Project Scope Document

Project Title: Optimized Warehouse Management Of Perishable Goods For A Food Delivery Company

Date Prepared: 30/06/2020

Project Summary

The recent times have clearly brought into light the importance of an optimized warehouse system. A well run warehouse system not only supports its consumers smoothly during normal days but it also makes sure that enough stock is maintained and important information is displayed about the inventory during calamities.

Apart from maintaining a good stock, a good warehouse system can also be optimized by taking into consideration its history of longevity of goods to reduce the amount of wastage. This becomes all the more crucial when a region is hit by a natural or man made disaster.

This project involves building a machine learning model to predict the demand of perishable raw materials for the next few weeks in order to optimize warehhouse management. The model takes into account the different types of holidays and the sales of the perishable goods in these holidays.

Some of the Regression based algorithms that have been used to make predictions are:

- Extra Trees Regression
- Random Forest Regression
- K-Neighbours Regression

The model that involves Python to code these regression techniques uses the Jupyter Notebook in IBM Watson Studio to import data and automate the ML model.A Node - RED flow is built to integrate the ML services.

Project Requirements:

The following are needed to successfully develop this project :
1. Datasets comprising of the perishable raw materials and factors influencing the demand of them in the coming weeks.
2. A regression based machine learning model coded in Python
3. A Git Hub account
4. An IBM Cloud account
5. A jupyter notebook in IBM Watson Studio to develop the Python codes in
6. A Node-RED flow application to integrate the model

Functional Requirements:

The following are the functionl requirements of this project:
1. A supervised Machine learning model based on Regression written in Python to process the dataset for the desired output.
2. An IBM Cloud service
3. An IBM Watson Studio service to create an ML model and automate it.
4. A Node-RED flow to integrate ML services

Software Requirements:

The following are the software requirements of this project:
1. Jupyter Notebook for developing python codes in
2. Git Tool
3. IBM cloud account
4. IBM Watson Studio service
5. Node-RED flow application

Project Deliverables:

A web application that can predict the demand of certain perishable raw materials for a warehouse by taking inputs like type of holidays,temperature etc.

Project Team:

The team includes Medha Chugh and Vinny Chamoli.

Project Schedule

Tasks	Duration
1. Register for IBM Academic Initiative and Create an IBM Cloud Account	0.5 Day
2. Create a Node-RED starter application	0.5 Day
3. Collect dataset for the project and modify it according to the project.	0.5 Day
4. Configure Watson studio and create a Machine Learning service	0.5 Day
5. Create A Jupyter Notebook In IBM Watson And Import Data	0.5 Day
6. Build A Machine Learning Model and Create Endpoints ForNode-RED Integration	3 Days
7. Build Node-RED Flow to Integrate ML Services	2 Days

Project Report

Optimized Warehouse Management Of Perishable Goods For A Food Delivery Company

1. Introduction

1.1 Overview

IBM Hack Challenge is an annual hackathon hosted by IBM at pan India level. It is an opportunity for students to show their coding skills, learn new technologies , create, build and deploy an efficient solution to the given problem on the cloud or otherwise.

IBM Hack Challenge 2020 was based around the current times where our society is surrounded with various problems like food shortage, transportation problems etc. The problem statement options were also around these issues.

Our project is based on the problem statement - **Optimized Warehouse Management Of Perishable Goods For A Food Delivery Company.**

1.2 Purpose

The recent times have clearly brought into light the importance of an optimized warehouse system. A well run warehouse system not only supports its consumers smoothly during normal days but it also makes sure that enough stock is maintained and important information is displayed about the inventory during calamities.

Apart from maintaining a good stock, a good warehouse system can also be optimized by taking into consideration its history of longevity of goods to reduce the amount of wastage. This becomes all the more crucial when a region is hit by a natural or man made disaster.

This project involves building a machine learning model to predict the demand of perishable raw materials for the next few weeks in order to optimize warehhouse management. The model takes into account the different types of holidays and the sales of the perishable goods in these holidays.

Various other services have been added in the developed UI like Add a new warehouse, remove warehouse, add a new product or a supplier and a chatbot to facilitate the warehouse operators.

2. Literature Survey

2.1. Existing Problem

The current system implements a successful marketing growth strategy that has doubled your customer base and sales order volume. Terrific! But before one can start celebrating well-earned success, the warehouse team notices that this new flood of sales orders has overwhelmed the current systems, slowing fulfillment productivity and causing shipments to be delayed. And if one is not using a multichannel inventory management solution, it is likely to oversold a number of items and one will be forced to inform the new customers that the items they've bought are actually out-of-stock, requiring even longer delays.

And that's just one common scenario. There are countless other potential business risks a company can incur from disorganized warehouse operations: everything from packaging errors and lost or damaged merchandise to mismarked items or even employee injury. All of these can slow your productivity, disrupt other processes in your supply chain, and ultimately eat away at your company's profits.

