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.

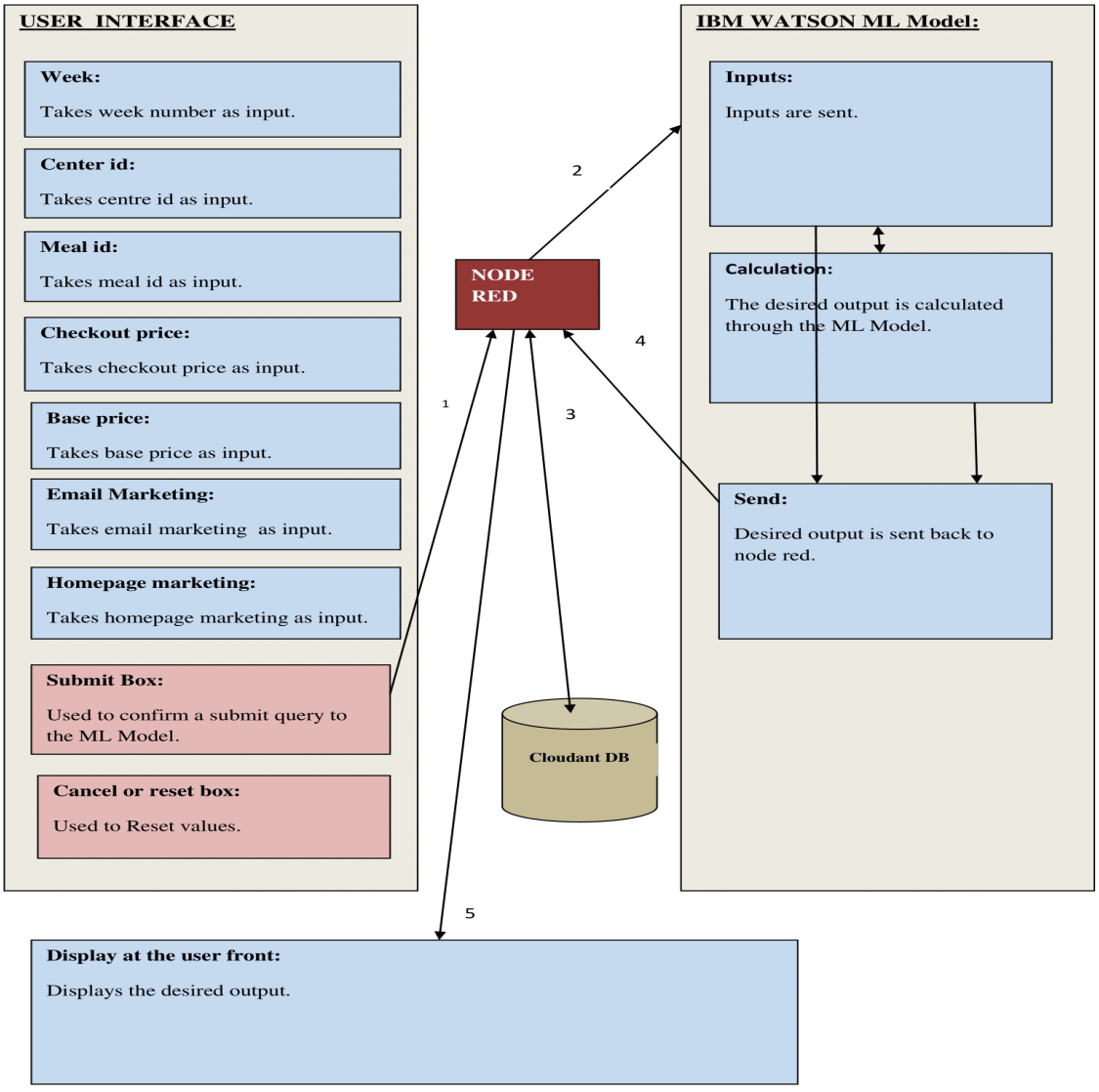
Chapter 3: Theoritical Analysis

3.1. Block Diagram

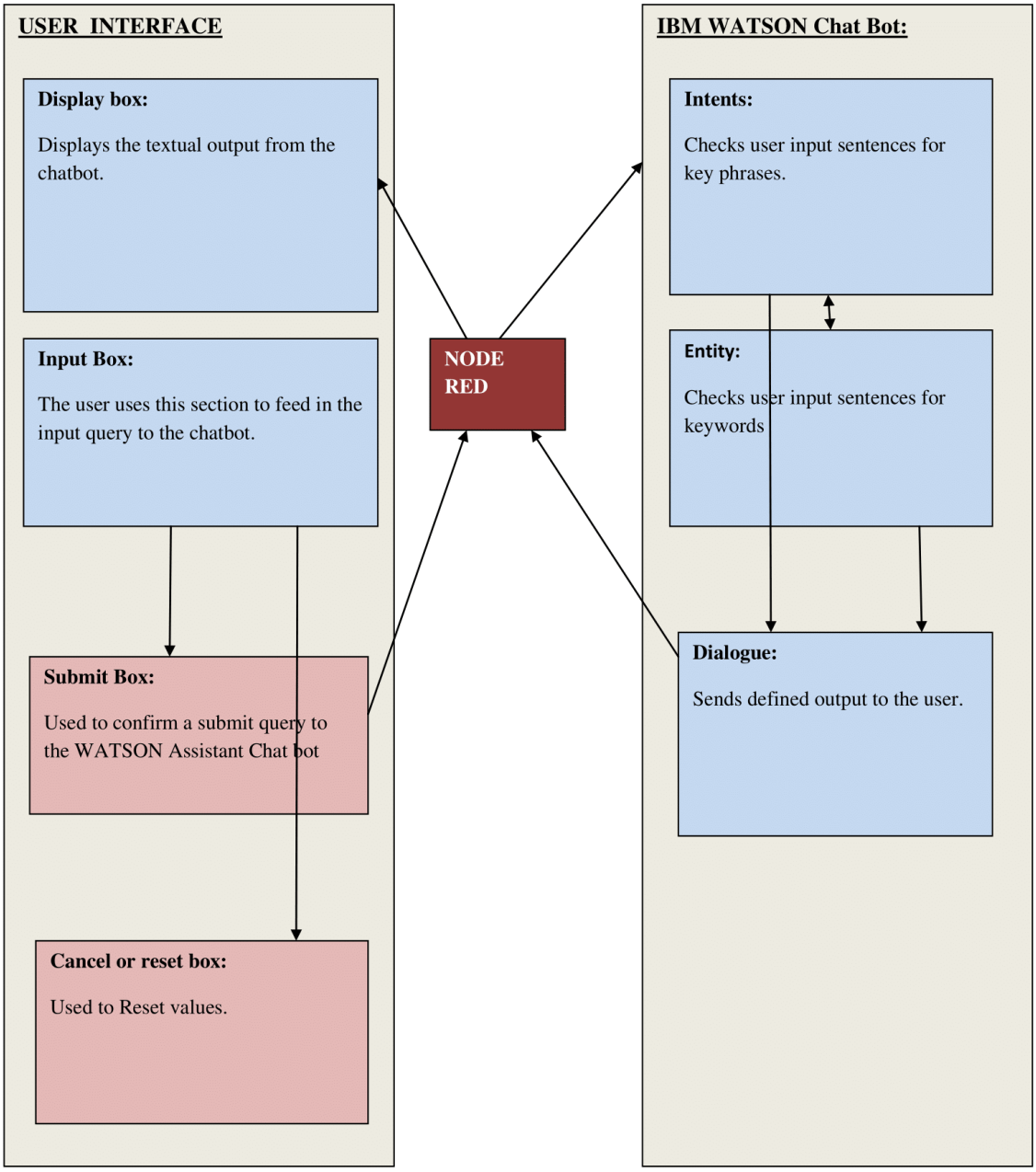
Although there are several small components in the entire web application, but the two main parts which form the bulk of the functioning and almost all of the interative user interface, are,

