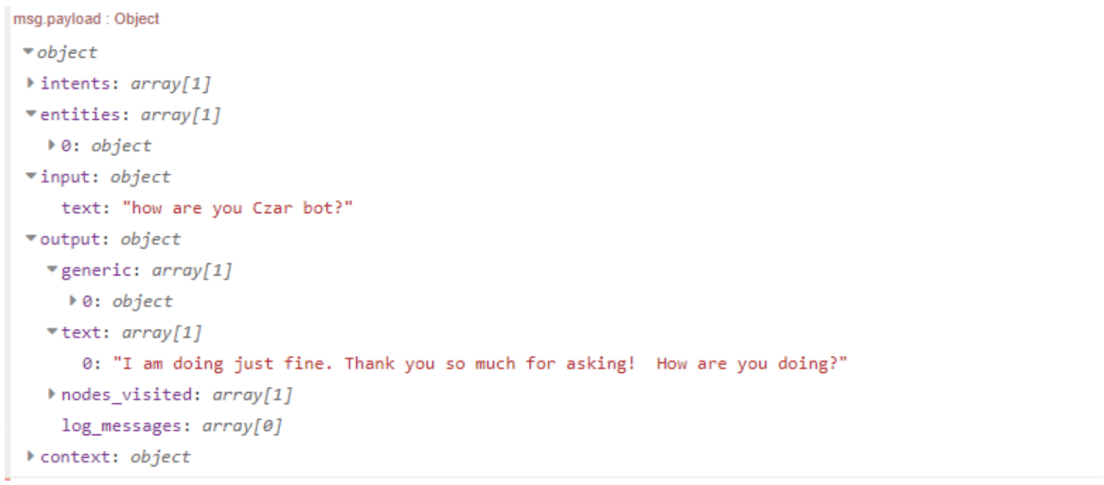




Chat bot results from NODE RED Debug section.




msg.payload : Object
object
intents: array[1]
0: object
intent: "General_Jokes"
confidence: 1
entities: array[1]
0: object
input: object
output: object
generic: array[1]
0: object
response_type: "text"
text: "Question: What is the longest word in the English language? Answer: "Smiles". Because there is a mile between its first and last letters!"
text: array[1]
0: "Question: What is the longest word in the English language? Answer: "Smiles". Because there is a mile between its first and last letters!"
nodes_visited: array[1]
log_messages: array[0]
context: object


msg.payload : Object
object
intents: array[1]
entities: array[1]
0: object
input: object
text: "how are you Czar bot?"
output: object
generic: array[1]
0: object
text: array[1]
0: "I am doing just fine. Thank you so much for asking! How are you doing?"
nodes_visited: array[1]
log_messages: array[0]
context: object

5.2. Advantages & Disadvantages

- **Advantages**

i. The web app predicts the number of orders for the upcoming weeks in advance, with utmost accuracy, thus decreasing the wastage of consumables while stocking up.
ii. The web app takes the mental well being of the user into consideration and acts on it through the Czar bot, also keeping a routine tab on the same.
iii. The web app shells out preliminary legal and corporate counselling through the smart document understanding system and geenral Q & A embedded within the Czar bot.

iv. The web app gives a better understanding of entity relations and dependencies to the user through the intricately designed interactive dashboard.

- **Disadvantages**

i. Although the web app provides preliminary solution to almost any problems that an user is expected to encounter, for more advanced stages of the same or different problems, the user might have to search beyond the scope of this web app.

5.3 Conclusion

It can be safely and economically concluded that, the web application has served its purpose of prediciting the number of orders, categorically for each combination of center and meal id, for atleast 11 weeks, i.e. for the given week and 10 weeks ahead of it. Theaccuracy is a stunning 76.2%
The web app also takes into account the much debated and essential mental healthcare and has initiated a willful step towards championing the cause of the same. The interactive dashboard and extensive business and governmental policy tips and updates, provide a whole new dimensional experience for the user, who might find it ever so more engaging to work with the web app.
The news feed section, combined with the videos and about section keeps the user updated on the new happenings all around the globe. Through the web app, apart from availing the core functionality of prediction, the user can avail these spectacular and interactive add ons and thus keep himself/herself engaged for a longer period of the business hours with much ease.

5.4. Future Scope

Although the current web application uses NODE RED UI platform when it comes to hosting, in the not so distant future, a flask app would seem more appropriate for the same, given the greater visual appeal of a flask app when compared to its NODE RED contemporary.
The Machine learning algorithm can also be switched with a more advanced algorithm of deep learning or AI, such as Recurrent Neural Networks or Time series forecasting, for better accuracy and smoother experience. The chatbot can be upgraded with more funtionalities.

Bibliogarphy

1. SmartBridge Bootcamp on the various related topics of the project.
2. NODE RED discussion forum.
3. IBM Documents, regarding the services in use, the user manuals.
4. IBM Cognitive Classes.