2.2. Proposed Solution

The proposed solution will allow the admin to login and manage different stores present at various regions of the country. The data collected on each store is helpful to determine the demand of particular perishable products in that region.The admin can add, update or remove store, supplier and store manager in the system.The store manager will be able to have access to all the products present in the store. The store manager will be responsible to maintain records of new products and orders made. All

the data of the sales is collected and stored in database as per weekly orders and sales done. Later, the machine learning model is used to check which products are needed to be ordered.

To make system effective and unique, different types of holidays are taken into account where the stores have huge number of sales due to discount offers. The machine learning helps the manager to fill the stocks according to the needs for such days.

3. Theoretical Analysis

3.1. Block Diagram

Work flow for this project can be divided into four sub-tasks.These include acquiring the data and understanding various features of the data, preprocessing the data set to align it with our requirements and remove any inconsistency, analyzing the data using regression based prediction algorithm with the key performance index being accuracy of prediction. and finally creating the UI for the model.

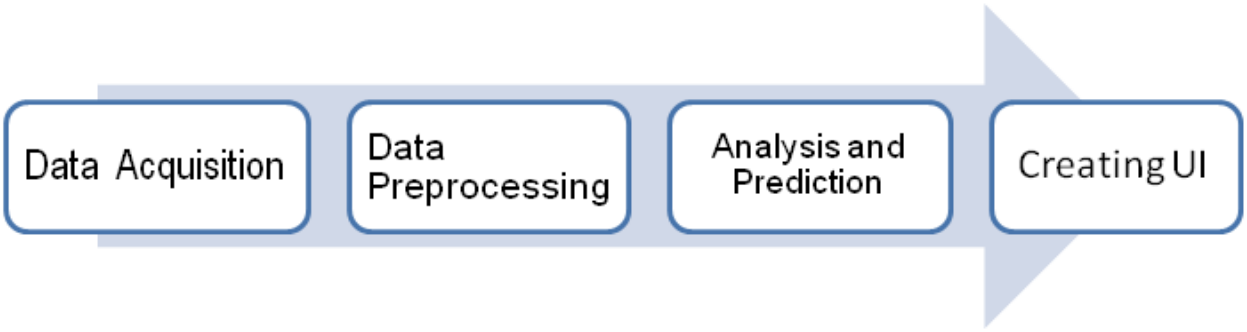


Figure 3.1 Workflow of the project

3.2 Software Designing

An IBM cloud account was set up to access various services to create and deploy the model.

The following services have been used in the project:

1. **Watson Studio** - This is where the notebook has been created in a project to write the regression code along with the data set.
2. **Watson Assistant** - This is where the chatbot was created.
3. **Node Red** - Node Red is the front end application that uses interconnecting nodes to interact with machine learning services of the cloud and the model to show predictions when inputs are given.

4. Experimental Investigations

4.1 Data Acquisition

A dataset was acquired from kaggle which had columns like store(warehouse number), dept(the department number to which a product might belong to), holiday, markdown sales(sales during different holidays), month, size of the store and temperature of the region and it was used with minor modification for the project.

4.2 Model Requirements

4.2.1 Python

Python is a multi-paradigm, general purpose, high level programming language, which focuses on code readability.
It has a large library, which provides tools for many tasks and has a wide support base.
This project
uses python 3.5.

4.2.2 Python Libraries

4.2.2.1 Pandas

Pandas is used for data manipulation and analysis through operations and data structures on numerical tables

and time series.

4.2.2.2 Numpy

It adds support as well as contains high-level mathematical functions to operate on large multidimensional arrays and matrices.

4.2.2.3 Matplotlib

It is a plotting library that that enables 2d diagramming and begetting of bar charts, histograms and so forth.

4.2.2.4 Sci-kit learn

It is a free software machine learning library that features various regression, clustering and classification algorithms. It works in conjunction with numPy and python scientific library sciPy.

4.3 Data Preprocessing

Data preprocessing is an essential step in order to increase the accuracy of machine learning models. It involves handling inaccurate and missing data, noisy data in the form of outliers, and inconsistent data in the form of duplication and others.

4.4 Data Cleaning

Data was often not consistent; missing values or values out of range was common. The methods used for cleaning is to replace the missing or noisy values by forward filling them using mean of the feature.

4.5 Analysis and Prediction

Some of the Regression based algorithms that have been used to analyze and make predictions are:

1. Extra Trees Regression
2. Random Forest Regression

3. K-Neighbours Regression

K - fold cross validation is applied to select the best model for training and testing the data then.

4.5.1. Random Forest Regression

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generatedon a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

4.5.2. Extra Trees Regression

In extremely randomized trees (ExtraTreesRegressor classes), randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of

these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.