- 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.

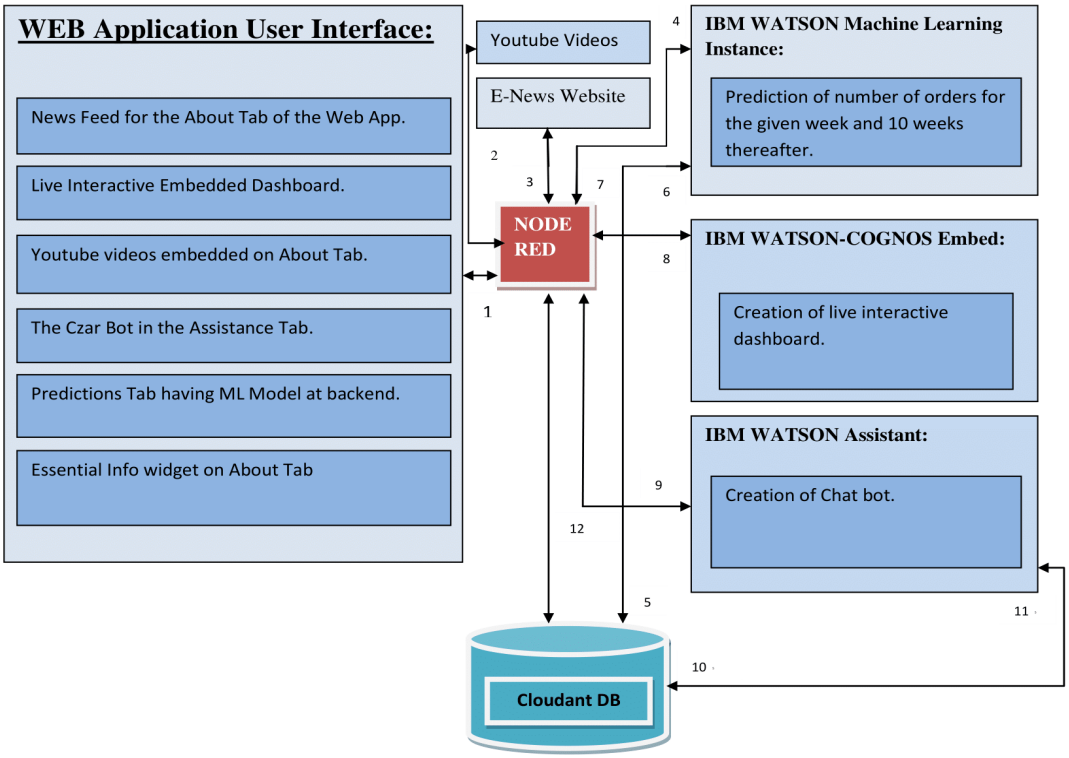
Youtube Video:

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

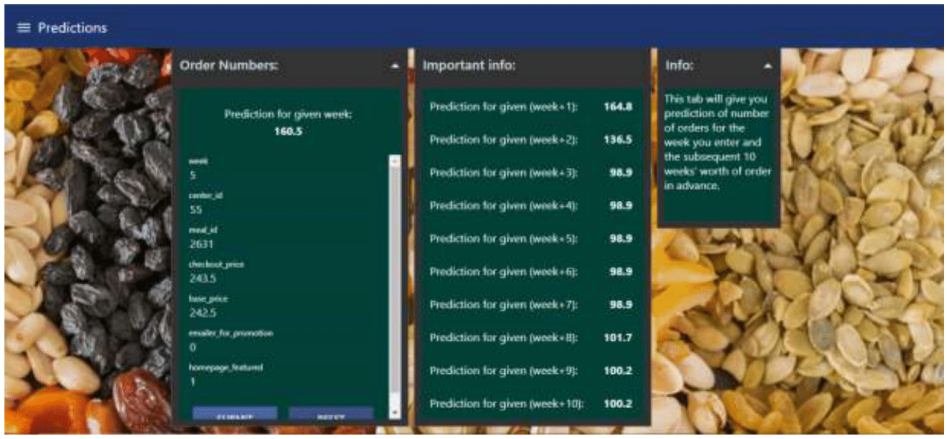
Experimental investigation done over a wide range of experimental data and subsequent output only go on to show that,