Appendix

A.1. Source code for the Machine Learning Notebook (Main Part):

```
5. Preprocess Data
# Drop rows whose target is not defined
target = target_label_name # your target name here
if learning_type == 'regression':
    df[target] = pd.to_numeric(df[target], errors='coerce')
df.dropna('rows', how='any', subset=[target], inplace=True)

# extract X and y
df_X = df.drop(columns=[target])
df_y = df[target]

# Detach preprocessing pipeline (which needs to see all
  training data)
preprocessor_index = -1
preprocessing_steps = []
for i, step in enumerate(pipeline.steps):
    preprocessing_steps.append(step)
    if step[0]=='preprocessor':
        preprocessor_index = i
        break
#if len(pipeline.steps) > preprocessor_index+1 and
```

```
  pipeline.steps[preprocessor_index + 1][0] == 'cognito':
21      #preprocessor_index += 1
22
  #preprocessing_steps.append(pipeline.steps[preprocessor_index
  ])
23 if preprocessor_index >= 0:
24      preprocessing_pipeline = Pipeline(memory=pipeline.memory,
  steps=preprocessing_steps)
25      pipeline =
  Pipeline(steps=pipeline.steps[preprocessor_index+1:])
26 # Preprocess X
27 # preprocessor should see all data for cross_validate on the
  remaining steps to match autoai scores
28 known_values_list.clear()  #  known_values_list is filled in
  by the preprocessing_pipeline if needed
29 preprocessing_pipeline.fit(df_X.values, df_y.values)
30 X_prep = preprocessing_pipeline.transform(df_X.values)
31 6. Split data into Training and Holdout sets
32 # determine learning_type and perform holdout split (stratify
  conditionally)
33 if learning_type is None:
34      # When the problem type is not available in the metadata,
  use the sklearn type_of_target to determine whether to
  stratify the holdout split
35      # Caution:  This can mis-classify regression targets that
  can be expressed as integers as multiclass, in which case
  manually override the learning_type
36      from sklearn.utils.multiclass import type_of_target
37      if type_of_target(df_y.values) in ['multiclass',
  'binary']:
38          learning_type = 'classification'
39      else:
40          learning_type = 'regression'
41      print('learning_type determined by type_of_target
  as:',learning_type)
42 else:
43      print('learning_type specified as:',learning_type)
```

```
44
45 from sklearn.model_selection import train_test_split
46 if learning_type == 'classification':
47     X, X_holdout, y, y_holdout = train_test_split(X_prep,
   df_y.values, test_size=holdout_fraction,
   random_state=random_state, stratify=df_y.values)
48 else:
49     X, X_holdout, y, y_holdout = train_test_split(X_prep,
   df_y.values, test_size=holdout_fraction,
   random_state=random_state)
50
51 7. Generate features via Feature Engineering pipeline
52 #Detach Feature Engineering pipeline if next, fit it, and
   transform the training data
53 fe_pipeline = None
54 if pipeline.steps[0][0] == 'cognito':
55     try:
56         fe_pipeline = Pipeline(steps=[pipeline.steps[0]])
57         X = fe_pipeline.fit_transform(X, y)
58         X_holdout = fe_pipeline.transform(X_holdout)
59         pipeline.steps = pipeline.steps[1:]
60     except IndexError:
61         try:
62             print('Trying to compose pipeline with some of
   cognito steps')
63             fe_pipeline = Pipeline(steps =
   list([pipeline.steps[0][1].steps[0],pipeline.steps[0][1].step
   s[1]]))
64             X = fe_pipeline.fit_transform(X, y)
65             X_holdout = fe_pipeline.transform(X_holdout)
66             pipeline.steps = pipeline.steps[1:]
67         except IndexError:
68             print('Composing pipeline without cognito
   steps!')
69             pipeline.steps = pipeline.steps[1:]
70 9. Fit pipeline, predict on Holdout set, calculate score,
```

```
  perform cross-validation
# fit the remainder of the pipeline on the training data
pipeline.fit(X,y)
# predict on the holdout data
y_pred = pipeline.predict(X_holdout)
# compute score for the optimization metric
# scorer may need pos_label, but not all scorers take
  pos_label parameter
from sklearn.metrics import get_scorer
scorer = get_scorer(optimization_metric)
score = None
#score = scorer(pipeline, X_holdout, y_holdout)  # this would
  suffice for simple cases
pos_label = None  # if you want to supply the pos_label,
  specify it here
if pos_label is None and 'pos_label' in _input_metadata:
    pos_label=_input_metadata['pos_label']
try:
    score = scorer(pipeline, X_holdout, y_holdout)
except Exception as e1:
    if pos_label is None or str(pos_label)=='':
        print('You may have to provide a value for pos_label
  in order for a score to be calculated.')
        raise(e1)
    else:
        exception_string=str(e1)
        if 'pos_label' in exception_string:
            try:
                scorer = make_pos_label_scorer(scorer,
  pos_label=pos_label)
                score = scorer(pipeline, X_holdout,
  y_holdout)
                print('Retry was successful with pos_label
  supplied to scorer')
            except Exception as e2:
                print('Initial attempt to use scorer failed.
  Exception was:')
```

```
                print(e1)
                print('')
                print('Retry with pos_label failed.
  Exception was:')
                print(e2)
        else:
            raise(e1)

if score is not None:
    print(score)
# cross_validate pipeline using training data
from sklearn.model_selection import cross_validate
from sklearn.model_selection import StratifiedKFold, KFold
if learning_type == 'classification':
    fold_generator = StratifiedKFold(n_splits=cv_num_folds,
  random_state=random_state)
else:
    fold_generator = KFold(n_splits=cv_num_folds,
  random_state=random_state)
cv_results = cross_validate(pipeline, X, y,
  cv=fold_generator, scoring={optimization_metric:scorer},
  return_train_score=True)
import numpy as np
np.mean(cv_results['test_' + optimization_metric])
cv_results
```