4.5.3. K-Neighbours Regression

Neighbors-based regression can be used in cases where the data labels are continuous rather than discrete variables. The label assigned to a query point is computed based on the mean of the labels of its nearest neighbors.

Scikit-learn implements KNeighborsRegressor, a learning based on the nearest neighbors of each query point, where is an integer value specified by the user.

5. Flowchart

It works in four steps:

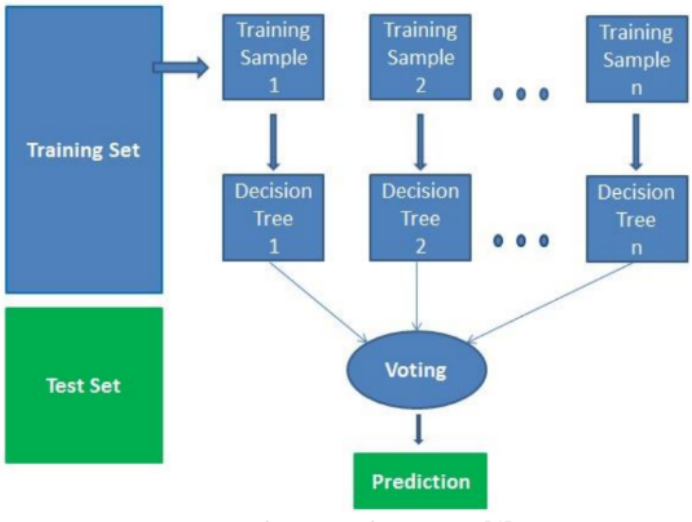
1) Select random samples from a given dataset.

2) Construct a decision tree for each sample and get a prediction result from each decision tree.

3) Perform a vote for each predicted result.

4) Select the prediction result with the most votes as the final prediction.