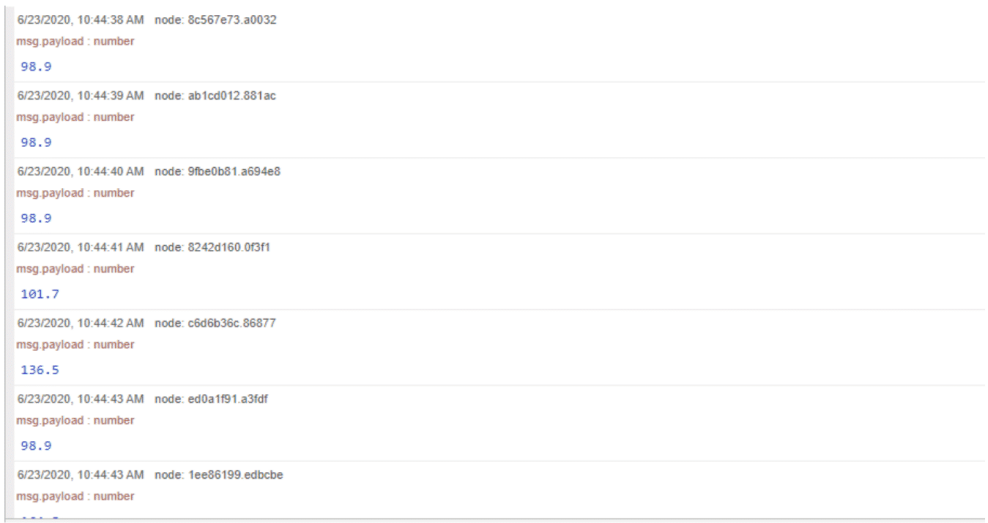
- Meal-id 2290, has the highest number of orders almost throughout the year, in almost all of the centers, whereas Meal-id 177 has the lowest number of orders in almost all of the centers and that too throughout the year.
- Center-id 10 has the highest rating amongst the other centers available for this 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.

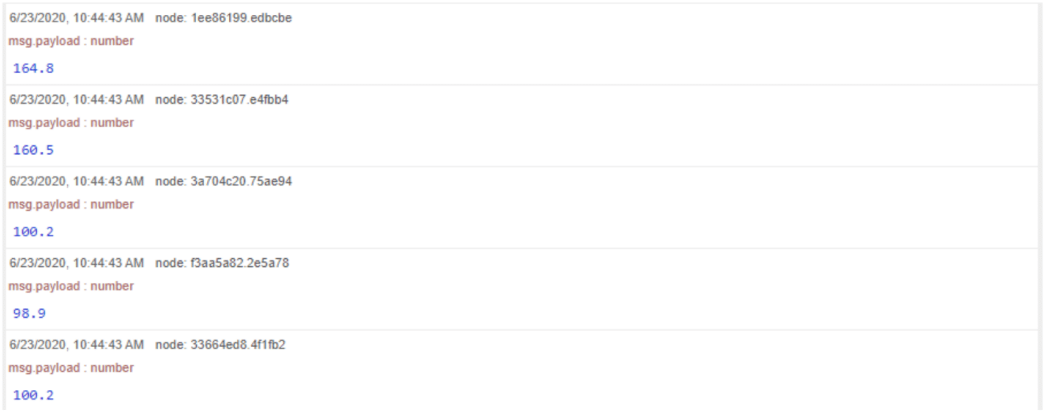
Chapter 5

5.1. Result

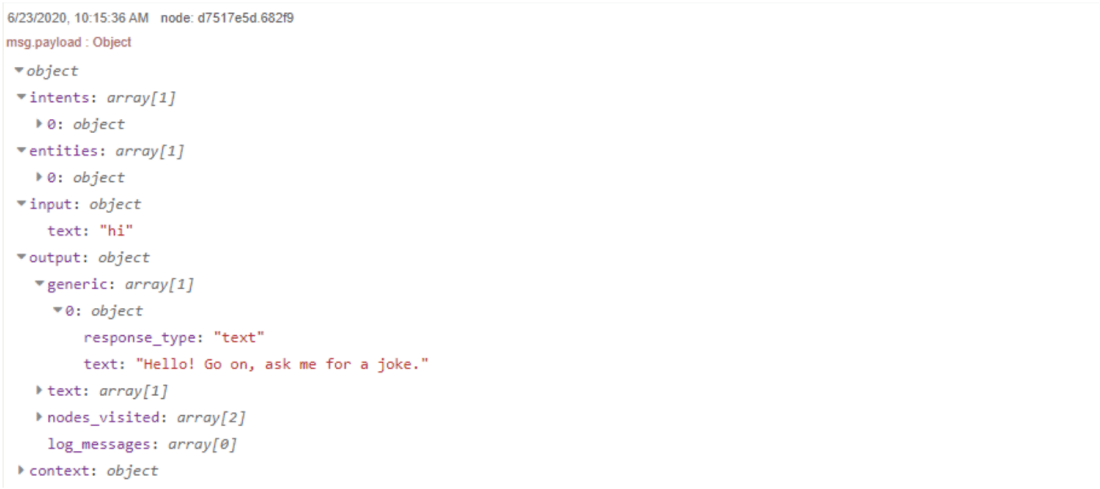
Results for Machine Learning model, from the debug section of NODE RED flow.

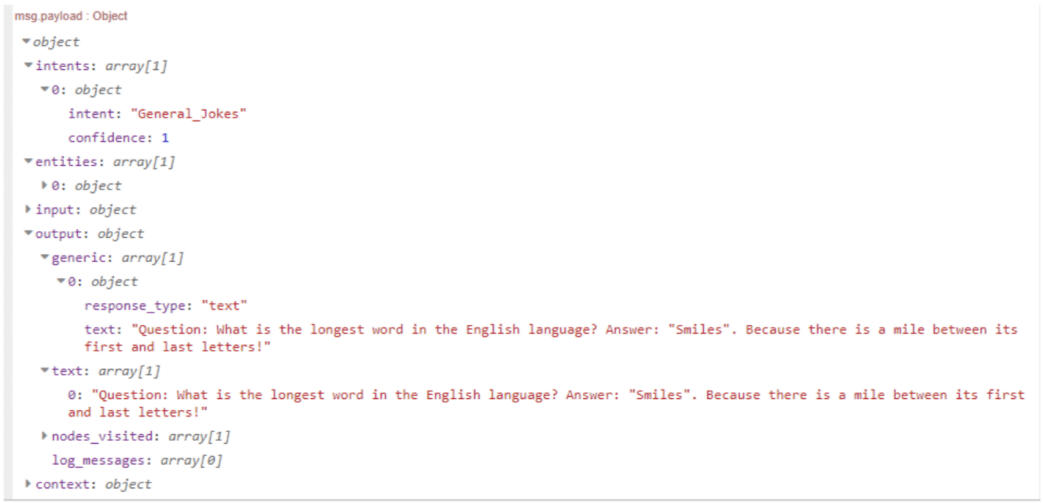


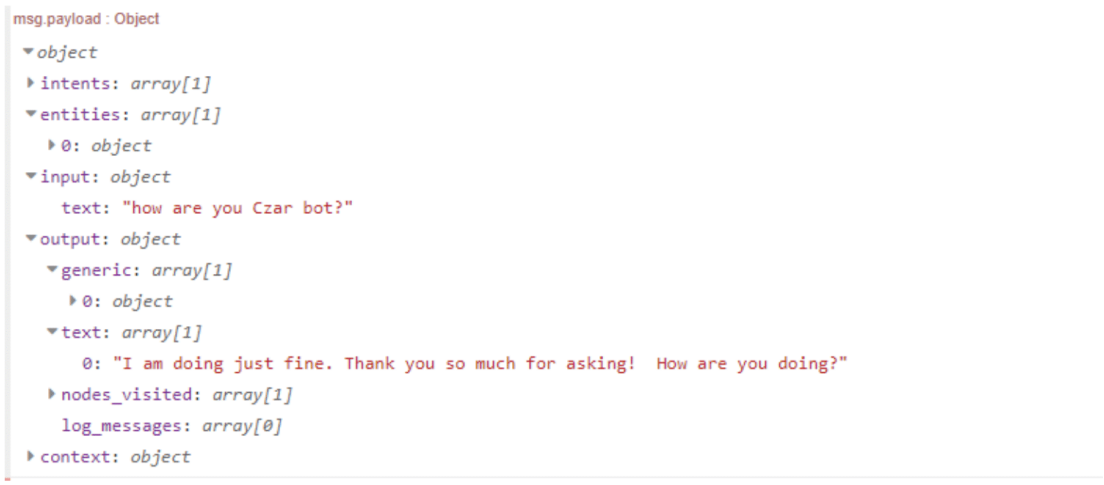




Chat bot results from NODE RED Debug section.