Figure 4.1: Workflow of the Random Forest Regression

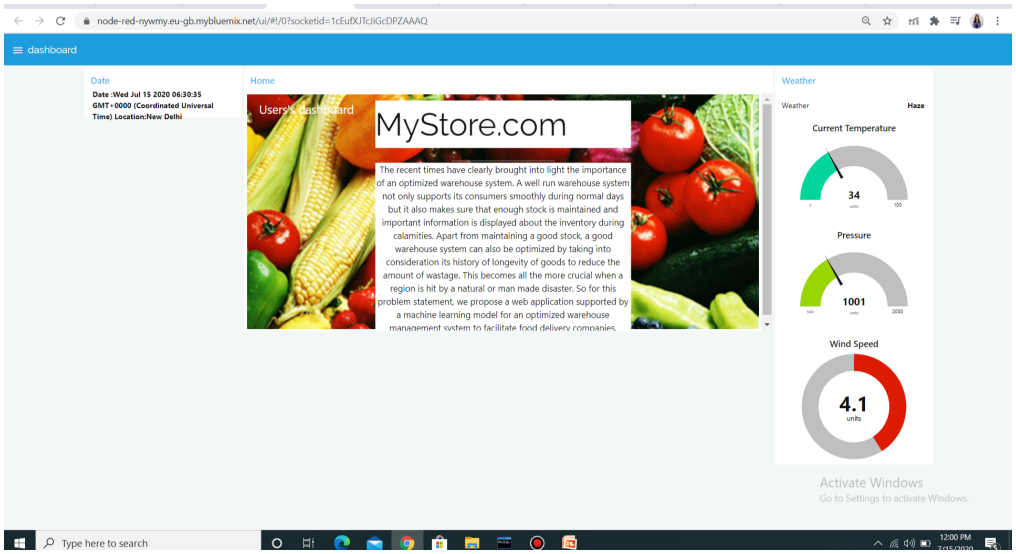


6. Result

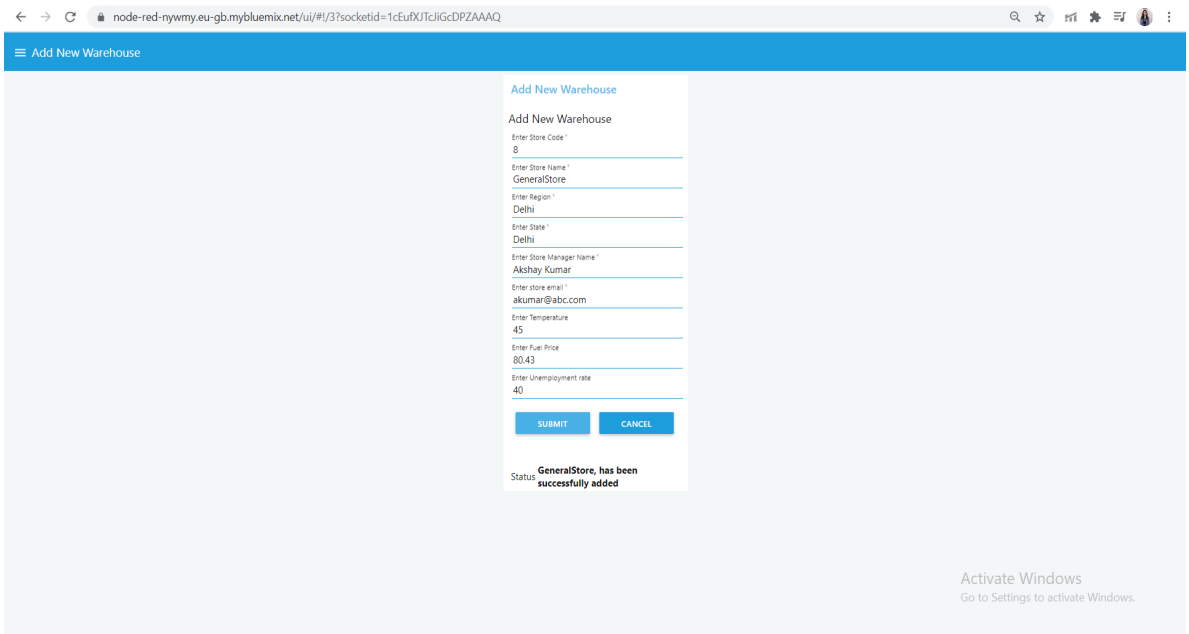
After the creation of Node-RED flows to build a UI, a form type interface was made for the machine learning model by using scoring endpoint and service credentials of machine learning instance of IBM. Various other databases were also created for tabs like add a new warehouse, add a new product etc. A chatbot was built with the help of Watson Assistant and was integrated with a form made in node-RED.

Screenshots of UI:

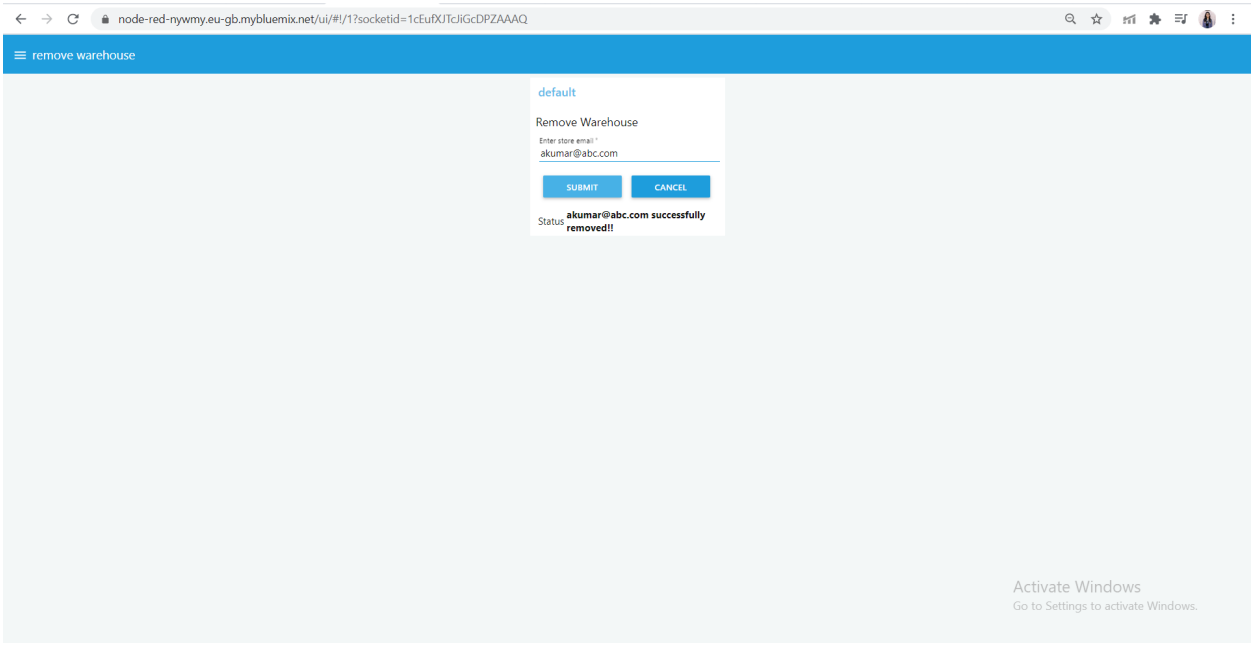
1. Dashboard



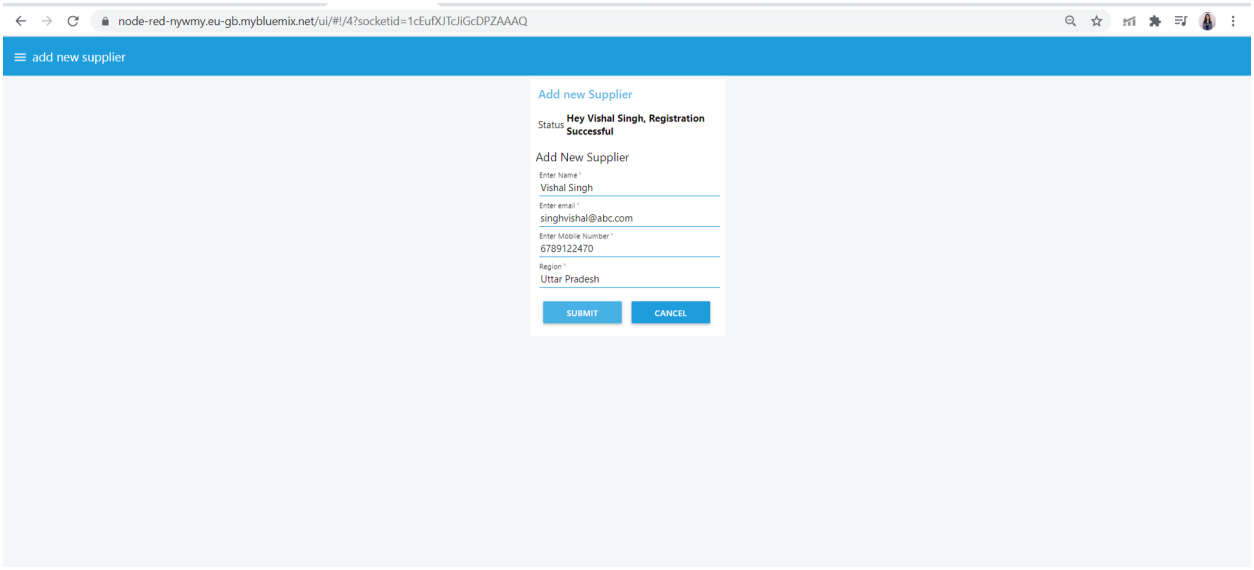
2. Add new warehouse



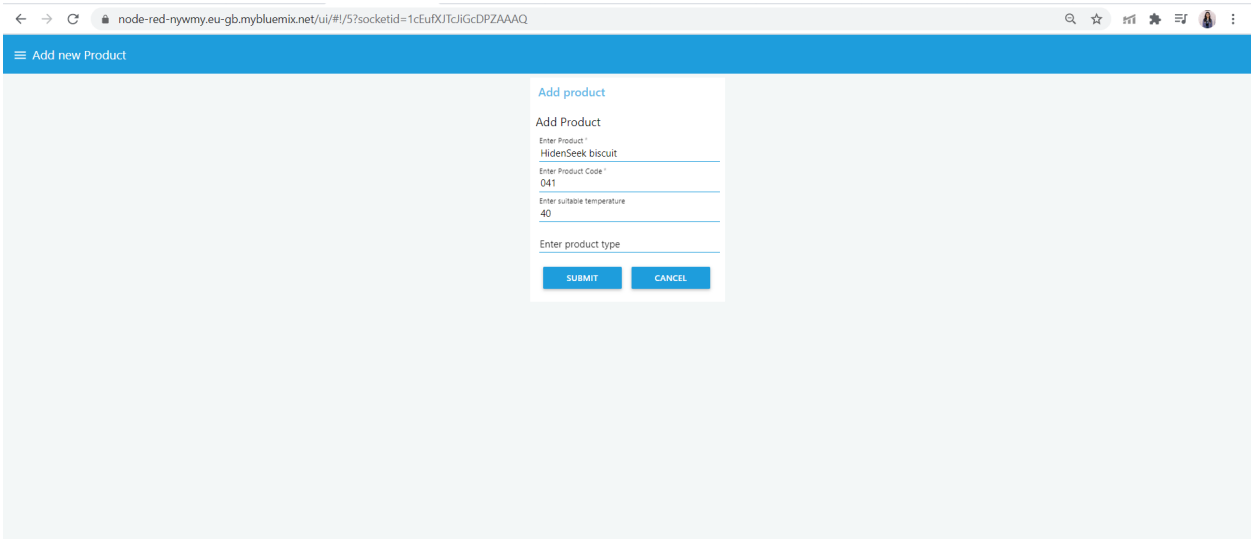
3. Remove warehouse



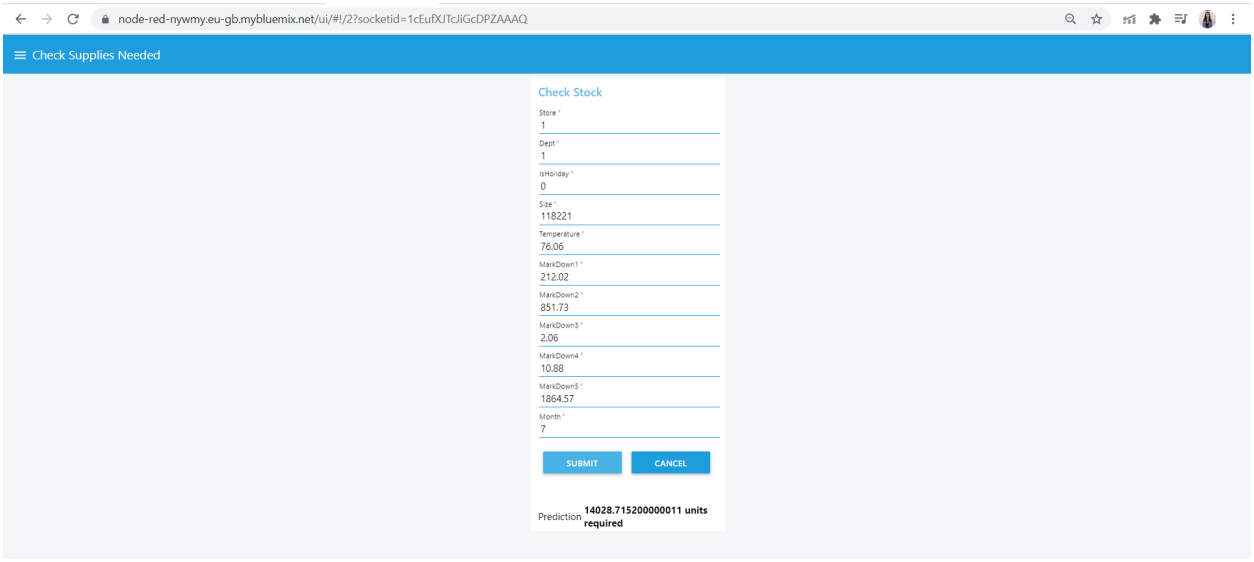
3. Add new supplier



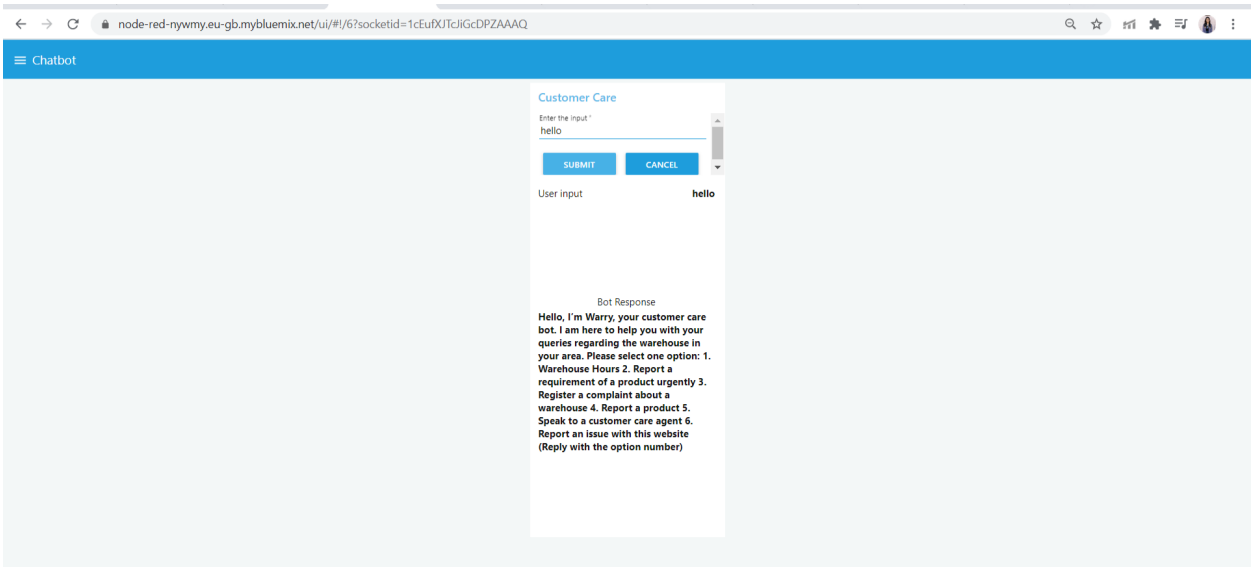
4. Add new product



5. Check supplies needed



6. Chatbot



7. Advantages and Disadvantages

Advantages:

- **Minimize errors, delays, and disruptions** – With the proper processes and technologies in place, one should be able to automate and streamline warehouse operations to avoid costly delays and reduce human and system errors.
- **Maximize productivity** – Establishing and maintaining an efficient flow of inbound and outbound activity throughout warehouse will increase the overall quantity of products, company can process and improve the speed at which you can ship them to customers.
- **Ensure compliance and security** – The phrase “safety first” is especially important in warehouses, which require workers to operate heavy machinery. It is needed to make sure one is operating in compliance with various supply chain vendors and partners to prevent any unnecessary roadblocks.
- **Provide supply chain visibility** – The more visibility vendors, and logistics partners have into the receiving, packaging, labeling, and shipping methods, the more smoothly operations will run.