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:

```
1. Set Up
In [ ]:
try:
    import autoai_libs
except Exception as e:
    import subprocess
    out = subprocess.check_output('pip install autoai-libs'.split(' '))
    for line in out.splitlines():
        print(line)
    import autoai_libs
import sklearn
try:
    import xgboost
except:
    print('xgboost, if needed, will be installed and imported later')
try:
    import lightgbm
except:
    print('lightgbm, if needed, will be installed and
```

```
  imported later')
from sklearn.cluster import FeatureAgglomeration
import numpy
from numpy import inf, nan, dtype, mean
from autoai_libs.sklearn.custom_scorers import CustomScorers
import sklearn.ensemble
from autoai_libs.cognito.transforms.transform_utils import
  TExtras, FC
from autoai_libs.transformers.exportable import *
from autoai_libs.utils.exportable_utils import *
from sklearn.pipeline import Pipeline
known_values_list=[]
In [ ]:
# compose a decorator to assist pipeline instantiation via
  import of modules and installation of packages
def decorator_retries(func):
    def install_import_retry(*args, **kwargs):
        retries = 0
        successful = False
        failed_retries = 0
        while retries < 100 and failed_retries < 10 and not
  successful:
            retries += 1
            failed_retries += 1
            try:
                result = func(*args, **kwargs)
                successful = True
            except Exception as e:
                estr = str(e)
                if estr.startswith('name ') and
  estr.endswith(' is not defined'):
                    try:
                        import importlib
                        module_name = estr.split("'")[1]
                        module =
  importlib.import_module(module_name)
                        globals().update({module_name:
```

```
module})
                        print('import successful for ' +
module_name)
                        failed_retries -= 1
                    except Exception as import_failure:
                        print('import of ' + module_name + '
failed with: ' + str(import_failure))
                        import subprocess
                        if module_name == 'lightgbm':
                            try:
                                print('attempting pip install
of ' + module_name)
                                process =
subprocess.Popen('pip install ' + module_name, shell=True)
                                process.wait()
                            except Exception as E:
                                print(E)
                                try:
                                    import sys
                                    print('attempting conda
install of ' + module_name)
                                    process =
subprocess.Popen('conda install --yes --prefix {sys.prefix} -c
powerai ' + module_name, shell = True)
                                    process.wait()
                                except Exception as
lightgbm_installation_error:
                                    print('lightgbm
installation failed!' + lightgbm_installation_error)
                        else:
                            print('attempting pip install of
' + module_name)
                            process = subprocess.Popen('pip
install ' + module_name, shell=True)
                            process.wait()
                        try:
                            print('re-attempting import of '
```

```
    + module_name)
76                              module =
    importlib.import_module(module_name)
77                              globals().update({module_name:
    module})
78                              print('import successful for ' +
    module_name)
79                              failed_retries -= 1
80                          except Exception as
    import_or_installation_failure:
81                              print('failure installing and/or
    importing ' + module_name + ' error was: ' + str(
82
    import_or_installation_failure))
83                              raise
    (ModuleNotFoundError('Missing package in environment for ' +
    module_name +
84                                                          '? Try
    import and/or pip install manually?'))
85                  elif type(e) is AttributeError:
86                      if 'module ' in estr and ' has no
    attribute ' in estr:
87                          pieces = estr.split("'")
88                          if len(pieces) == 5:
89                              try:
90                                  import importlib
91                                  print('re-attempting import
    of ' + pieces[3] + ' from ' + pieces[1])
92                                  module =
    importlib.import_module('.' + pieces[3], pieces[1])
93                                  failed_retries -= 1
94                              except:
95                                  print('failed attempt to
    import ' + pieces[3])
96                                  raise (e)
97                          else:
98                              raise (e)
```

```
99                 else:
100                     raise (e)
101         if successful:
102             print('Pipeline successfully instantiated')
103         else:
104             raise (ModuleNotFoundError(
105                 'Remaining missing imports/packages in
  environment? Retry cell and/or try pip install manually?'))
106         return result
107     return install_import_retry
108 2. Compose Pipeline
109 In [ ]:
110 # metadata necessary to replicate AutoAI scores with the
  pipeline
111 _input_metadata = {'target_label_name': 'num_orders',
  'learning_type': 'regression', 'run_uid':
  '94f92dbf-657b-42a7-bc44-3edd70d2fea2', 'pn': 'P3',
  'cv_num_folds': 3, 'holdout_fraction': 0.1,
  'optimization_metric': 'neg_root_mean_squared_error',
  'pos_label': None, 'random_state': 33, 'data_source': ''}
112
113 # define a function to compose the pipeline, and invoke it
114 @decorator_retries
115 def compose_pipeline():
116     import numpy
117     from numpy import nan, dtype, mean
118     #
119     # composing steps for toplevel Pipeline
120     #
121     _input_metadata = {'target_label_name': 'num_orders',
  'learning_type': 'regression', 'run_uid':
  '94f92dbf-657b-42a7-bc44-3edd70d2fea2', 'pn': 'P3',
  'cv_num_folds': 3, 'holdout_fraction': 0.1,
  'optimization_metric': 'neg_root_mean_squared_error',
  'pos_label': None, 'random_state': 33, 'data_source': ''}
122     steps = []
```

```
123       #
124       # composing steps for preprocessor Pipeline
125       #
126       preprocessor__input_metadata = None
127       preprocessor_steps = []
128       #
129       # composing steps for preprocessor_features
    FeatureUnion
130       #
131       preprocessor_features_transformer_list = []
132       #
133       # composing steps for
    preprocessor_features_categorical Pipeline
134       #
135       preprocessor_features_categorical__input_metadata =
    None
136       preprocessor_features_categorical_steps = []
137
    preprocessor_features_categorical_steps.append(('cat_column_s
    elector',
    autoai_libs.transformers.exportable.NumpyColumnSelector(colum
    ns=[0, 1, 2, 5, 6])))
138
    preprocessor_features_categorical_steps.append(('cat_compress
    _strings',
    autoai_libs.transformers.exportable.CompressStrings(activate_
    flag=True, compress_type='hash', dtypes_list=['float_int_num',
    'float_int_num', 'float_int_num', 'float_int_num',
    'float_int_num'], missing_values_reference_list=['', '-', '?',
    nan], misslist_list=[[], [], [], [], []])))
139
    preprocessor_features_categorical_steps.append(('cat_missing_
    replacer',
    autoai_libs.transformers.exportable.NumpyReplaceMissingValues
    (filling_values=100001, missing_values=[])))
140
    preprocessor_features_categorical_steps.append(('cat_unknown_
```

```
    replacer',
    autoai_libs.transformers.exportable.NumpyReplaceUnknownValues
    (filling_values=100001, filling_values_list=[100001, 100001,
    100001, 100001, 100001], known_values_list=[[1.0, 2.0, 3.0,
    4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0,
    15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0,
    25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0,
    35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 43.0, 44.0,
    45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0,
    55.0], [10.0, 11.0, 13.0, 14.0, 17.0, 20.0, 23.0, 24.0, 26.0,
    27.0, 29.0, 30.0, 32.0, 34.0, 36.0, 39.0, 41.0, 42.0, 43.0,
    50.0, 51.0, 52.0, 53.0, 55.0, 57.0, 58.0, 59.0, 61.0, 64.0,
    65.0, 66.0, 67.0, 68.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.0,
    80.0, 81.0, 83.0, 86.0, 88.0, 89.0, 91.0, 92.0, 93.0, 94.0,
    97.0, 99.0, 101.0, 102.0, 104.0, 106.0, 108.0, 109.0, 110.0,
    113.0, 124.0, 126.0, 129.0, 132.0, 137.0, 139.0, 143.0,
    145.0, 146.0, 149.0, 152.0, 153.0, 157.0, 161.0, 162.0,
    174.0, 177.0, 186.0], [1062.0, 1109.0, 1198.0, 1207.0,
    1216.0, 1230.0, 1247.0, 1248.0, 1311.0, 1438.0, 1445.0,
    1525.0, 1543.0, 1558.0, 1727.0, 1754.0, 1770.0, 1778.0,
    1803.0, 1847.0, 1878.0, 1885.0, 1902.0, 1962.0, 1971.0,
    1993.0, 2126.0, 2139.0, 2290.0, 2304.0, 2306.0, 2322.0,
    2444.0, 2490.0, 2492.0, 2494.0, 2539.0, 2569.0, 2577.0,
    2581.0, 2631.0, 2640.0, 2664.0, 2704.0, 2707.0, 2760.0,
    2826.0, 2867.0], [0.0, 1.0], [0.0, 1.0]],
    missing_values_reference_list=['', '-', '?', nan])))
141
    preprocessor_features_categorical_steps.append(('boolean2float
    _transformer',
    autoai_libs.transformers.exportable.boolean2float(activate_flag
    =True)))
142
    preprocessor_features_categorical_steps.append(('cat_imputer'
    ,
    autoai_libs.transformers.exportable.CatImputer(activate_flag=T
    rue, missing_values=100001, sklearn_version_family='20',
    strategy='most_frequent')))
```

```
143
  preprocessor_features_categorical_steps.append(('cat_encoder'
  ,
  autoai_libs.transformers.exportable.CatEncoder(activate_flag=T
  rue, categories='auto', dtype=numpy.float64,
  encoding='ordinal', handle_unknown='error',
  sklearn_version_family='20')))
144
  preprocessor_features_categorical_steps.append(('float32_trans
  former',
  autoai_libs.transformers.exportable.float32_transform(activate
  _flag=True)))
145       # assembling preprocessor_features_categorical_
  Pipeline
146       preprocessor_features_categorical_pipeline =
  sklearn.pipeline.Pipeline(steps=preprocessor_features_categor
  ical_steps)
147
  preprocessor_features_transformer_list.append(('categorical',
  preprocessor_features_categorical_pipeline))
148       #
149       # composing steps for preprocessor_features_numeric
  Pipeline
150       #
151       preprocessor_features_numeric__input_metadata = None
152       preprocessor_features_numeric_steps = []
153
  preprocessor_features_numeric_steps.append(('num_column_selec
  tor',
  autoai_libs.transformers.exportable.NumpyColumnSelector(colum
  ns=[3, 4])))
154
  preprocessor_features_numeric_steps.append(('num_floatstr2float
  _transformer',
  autoai_libs.transformers.exportable.FloatStr2Float(activate_fl
  ag=True, dtypes_list=['float_num', 'float_num'],
```

```
    missing_values_reference_list=[])))
155
    preprocessor_features_numeric_steps.append(('num_missing_repl
    acer',
    autoai_libs.transformers.exportable.NumpyReplaceMissingValues
    (filling_values=nan, missing_values=[])))
156
    preprocessor_features_numeric_steps.append(('num_imputer',
    autoai_libs.transformers.exportable.NumImputer(activate_flag=T
    rue, missing_values=nan, strategy='median')))
157
    preprocessor_features_numeric_steps.append(('num_scaler',
    autoai_libs.transformers.exportable.OptStandardScaler(num_sca
    ler_copy=None, num_scaler_with_mean=None,
    num_scaler_with_std=None, use_scaler_flag=False)))
158
    preprocessor_features_numeric_steps.append(('float32_transform
    er',
    autoai_libs.transformers.exportable.float32_transform(activate
    _flag=True)))
159         # assembling preprocessor_features_numeric_ Pipeline
160         preprocessor_features_numeric_pipeline =
    sklearn.pipeline.Pipeline(steps=preprocessor_features_numeric
    _steps)
161
    preprocessor_features_transformer_list.append(('numeric',
    preprocessor_features_numeric_pipeline))
162         # assembling preprocessor_features_ FeatureUnion
163         preprocessor_features_pipeline =
    sklearn.pipeline.FeatureUnion(transformer_list=preprocessor_f
    eatures_transformer_list)
164         preprocessor_steps.append(('features',
    preprocessor_features_pipeline))
165         preprocessor_steps.append(('permuter',
    autoai_libs.transformers.exportable.NumpyPermuteArray(axis=0,
    permutation_indices=[0, 1, 2, 5, 6, 3, 4])))
166         # assembling preprocessor_ Pipeline
```

```
167       preprocessor_pipeline =
    sklearn.pipeline.Pipeline(steps=preprocessor_steps)
168       steps.append(('preprocessor', preprocessor_pipeline))
169       #
170       # composing steps for cognito Pipeline
171       #
172       cognito__input_metadata = None
173       cognito_steps = []
174       cognito_steps.append(('0',
    autoai_libs.cognito.transforms.transform_utils.TA1(fun=numpy.
    tan, name='tan', datatypes=['float'],
    feat_constraints=[autoai_libs.utils.fc_methods.is_not_categor
    ical], tgraph=None, apply_all=True, col_names=['week',
    'center_id', 'meal_id', 'checkout_price', 'base_price',
    'emailer_for_promotion', 'homepage_featured'],
    col_dtypes=[dtype('float32'), dtype('float32'),
    dtype('float32'), dtype('float32'), dtype('float32'),
    dtype('float32'), dtype('float32')],
    col_as_json_objects=None)))
175       cognito_steps.append(('1',
    autoai_libs.cognito.transforms.transform_utils.FS1(cols_ids_m
    ust_keep=range(0, 7), additional_col_count_to_keep=8,
    ptype='regression')))
176       cognito_steps.append(('2',
    autoai_libs.cognito.transforms.transform_utils.TA1(fun=numpy.
    sin, name='sin', datatypes=['float'],
    feat_constraints=[autoai_libs.utils.fc_methods.is_not_categor
    ical], tgraph=None, apply_all=True, col_names=['week',
    'center_id', 'meal_id', 'checkout_price', 'base_price',
    'emailer_for_promotion', 'homepage_featured', 'tan(week)',