Disadvantages:

- **Lack of Internet:** The current application uses cloud services and in case of internet issues, the application becomes useless.
- **Not accessible by all:** Not so educated people cannot work using the application and one has to be well versed with the computer to use the application.
- **Can't guarantee the stock numbers**: Incase of any disease or natural claamity like famine, the application cant give right predictions.

8. Applications:

Inventory Management Software – Since inventory is at the core of every warehouse operation, the most critical technology need is a comprehensive inventory management software. Manual processes are error-prone and not scalable. Fully integrated inventory control systems are required to not only push the right stock quantities out to your multiple sales channels, but to integrate with all your other eCommerce platforms and track your end-to-end retail processes.

Warehouse Automation Systems – Automated warehouse systems utilize conveyor belts and sortation and retrieval technologies to streamline sorting, routing, and packing processes throughout your facility. To aid in the efficiency and interoperability of these systems, you must integrate them with your other crucial warehouse technologies like inventory, barcoding, label printing, and shipping solutions.

Barcoding Solutions – You can make your warehouse employees more productive, gain greater visibility into your warehouse activity, and increase the value and accuracy of your inventory systems by utilizing barcoding technology. Barcode scanning can be used through every step of your warehouse operations, helping to ensure the right products get received, stored, picked, packed, and shipped to the right people in the most timely fashion. Barcode scanning technology is also useful for performing regular stock takes by allowing workers to quickly scan all the items in a box, counting what should be in the box and identifying items that might have been mistakenly placed in the wrong box. Top inventory management and warehouse automation solutions will integrate with barcoding systems to make this process even more efficient.

9. Conclusion:

The solution can help the food vendoring companies to handle the warehouses in a

more effiecient way by considering the holidays, temperature, weather conditions that can effect the stock of preishable goods and refill their stocks as and when needed. The model can predict the need of order 10 weeks before the stocks end. This way, its an efficient application where one can maintain all the records and predict the orders needed.

10. Future Scope:

Further, the analytics, and big data can be used to analyse the data collected at a very large level and get better insights, This data can be used for better learning. Real time data can be used to make model relearn and process data to give better results.
The occurence of natural calamities, drastic changes in weather conditions can be taken into account to optimise the storage of perishable goods.

11. Bibliography:

Appendix

A. Source Code

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns; sns.set(style="ticks", color_codes=True)
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesRegressor, RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0
```

```
14 import types
15 import pandas as pd
16 from botocore.client import Config
17 import ibm_boto3
18
19 def __iter__(self): return 0
20
21 # @hidden_cell
22 # The following code accesses a file in your IBM Cloud Object
   Storage. It includes your credentials.
23 # You might want to remove those credentials before you share
   the notebook.
24 client_bf71b2aaab5e4141bd379fd5c105495e =
   ibm_boto3.client(service_name='s3',
25
   ibm_api_key_id='4qPJVrD4ZRfknnVDE-q7JwH_FODU4qoX20F0_gzZBjP8'
   ,
26     ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
27     config=Config(signature_version='oauth'),
28
   endpoint_url='https://s3.eu-geo.objectstorage.service.network
   layer.com')
29
30 body =
   client_bf71b2aaab5e4141bd379fd5c105495e.get_object(Bucket='ib
   mhack-donotdelete-pr-oznyiafgasxlrr',Key='train.csv')['Body']
31 # add missing __iter__ method, so pandas accepts body as
   file-like object
32 if not hasattr(body, "__iter__"): body.__iter__ =
   types.MethodType( __iter__, body )
33
34 train = pd.read_csv(body)
35 train.head()
36 body =
   client_bf71b2aaab5e4141bd379fd5c105495e.get_object(Bucket='ib
   mhack-donotdelete-pr-oznyiafgasxlrr',Key='features.csv')['Bod
```

```
y']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

features = pd.read_csv(body)
features.head()
body = client_bf71b2aaab5e4141bd379fd5c105495e.get_object(Bucket='ibmhack-donotdelete-pr-oznyiafgasxlrr',Key='stores.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

stores = pd.read_csv(body)
stores.head()
body = client_bf71b2aaab5e4141bd379fd5c105495e.get_object(Bucket='ibmhack-donotdelete-pr-oznyiafgasxlrr',Key='test.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

test = pd.read_csv(body)
test.head()

from sklearn.feature_selection import RFE
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import KFold
import matplotlib.pyplot as plt
train = train.merge(stores, how='left').merge(features, how='left')
```