    'tan(center_id)', 'tan(meal_id)', 'tan(checkout_price)',
    'tan(base_price)'], col_dtypes=[dtype('float32'),
    dtype('float32'), dtype('float32'), dtype('float32'),
    dtype('float32'), dtype('float32'), dtype('float32'),
    dtype('float32'), dtype('float32'), dtype('float32'),
    dtype('float32'), dtype('float32')],
```

```
   col_as_json_objects=None)))
177       cognito_steps.append(('3',
   autoai_libs.cognito.transforms.transform_utils.FS1(cols_ids_m
   ust_keep=range(0, 7), additional_col_count_to_keep=8,
   ptype='regression')))
178       cognito_steps.append(('4',
   autoai_libs.cognito.transforms.transform_utils.TA1(fun=numpy.
   rint, name='round', datatypes=['numeric'],
   feat_constraints=[autoai_libs.utils.fc_methods.is_not_categor
   ical], tgraph=None, apply_all=True, col_names=['week',
   'center_id', 'meal_id', 'checkout_price', 'base_price',
   'emailer_for_promotion', 'homepage_featured', 'tan(meal_id)',
   'tan(base_price)', 'sin(week)', 'sin(meal_id)',
   'sin(checkout_price)', 'sin(base_price)',
   'sin(tan(center_id))', 'sin(tan(meal_id))'],
   col_dtypes=[dtype('float32'), dtype('float32'),
   dtype('float32'), dtype('float32'), dtype('float32'),
   dtype('float32'), dtype('float32'), dtype('float32'),
   dtype('float32'), dtype('float32'), dtype('float32'),
   dtype('float32'), dtype('float32'), dtype('float32'),
   dtype('float32')], col_as_json_objects=None)))
179       cognito_steps.append(('5',
   autoai_libs.cognito.transforms.transform_utils.FS1(cols_ids_m
   ust_keep=range(0, 7), additional_col_count_to_keep=8,
   ptype='regression')))
180       # assembling cognito_ Pipeline
181       cognito_pipeline =
   sklearn.pipeline.Pipeline(steps=cognito_steps)
182       steps.append(('cognito', cognito_pipeline))
183       steps.append(('estimator',
   sklearn.ensemble.forest.RandomForestRegressor(bootstrap=True,
   criterion='mse', max_depth=None, max_features='auto',
   max_leaf_nodes=None, min_impurity_decrease=0.0,
   min_impurity_split=None, min_samples_leaf=1,
   min_samples_split=2, min_weight_fraction_leaf=0.0,
   n_estimators=10, n_jobs=2, oob_score=True, random_state=33,
   verbose=0, warm_start=False)))
```

```
184       # assembling  Pipeline
185       pipeline = sklearn.pipeline.Pipeline(steps=steps)
186       return pipeline
187   pipeline = compose_pipeline()
188   3. Extract needed parameter values from AutoAI run
    metadata
189   In [ ]:
190   # Metadata used in retrieving data and computing metrics.
    Customize as necessary for your environment.
191   #data_source='replace_with_path_and_csv_filename'
192   target_label_name = _input_metadata['target_label_name']
193   learning_type = _input_metadata['learning_type']
194   optimization_metric =
    _input_metadata['optimization_metric']
195   random_state = _input_metadata['random_state']
196   cv_num_folds = _input_metadata['cv_num_folds']
197   holdout_fraction = _input_metadata['holdout_fraction']
198   if 'data_provenance' in _input_metadata:
199       data_provenance = _input_metadata['data_provenance']
200   else:
201       data_provenance = None
202   if 'pos_label' in _input_metadata and learning_type ==
    'classification':
203       pos_label = _input_metadata['pos_label']
204   else:
205       pos_label = None
206   4. Create dataframe from dataset in Cloud Object Storage
207   In [ ]:
208   # @hidden_cell
209   # The following code contains the credentials for a file in
    your IBM Cloud Object Storage.
210   # You might want to remove those credentials before you
    share your notebook.
211   credentials_0 = {
212       'ENDPOINT':
    'https://s3-api.us-geo.objectstorage.softlayer.net',
213       'IBM_AUTH_ENDPOINT':
```

```
    'https://iam.bluemix.net/oidc/token/',
214         'APIKEY':
    '_wsK86uTdRgevFlmjANklp0tzMyuvUut6s7ddVqjLYKj',
215         'BUCKET': 'optfood-donotdelete-pr-tpuulakeaudnfv',
216         'FILE': 'FoodProj.csv',
217         'SERVICE_NAME': 's3',
218         'ASSET_ID': '1',
219         }
220   In [ ]:
221   #  Read the data as a dataframe
222   import pandas as pd
223
224   csv_encodings=['UTF-8','Latin-1'] # supplement list of
    encodings as necessary for your data
225   df = None
226   readable = None  # if automatic detection fails, you can
    supply a filename here
227
228   # First, obtain a readable object
229   # Cloud Object Storage data access
230   # Assumes COS credentials are in a dictionary named
    'credentials_0'
231
232   credentials = df = globals().get('credentials_0')
233   if readable is None and credentials is not None :
234       try:
235           import types
236           import pandas as pd
237           import io
238           import os
239       except Exception as import_exception:
240           print('Error with importing packages - check if
    you installed them on your environment')
241       try:
242           if credentials['SERVICE_NAME'] == 's3':
243               try:
244                   from botocore.client import Config
```

```
245                 import ibm_boto3
246             except Exception as import_exception:
247                 print('Installing required packages!')
248                 !pip install ibm-cos-sdk
249                 print('accessing data via Cloud Object
  Storage')
250             try:
251                 cos_client =
  ibm_boto3.resource(service_name=credentials['SERVICE_NAME'],
252
  ibm_api_key_id=credentials['APIKEY'],
253
  ibm_auth_endpoint=credentials['IBM_AUTH_ENDPOINT'],
254
  config=Config(signature_version='oauth'),
255
  endpoint_url=credentials['ENDPOINT'])
256             except Exception as cos_exception:
257                 print('unable to create client for cloud
  object storage')
258             try:
259
  cos_client.meta.client.download_file(Bucket=credentials['BUCKE
  T'], Filename=credentials['FILE'], Key=credentials['FILE'])
260             except Exception as cos_access_exception:
261                 print('unable to access data object in
  cloud object storage with credentials supplied')
262             try:
263                 for encoding in csv_encodings:
264                     df = pd.read_csv(credentials['FILE'],
  encoding = encoding, sep = None, engine = 'python')
265                     os.remove(credentials['FILE'])
266                     print('Data loaded from cloud object
  storage with encoding ' + encoding)
267                     break
268             except Exception as cos_object_read_exception:
269                 print('unable to access data object from
```

```
  cos object with encoding ' + encoding)
270           elif credentials['SERVICE_NAME'] == 'fs':
271               print('accessing data via File System')
272               try:
273                   df = pd.read_csv(credentials['FILE'], sep
  = None, engine = 'python')
274               except Exception as FS_access_exception:
275                   print('unable to access data object in
  File System with path supplied')
276       except Exception as data_access_exception:
277           print('unable to access data object with
  credentials supplied')
278
279   # IBM Cloud Pak for Data data access
280   project_filename = globals().get('project_filename')
281   if readable is None and 'credentials_0' in globals() and
  'ASSET_ID' in credentials_0:
282       project_filename = credentials_0['ASSET_ID']
283   if project_filename != 'None' and project_filename != '1':
284       print('attempting project_lib access to ' +
  str(project_filename))
285       try:
286           from project_lib import Project
287           project = Project.access()
288           storage_credentials =
  project.get_storage_metadata()
289           readable = project.get_file(project_filename)
290       except Exception as project_exception:
291           print('unable to access data using the project_lib
  interface and filename supplied')
292
293
294   # Use data_provenance as filename if other access
  mechanisms are unsuccessful
295   if readable is None and type(data_provenance) is str:
296       print('attempting to access local file using path and
  name ' + data_provenance)
```

```
297      readable = data_provenance
298
299  # Second, use pd.read_csv to read object, iterating over
   list of csv_encodings until successful
300  if readable is not None:
301      for encoding in csv_encodings:
302          try:
303              df = pd.read_csv(readable, encoding=encoding,
   sep = None, engine = 'python')
304              print('successfully loaded dataframe using
   encoding = ' + str(encoding))
305              break
306          except Exception as exception_csv:
307              print('unable to read csv using encoding ' +
   str(encoding))
308              print('handled error was ' +
   str(exception_csv))
309      if df is None:
310          print('unable to read file/object as a dataframe
   using supplied csv_encodings ' + str(csv_encodings))
311          print(f'Please use \'insert to code\' on data
   panel to load dataframe.')
312          raise(ValueError('unable to read file/object as a
   dataframe using supplied csv_encodings ' +
   str(csv_encodings)))
313
314  if isinstance(df,pd.DataFrame):
315      print('Data loaded succesfully')
316  5. Preprocess Data
317  In [ ]:
318  # Drop rows whose target is not defined
319  target = target_label_name # your target name here
320  if learning_type == 'regression':
321      df[target] = pd.to_numeric(df[target],
   errors='coerce')
322  df.dropna('rows', how='any', subset=[target],
```

```
  inplace=True)
In [ ]:
# extract X and y
df_X = df.drop(columns=[target])
df_y = df[target]
In [ ]:
# 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':
    #preprocessor_index += 1

#preprocessing_steps.append(pipeline.steps[preprocessor_index])
if preprocessor_index >= 0:
    preprocessing_pipeline = Pipeline(memory=pipeline.memory, steps=preprocessing_steps)
    pipeline = Pipeline(steps=pipeline.steps[preprocessor_index+1:])
In [ ]:
# Preprocess X
# preprocessor should see all data for cross_validate on the remaining steps to match autoai scores
known_values_list.clear()  #  known_values_list is filled in by the preprocessing_pipeline if needed
preprocessing_pipeline.fit(df_X.values, df_y.values)
X_prep = preprocessing_pipeline.transform(df_X.values)
6. Split data into Training and Holdout sets
In [ ]:
# determine learning_type and perform holdout split
```

```
   (stratify conditionally)
351  if learning_type is None:
352      # When the problem type is not available in the
   metadata, use the sklearn type_of_target to determine whether
   to stratify the holdout split
353      # Caution:  This can mis-classify regression targets
   that can be expressed as integers as multiclass, in which
   case manually override the learning_type
354      from sklearn.utils.multiclass import type_of_target
355      if type_of_target(df_y.values) in ['multiclass',
   'binary']:
356          learning_type = 'classification'
357      else:
358          learning_type = 'regression'
359      print('learning_type determined by type_of_target
   as:',learning_type)
360  else:
361      print('learning_type specified as:',learning_type)
362
363  from sklearn.model_selection import train_test_split
364  if learning_type == 'classification':
365      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)
366  else:
367      X, X_holdout, y, y_holdout = train_test_split(X_prep,
   df_y.values, test_size=holdout_fraction,
   random_state=random_state)
368  7. Generate features via Feature Engineering pipeline
369  In [ ]:
370  #Detach Feature Engineering pipeline if next, fit it, and
   transform the training data
371  fe_pipeline = None
372  if pipeline.steps[0][0] == 'cognito':
373      try:
374          fe_pipeline = Pipeline(steps=[pipeline.steps[0]])
375          X = fe_pipeline.fit_transform(X, y)
```

```
            X_holdout = fe_pipeline.transform(X_holdout)
            pipeline.steps = pipeline.steps[1:]
        except IndexError:
            try:
                print('Trying to compose pipeline with some of cognito steps')
                fe_pipeline = Pipeline(steps = list([pipeline.steps[0][1].steps[0],pipeline.steps[0][1].steps[1]]))
                X = fe_pipeline.fit_transform(X, y)
                X_holdout = fe_pipeline.transform(X_holdout)
                pipeline.steps = pipeline.steps[1:]
            except IndexError:
                print('Composing pipeline without cognito steps!')
                pipeline.steps = pipeline.steps[1:]
8. Additional setup: Define a function that returns a scorer for the target's positive label
In [ ]:
# create a function to produce a scorer for a given positive label
def make_pos_label_scorer(scorer, pos_label):
    kwargs = {'pos_label':pos_label}
    for prop in ['needs_proba', 'needs_threshold']:
        if prop+'=True' in scorer._factory_args():
            kwargs[prop] = True
    if scorer._sign == -1:
        kwargs['greater_is_better'] = False
    from sklearn.metrics import make_scorer
    scorer=make_scorer(scorer._score_func, **kwargs)
    return scorer
9. Fit pipeline, predict on Holdout set, calculate score, perform cross-validation
In [ ]:
# fit the remainder of the pipeline on the training data
pipeline.fit(X,y)
```

```
In [ ]:
# predict on the holdout data
y_pred = pipeline.predict(X_holdout)
In [ ]:
# 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)
In [ ]:
# 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])
In [ ]:
cv_results
```

A.2. Source Code for the main NODE RED flow:

```
[{"id":"42e7756f.e5d5fc","type":"ui_template","z":"aab06066.8
dbcb","group":"bd20e955.8237f8","name":"","order":0,"width":0
,"height":0,"format":"<p>\n    This tab will give you
prediction of number of orders for the week you enter and the
```

```
subsequent 10 weeks' worth of order in
advance.\n</p>","storeOutMessages":true,"fwdInMessages":true,
"resendOnRefresh":true,"templateScope":"local","x":720,"y":12
20,"wires":[[]]},{"id":"bd20e955.8237f8","type":"ui_group","z
":"","name":"Info:","tab":"8cee228d.c1fbc","order":4,"disp":t
rue,"width":"3","collapse":true},{"id":"8cee228d.c1fbc","type
":"ui_tab","z":"","name":"Predictions","icon":"dashboard","or
der":3,"disabled":false,"hidden":false}]
```