```
train
def scatter(train, column):
    plt.figure()
    plt.scatter(train[column] , train['Weekly_Sales'])
    plt.ylabel('weeklySales')
    plt.xlabel(column)
scatter(train,"Fuel_Price")
scatter(train,"Size")
scatter(train,'CPI')
scatter(train,'Type')
scatter(train,'IsHoliday')
scatter(train,'Unemployment')
scatter(train,'Temperature')
scatter(train,'Store')
scatter(train,'Dept')
fig = plt.figure(figsize=(18, 14))
corr = train.corr()
c = plt.pcolor(corr)
plt.yticks(np.arange(0.5, len(corr.index), 1), corr.index)
plt.xticks(np.arange(0.5, len(corr.columns), 1),
  corr.columns)
fig.colorbar(c)
for name, group in train.groupby(["Store", "Dept"]):
    plt.title(name)
    plt.scatter(range(len(group)), group["Weekly_Sales"])
    plt.show()
    break
```

Data Manipulation:

```
train=train.drop("Type", axis = 1)
train[['MarkDown1','MarkDown2','MarkDown3','MarkDown4',
'MarkDown5']] =
train[['MarkDown1','MarkDown2','MarkDown3','MarkDown4','Mar
kDown5']].fillna(0)
train['Month'] = pd.to_datetime(train['Date']).dt.month
train = train.drop(columns=["Date", "CPI", "Fuel_Price",
'Unemployment'])
train
```

Algorithms:

```
def knn():
    knn = KNeighborsRegressor(n_neighbors=10)
    return knn

def extraTreesRegressor():
    clf =
ExtraTreesRegressor(n_estimators=100,max_features='auto',
verbose=1, n_jobs=1)
    return clf

def randomForestRegressor():
    clf =
RandomForestRegressor(n_estimators=100,max_features='log2',
verbose=1)
    return clf

def svm():
    clf = SVR(kernel='rbf', gamma='auto')
    return clf

def nn():
    clf = MLPRegressor(hidden_layer_sizes=(10,),
```

```
    activation='relu', verbose=3)
    return clf

def predict_(m, test_x):
    return pd.Series(m.predict(test_x))

def model_():
#     return knn()
    return extraTreesRegressor()
#     return svm()
#     return nn()
#     return randomForestRegressor()

def train_(train_x, train_y):
    m = model_()
    m.fit(train_x, train_y)
    return m

def train_and_predict(train_x, train_y, test_x):
    m = train_(train_x, train_y)
    return predict_(m, test_x), m
def calculate_error(test_y, predicted, weights):
    return mean_absolute_error(test_y, predicted,
    sample_weight=weights)
```

K-folds cross validation

```
kf = KFold(n_splits=5)
splited = []
# dataset2 = dataset.copy()
for name, group in train.groupby(["Store", "Dept"]):
    group = group.reset_index(drop=True)
    trains_x = []
    trains_y = []
    tests_x = []
```

```
    tests_y = []
    if group.shape[0] <= 5:
        f = np.array(range(5))
        np.random.shuffle(f)
        group['fold'] = f[:group.shape[0]]
        continue
    fold = 0
    for train_index, test_index in kf.split(group):
        group.loc[test_index, 'fold'] = fold
        fold += 1
    splited.append(group)

splited = pd.concat(splited).reset_index(drop=True)
splited
best_model = None
error_cv = 0
best_error = np.iinfo(np.int32).max
for fold in range(5):
    dataset_train = splited.loc[splited['fold'] != fold]
    dataset_test = splited.loc[splited['fold'] == fold]
    train_y = dataset_train['Weekly_Sales']
    train_x = dataset_train.drop(columns=['Weekly_Sales',
'fold'])
    test_y = dataset_test['Weekly_Sales']
    test_x = dataset_test.drop(columns=['Weekly_Sales',
'fold'])
    print(dataset_train.shape, dataset_test.shape)
    predicted, model = train_and_predict(train_x, train_y,
test_x)
    weights = test_x['IsHoliday'].replace(True,
5).replace(False, 1)
    error = calculate_error(test_y, predicted, weights)
    error_cv += error
    print(fold, error)
    if error < best_error:
        print('Find best model')
```

```
        best_error = error
        best_model = model
error_cv /= 5
error_cv
best_error
```

Test Part

```
test = test.merge(stores, how='left').merge(features, how='left')
test
test=test.drop("Type", axis = 1)
test[['MarkDown1','MarkDown2','MarkDown3','MarkDown4', 'MarkDown5']] = test[['MarkDown1','MarkDown2','MarkDown3','MarkDown4','MarkDown5']].fillna(0)
test = test.fillna(0)
column_date = test['Date']
test['Month'] = pd.to_datetime(test['Date']).dt.month
test = test.drop(columns=["Date", "CPI", "Fuel_Price", 'Unemployment'])
test
predicted_test = best_model.predict(test)predicted_test = best_model.predict(test)
```